بسمه تعالی

وزارت ارتباطات و فناوری اطلاعات
سازمان فناوری اطلاعات ایران
معاونت امنیت فضای تولید و تبادل اطلاعات

مرکز ماهر
مرکز مدیریت امداد و هماهنگی عملیات رخدادهای رایانه ای

راهنمای امن سازی SSL/TLS

مرجع امنیتی

شناسه سند	MaherReportsTemplate_۱٤۰٤۱۰۰۷
نوع سند	گزارش فنی
شماره نگارش	۰.۱
تاریخ نگارش	۱۴۰۴/۱۰/۰۷
طبقه‌بندی سند	محرمانه

تهران، خیابان شهید بهشتی- بین بزرگراه شهید مدرس و خیابان احمد قصیر- پلاک۲۶۷

(۰۲۱) ۴۲۶۵۰۰۰۰ (۰۲۱) ۴۲۶۵۰۰۰۰ cert.ir

۱ فهرست مطالب

۲ مقدمه

پروتکل SSL(Secure Sockets Layer) و پروتکل TLS(Transport Layer Security) دو پروتکل برای ارسال و دریافت ایمن داده‌ها میان دو سیستم می‌باشند. کاربرد اصلی این پروتکل در ارسال و دریافت امن داده‌ها میان یک کلاینت و یک وب‌سایت می‌باشد، اما در حقیقت کاربرد این پروتکل‌ها بسیار گسترده‌تر بوده و در ارتباطات ایمیل سرورها، ارتباطات VPN، پیام‌رسان‌ها و همچنین FTPS و LDAPS استفاده می‌شوند. عموما در مدل OSI، این دو پروتکل را در لایه‌ی ششم (لایه نمایش) قرار می‌دهند. هر دو پروتکل، دارای نسخه‌های مختلفی هستند و TLS امروزه جانشین پروتکل SSL شناخته می‌شود، این پروتکل به دلیل مشکلات امنیتی مختلف و قابل توجه به فواید پروتکل TLS کنار گذاشته شد. امروزه نسخه‌های قدیمی‌تر پروتکل TLS نیز کنار گذاشته شده یا در حال کنار گذاشته شدن هستند؛ اکوسیستم دنیای وب با سرعت قابل توجهی حرکت می‌کند و نرم‌افزارهای مرتبط مانند مرورگرها نیز ناچار به پیروی از این استانداردهای جدید می‌باشند. با توجه به این که استفاده از این پروتکل‌ها محدود به صفحات وب نیست، ذکر این نکته لازم است که سرعت جایگزینی نسخه‌های قدیمی این پروتکل‌ها در اکوسیستمی مانند اکوسیستم ایمیل، پایین‌تر از اکوسیستم وب است و در نتیجه نسخه‌های قدیمی‌تر این پروتکل، برای مدت بیشتری در این پلتفرم‌ها استفاده می‌شوند.

بنابراین استفاده از بهترین پیکربندی ممکن برای TLS بسیار بااهمیت است. در این گزارش، موارد لازم برای پیکربندی صحیح امنیتی این پروتکل ارائه شده است.

۳ نحوه عملکرد پروتکل TLS

هر اتصال TLS میان یک کلاینت و سرور به نام یک نشست TLS شناخته می‌شود. هر نشست از دو فاز تشکیل شده، فاز دست‌تکانی و فاز برنامه.

در فاز دست تکانی، کلاینت و سرور بر روی نحوه ایجاد نشست توافق می‌کنند، الگوریتم‌های رمزنگاری مورد استفاده، نحوه اشتراک کلیدها و نسخه TLS مورد استفاده در کنار موارد دیگر مورد بررسی قرار می‌گیرند و در صورتی که کلاینت و سرور بتوانند بر روی نسخه و الگوریتم‌های مشترک توافق کنند، نشستی ایجاد خواهد شد. این فاز که توسط کلاینت آغاز می‌شود، سرور و کلاینت برای انتخاب چهار الگوریتم رمزنگاری مذاکره می‌کنند:

- الگوریتم شیوه تبادل و اشتراک کلید
- الگوریتم امضای دیجیتال

- الگوریتم رمزنگاری تمامی داده‌های اشتراکی
- الگوریتم اجرای فرآیند هشینگ

پس از انتخاب الگوریتم‌ها، کلاینت اصالت گواهی‌نامه ارسالی سرور را مورد بررسی و تایید قرار می‌دهد و اگر کلاینت نیز یک گواهی‌نامه به سرور ارسال کند، سرور آن‌را مورد بررسی قرار خواهد داد.

پس از این فاز، دست تکانی اتمام یافته و وارد فاز برنامه می‌شویم. در این مرحله، نشست TLS به عنوان کانالی امن برای انتقال داده‌ها در دسترس خواهد بود. برنامه‌ها در اینجا می‌توانند از این تونل برای ارسال ترافیک میان سرور و کلاینت بدون نیاز به درگیر شدن با شیوه کارکرد این تونل استفاده کنند و به عنوان یک کانال ارتباطی که محرمانگی و یکپارچگی اطلاعات را تضمین می‌کند از آن استفاده کنند.

٤ ملاحظات کلی امنیتی

برای حفظ امنیت این پروتکل لازم است نکاتی مد نظر قرار داده شوند. نخستین مورد، پروتکل و نسخه پروتکل مورد نظر است. پروتکل SSL امروزه بنابر استانداردهای امنیتی صرف نظر از نسخه مورد استفاده بطور کلی منسوخ شمرده شده و نباید مورد استفاده قرار گیرد. بنابراین همواره باید از پروتکل TLS و البته جدیدترین نسخه آن استفاده شود. اما مسئله به همین سادگی نیست، زیرا پشتیبانی سرور از جدیدترین نسخه TLS به این معنا نیست که کلاینت‌ها نیز توانایی پشتیبانی از جدیدترین نسخه لازم را داشته باشند و بنابراین در پیکربندی باید به این نکته دقت کرد که در صورت عدم توانایی پشتیبانی کلاینت‌های شبکه از جدیدترین نسخه، سرور پشتیبانی لازم برای نسخه‌های قدیمی‌تر را نیز ارائه کند. در زمینه چگونگی ارائه صحیح این نسخه‌های قدیمی، دستورالعمل‌هایی وجود دارد. بهتر است که در ابتدا به جدول زیر دقت شود:

Version	Status
TLS 1.3	Good (3; 4)
TLS 1.2	Sufficient (3; 4)
TLS 1.1	Phase out (3; 4)
TLS 1.0	
SSL 3.0	Insufficient (3; 4)
SSL 2.0	
SSL 1.0	

- مرحله «ناکافی» که با رنگ قرمز نمایش داده شده، به معنای این است که میزان امنیت این پیکربندی کافی نیست و نباید از آن استفاده شود. چنانکه می‌بینیم، طبق دیدگاه این سازمان پروتکل SSL را بطور کلی نباید استفاده کرد.

- مرحله «فاز خروج» یا «حذف تدریجی» که با رنگ نارنجی در جدول مشخص شده، به این معناست که این نسخه‌ها از امنیت کافی برای مقابله با حملات مدرن کافی نیستند و میزان اندکی امنیت ارائه کرده و در آینده وارد مرحله «ناکافی» خواهند شد. نسخه‌های ۱.۰و ۱.۱ پروتکل TLS با این درجه از ایمنی مشخص شدند و در حال حاضر همچنان برای ارائه خدمات به سیستم‌های بسیار قدیمی مورد نیازند. استفاده از این نسخه‌ها بسته به نیاز سازمان دارد و باید با دقت و با برنامه‌ای برای جایگزینی آن‌ها در آینده صورت پذیرد.
- مرحله «کافی» که تنها شامل نسخه ۱.۲ پروتکل TLS می‌شود، به این معناست که این نسخه در حال حاضر برای ایجاد امنیت کافی کفایت می‌کند و البته احتمالا کلاینت‌های زیادی همچنان وابسته به این نسخه بوده و در صورت درستی این گزاره برای یک سازمان، پروتکل TLS این سازمان باید به پشتیبانی از این نسخه ادامه دهد.
- در پایان، مرحله «خوب» با نسخه ۱.۳ پروتکل TLS قرار دارد، این نسخه بیشترین استانداردهای امنیتی را ارائه می‌کند و برای استفاده ایمن است. در صورتی که تمام سیستم‌های یک سازمان بتوانند از این نسخه استفاده کنند، آنگاه پروتکل سازمان باید تنها از همین نسخه پشتیبانی کند. طبیعی است که تمام این نسخه‌ها در آینده با پیدا شدن آسیب‌پذیری‌های بیشتر به سطوح پایین تر سقوط خواهند کرد.

بطور کلی پیشنهاد می‌شود که ترکیبی از «خوب» و «کافی» برای پشتیبانی از دستگاه‌ها استفاده شود و تنها از «فاز خروج» زمانی استفاده شود که نیاز به پشتیبانی از دستگاه‌های قدیمی‌تر بیشتری وجود دارد و همزمان بطور کلی تحت هیچ شرایطی از «ناکافی» که شامل پروتکل‌های SSL می‌شوند استفاده نشود.

نسخه پروتکل یکی از موارد امنیتی است که می‌بایست رعایت شود، باید دقت کرد که بیشتر نرم‌افزارها به دلیل پیچیدگی‌های بالای این پروتکل از کد شخصی خود استفاد نکرده و به جای آن از کتابخانه‌های آماده که این پروتکل را پیاده‌سازی کرده‌اند استفاده می‌کنند. برخی از این کتابخانه‌ها رایگان و برخی پولی و تجاری هستند و ممکن است که در سیستم‌عامل استفاده شده و یا بطور جداگانه عرضه شوند. از جمله کتابخانه‌های شناخته شده TLS می‌توان به OpenSSL, SCannel, NSS و mbes TLS اشاره کرد. راجع به ویژگی‌های امنیتی این کتابخانه‌ها تاکنون اظهار نظری نشده است، اما به طور کلی می‌توان گفت این کتابخانه‌ها تنظیمات متفاوتی دارند و هرکدام می‌توانند حاوی باگ‌هایی باشند که منجر به آسیب‌پذیری شوند. در نتیجه، انتخاب کتابخانه مناسب نیز مهم است و در پیکربندی یک پروتکل TLS امن، سازمان‌ها باید کتابخانه‌های مختلف را بررسی کرده و آنچه برای آن‌ها مناسب است را انتخاب کنند. بنابراین در کنار استفاده از بروزترین نسخه‌های TLS باید از بروزترین نسخه‌های کتابخانه‌های مربوطه نیز استفاده کرد.

طول کلیدها یا پارامترها که شامل کلیدهای RSA یا ECDSA می‌شود نیز جزو نکاتی است که باید مورد توجه قرار گیرد. برای برقراری ارتباط میان کلاینت و سرور، لازم است هر دو طرف از یک طول مشترک پشتیبانی کنند. نرم‌افزارهای قدیمی‌تر گاهی از طول کلید یا طول پارامتر کافی در سطح استاندارد نرم‌افزارهای مدرن استفاده نمی‌کنند و نرم افزارهای مدرن در صورتی که طول کلید و پارامترهای لازم کافی نباشد، گاهی جلوی این تعاملات را می‌گیرند.

پشتیبانی کتابخانه از قابلیت فشرده‌سازی داده‌ها که می‌تواند تنها در دو حالت فعال یا غیرفعال باشد، نیز می‌تواند در زمینه امنیت این پروتکل در سازمان تاثیرگذار باشد. فعال‌سازی فشرده‌سازی می‌تواند باعث رخ دادن تهدیدها و حملات خاصی برروی این پروتکل شود. درباره این موضوع در ادامه صحبت خواهد شد.

۵ انتخاب الگوریتم‌های رمزنگاری

همانگونه که اشاره شد، برای هر اتصال، کلاینت و سرور برای انتخاب چهار الگوریتم با یکدیگر مذاکره می‌کنند: یک الگوریتم برای تبادل کلید، یک الگوریتم برای امضای دیجیتال در جهت تایید گواهی‌نامه‌ها، یک الگوریتم برای برای رمزنگاری داده‌ها و یک الگوریتم برای انجام عملیات هشینگ. الگوریتم رمزنگاری داده‌ها و هشینگ در نسخه ۱.۳ به عنوان «مجموعه رمزگذاری»[۱] شناخته شده و برای حفاظت از رکوردها استفاده می‌شود. در نسخه‌های پیشین تا نسخه ۱.۲، این عبارت به الگوریتم تبادل کلید و الگوریتم امضای دیجیتال اشاره می‌کرد.

در زیر نمونه‌ای از این الگوریتم‌ها آورده شده است:

- **الگوریتم‌های اعتبارسنجی گواهی‌نامه:** RSA, ECDSA
- **الگوریتم‌های تبادل کلید:** ECDHE, DHE, RSA
- **الگوریتم‌های رمزنگاری کلی داده‌ها:** AES-GCM, ChaCha۲۰-Poly۱۳۰۵
- **الگوریتم‌های هشینگ:** SHA-۱, SHA-۲۵۶

به دلیل وجود الگوریتم‌های مختلف، هنگام مذاکره یک نشست TLS، صدها ترکیب صحیح مختلف می‌توانند برای اتصال در دسترس قرار داشته باشند، زیرا پیکربندی TLS می‌تواند از بیش از یک الگوریتم برای هرکدام

[۱] Cipher Suite

از چهار دسته بالا پشتیبانی کند. به همین دلیل پیشنهاد شده که سرور، الگوریتم‌های مورد استفاده خود را اولویت‌بندی کند. اگر کلاینت و سرور از بیش از یک الگوریتم پشتیبانی می‌کنند، سرور باید الگوریتم‌های ایمن‌تر و سریع‌تر را اولویت قرار داده و آن‌ها را برای ایجاد نشست پیشنهاد کند.

در سند سازمان NCSC [۱]، دسته بندی امنیت **کلی** این الگوریتم‌ها مشابه دسته‌بندی آن‌ها برای نسخه‌های TLS/SSL است، با این تفاوت که گزینه «ناکافی» وجود ندارد و بگونه کلی به آن پراخته نمی‌شود[۱]. این سند از چهار دسته الگوریتم بالا به عنوان دامنه یاد کرده و یک انتخاب **کلی** «خوب» در زمینه گزینش الگوریتم را انتخابی دانسته که در هر چهار دامنه، الگوریتمی با درجه «خوب» استفاده شود. به این معنی که الگوریتم‌های مورد استفاده در اعتبارسنجی گواهی‌نامه، تبادل کلید، رمزنگاری کلی داده‌ها و الگوریتم هشینگ همه الگوریتم‌هایی با درجه «خوب» باشند. از دیدگاه این سند، الگوریتم خوب الگوریتمی است که حداقل میزان امنیتی که ارائه می‌کند، به اندازه استفاده از یک کلید ۱۲۸ بیتی باشد. یک انتخاب **کلی** «کافی» از الگوریتم‌هایی با درجه ایمنی «کافی» و یا ترکیبی از الگوریتم‌های «کافی» و «خوب» برای هر دامنه تشکیل شده است. یک انتخاب **کلی** از الگوریتم‌های «فاز خروج» برای دامنه‌ها شامل الگوریتم‌هایی با درجه «فاز خروج» است که می‌توانند با الگوریتم‌هایی با درجه «ناکافی» و «خوب» نیز ترکیب شوند. بنابراین سطح کلی ایمنی الگوریتم‌های انتخابی در این چهار دامنه را می‌توان با پایین‌ترین سطح الگوریتم مورد استفاده در هرکدام از چهار دامنه سنجید. برای نمونه اگر از یک الگوریتم با درجه «فاز خروج» در دامنه تبادل کلید استفاده کنید، سطح ایمنی الگوریتم‌های شما در چهار دامنه، برابر با سطح ایمنی همین الگوریتم خواهد بود، حتی اگر سه الگوریتم انتخابی دیگر برابر با درجه «خوب» باشند. در جدول زیر درجه بندی امنیت این الگوریتم‌ها بر پایه این سند آورده شده است:

	Key exchange	Certificate verification	Bulk encryption	Hashing
Good	ECDHE	ECDSA RSA	AES_256_GCM CHACHA20_POLY1305 AES_128_GCM	(HMAC-)SHA-384 (HMAC-)SHA-256
Sufficient	DHE		AES_256_CBC AES_128_CBC	(HMAC-)SHA-1
Phase out	RSA*		3DES-CBC*	
	* Written as TLS_RSA_WITH_...		* Written as 3DES_EDE_CBC or DES_CBC3	

جدول بالا می‌تواند به عنوان یک راهنمای کلی برای انتخاب الگوریتم‌های امنیتی ایمن برای سازمان شما و همچنین سنجش وضعیت ایمنی فعلی پروتکل TLS مورد استفاده قرار گیرد.

---

[۲] این سند برای الگوریتم‌های رمزنگاری درجه ناکافی را در نظر گرفته، اما از آن برای دسته‌بندی امنیت کلی دامنه‌ها استفاده نمی‌کند.

۵-۱ الگوریتم‌های اعتبارسنجی گواهی‌نامه

برای اعتبارسنجی و تأیید گواهی‌نامه‌ها، از امضاهای دیجیتال استفاده می‌شود و اطمینان به اصالت یک اتصال نیازمند یک الگوریتم قابل اعتماد برای اعتبارسنجی گواهی‌نامه است. الگوریتمی که با آن گواهی‌نامه امضا می‌شود، توسط ارائه دهنده همان گواهی‌نامه تأمین می‌شود. گواهی‌نامه، الگوریتمی که برای امضای دیجیتال توسط ارائه دهنده آن استفاده شده را در مرحله **تبادل کلید** مشخص می‌کند. در صورت نیاز به پشتیبانی بیش از یک الگوریتم، این امکان وجود دارد که که چندین گواهی‌نامه مختلف را برای ارائه پیکربندی کرد. جدول زیر الگوریتم‌های اعتبارسنجی گواهی‌نامه و امنیت آن‌ها را مشخص می‌کند:

Algorithm	*Status*
ECDSA	Good (2; 3)
RSA	
DSS[24]	Insufficient
EXPORT-variants	
PSK	
Anon	
NULL	

۵-۲ الگوریتم‌های هش برای تأیید گواهی‌نامه‌ها

برای امضاهای دیجیتال گواهی‌نامه‌ها از توابع هشینگ استفاده می‌شود. امنیت الگوریتم‌های هش، به دلایلی مانند جلوگیری از برخورد و جلوگیری از دستکاری داده‌ها با اهمیت است. جدول زیر دسته‌بندی امنیتی این الگوریتم‌ها را نشان می‌دهد:

Algorithm	*Status*
SHA-512	Good (1; 3)
SHA-384	
SHA-256	
SHA-1	Insufficient (1; 3)
MD5	

۳-۵ الگوریتم‌های تبادل کلید

الگوریتم‌های تبادل کلید مشخص می‌کنند که یک کلاینت و سرور چگونه یک کلید را برای ارتباطات رمزگذاری شده خود انتخاب کنند. الگوریتم دیفی-هلمن[۱]، برای استخراج کلید اشتراکی موقت بین دو سمت تبادل استفاده می‌شود. این الگوریتم، انواع مختلفی بر اساس **میدان‌های متناهی (DHE)**[۲] و **منحنی‌های بیضوی(ECDHE)**[۳] دارد. یک اتصال TLS برای ایجاد یک کلید نشست با عملیات تبادل کلید آغاز می‌شود. الگوریتم‌های تبادل کلیدی که **رازداری رو به جلو**[۴] ارائه می‌کنند، اجازه می‌دهند حتی در صورت افشای کلید سرور، محرمانگی ارتباطات پیشین پایدار بماند[۵]. الگوریتم‌های **تبادل کلید ایستا** از کلید عمومی که در درون گواهی‌نامه تعبیه شده، برای انتقال رونوشت رمزگذاری شده کلید نشست استفاده می‌کنند. تبادل کلید بر پایه الگوریتم‌های ECDHE و DHE از قابلیت رازداری رو به جلو پشتیبانی می‌کنند، اما الگوریتم‌هایی ایستایی که بر پایه کلیدهای RSA, ECDH و DH در گواهی‌نامه‌ها می‌باشند از این قابلیت برخوردار نیستند. محرمانگی در این موارد بر این مبنا بنیان شده که کلیدهای طولانی مدت، همچنان محرمانه نگه داشته شوند. در این حالت مهاجم می‌تواند این کلید را به سرقت برده و ارتباطات رمزنگاری شده پیشین را رمزگشایی کند و محرمانگی ترافیک را به خطر بیندازد. اما در الگوریتم‌های ECDHE و DHE، کلیدها فقط مدت بسیار کوتاهی ذخیره می‌شوند و سپس از بین رفته و چیزی برای سرقت مهاجم باقی نمی‌گذارند.

---

[۱] Diffie-Hellman

[۲] Finite Fields

[۳] Elliptic Curves

[۴] Forward Secrecy

[۵] درباره این ویژگی در بخش‌های بعدی توضیح داده خواهد شد.

Algorithm	Status
ECDHE	Good (3)
DHE	Sufficient[26]
RSA	Phase out (2; 3)
DH[27]	Insufficient
ECDH[27]	
KRB5	
NULL	
PSK	
SRP	

۵-٤ الگوریتم‌های هش برای تبادل کلید

صاحبان گواهی‌نامه برای اثبات اینکه مالکیت کلید مخفی با گواهی‌نامه همخوانی دارد از یک امضای دیجیتال در مرحله تبادل کلید استفاده می‌کنند. ارائه دهنده، این امضای دیجیتال را با امضا کردن خروجی یک تابع هش ایجاد می‌کند. به این منظور، استفاده از یک الگوریتم هش ایمن بااهمیت است. در جدول زیر الگوریتم‌های مورد استفاده با دو دسته‌بندی «فاز خروج» و «خوب» آورده شده است، در این مورد خاص، پیشنهاد می‌شود که پشتیبانی از الگوریتم‌های «فاز خروج» قطع نشود، بلکه صرفا پشتیبانی از الگوریتم‌هایی با درجه «خوب» افزوده شود. این پیشنهاد به این دلیل است که TLS ۱،۲ به الگوریتم‌های دسته «فاز خروج» وابسته است.

SHA2 support for signatures	Status
Yes (SHA-256, SHA-384 or SHA-512 supported)	Good (3; 4)
No (SHA-256, SHA-384 or SHA-512 **not** supported)	Phase out (3; 4)

۵-۵ الگوریتم‌های رمزنگاری کلی داده‌ها

در فاز برنامه یک اتصال TLS، داده‌ها بصورت جمعی با استفاده از یک الگوریتم متقارن رمزنگاری می‌شوند. الگوریتم‌های رمزنگاری متقارن معمولا حاوی یک رمزگذار و یک مد عملیاتی هستند. در اینجا آن دسته از الگوریتم‌هایی که احرازهویت و رمزنگاری را در یک مد عملیاتی بصورت تنگانگ پیاده‌سازی می‌کنند باید در اولویت قرار گیرند. این الگوریتم‌ها که با نام AEAD شناخته می‌شوند، شامل الگوریتم‌های AES-GCM و

ChaCha۲۰-Poly۱۳۰۵ هستند. محبوب‌ترین الگوریتم رمزنگاری متقارن، الگوریتم AES است. پروتکل TLS از این الگوریتم با دو طول کلید ۱۲۸ بیت و ۲۵۶ بیت پشتیبانی می‌کند، اما پشتیبانی آن از الگوریتم ChaCha۲۰-Poly۱۳۰۵ تنها با یک طول کلید ۲۵۶ بیتی است. جدول زیر می‌تواند به عنوان راهنما برای انتخاب الگوریتم‌های ایمن و سنجش ایمنی الگوریتم‌های پیاده‌سازی شده مورد استفاده قرار بگیرد:

Algorithm	Status
AES-256-GCM	Good (3)
ChaCha20-Poly1305	
AES-128-GCM	
AES-256-CBC	Sufficient (4)
AES-128-CBC	
3DES-CBC	Phase out[32] (2; 3)
AES-256-CCM_8[33]	Insufficient
AES-128-CCM_8[32]	
IDEA	
DES	
RC4	
NULL	

الگوریتم‌های دیگری نیز وجود دارند که در جدول بالا نامشان آورده نشده است مانند الگوریتم AES-CCM-{۲۵۶،۱۲۸} با درجه «خوب»، الگوریتم CAMELLIA با درجه «کافی» و الگوریتم‌های SEED و ARIA با درجه «فاز خروج». این الگوریتم‌ها از جمله الگوریتم‌های دیگری می‌باشند که هرچند به ندرت استفاده می‌شوند، اما در صورت پشتیبانی سیستم از آن‌ها، بهتر است که مورد بررسی قرار گیرند.

۵-۶ الگوریتم‌های هش برای رمزنگاری کلی داده‌ها و تولید اعداد تصادفی

برخی از الگوریتم‌هایی که برای رمزنگاری کلی داده‌ها استفاده می‌شوند، از توابع هش برای ارائه اصالت داده‌ها (MAC) استفاده می‌کنند و از همین جهت امنیت تابع هش مورد استفاده مهم است[۱]. پروتکل TLS افزون بر این، از توابع هش انتخاب شده به عنوان یک بخش برای تولید اعداد تصادفی(PRF) استفاده می‌کند و در اینجا نیز امنیت الگوریتم انتخاب شده برای کارکرد صحیح این قابلیت مهم خواهد بود. به جدول زیر جهت انتخاب الگوریتم مناسب دقت کنید:

Algorithm	Status
HMAC-SHA-512	Good (3; 4)
HMAC-SHA-384	
HMAC-SHA-256	
HMAC-SHA-1	Sufficient (3; 4)
HMAC-MD5	Insufficient (3)

الگوریتم SHA-۲۲٤ نیز دارای درجه ایمنی «کافی» می‌باشد، اما چندان مورد استفاده قرار ندارد. همچنین باید دقت کرد که الگوریتم SHA-۱ تنها برای رمزنگاری کلی داده‌ها و به عنوان یک جزء از تولید اعداد تصادفی **کفایت** می‌کند، اما برای استفاده به عنوان امضاهای دیجیتال گواهی‌نامه‌ها، همانگونه که در بخش مربوطه دیدیم، **ناکافی** است.

۵-۷ ملاحظاتی درباره پیکربندی الگوریتم‌ها

امنیت الگوریتم RSA برای رمزنگاری و امضاهای دیجیتال به طول کلید عمومی مورد استفاده وابسته است. برای بیشتر وب سایت‌ها یک کلید ۲۰۴۸ بیتی که امنیتی معادل ۱۱۲ بیت بطور تقریبی فراهم می‌کند، کافی است. برای رسیدن به امنیت معادل ۱۲۸ بیت، کلیدی با طول ۳۰۷۲ بیت مورد نیاز است که به طور قابل توجهی کندتر خواهد بود. به جدول زیر برای انتخاب صحیح طول کلید دقت کنید:

---

[۱] الگوریتم‌های AEAD احراز هویت را بصورت تنگانگی پیاده‌سازی می‌کنند و بنابراین نیازی به یک تابع هش جداگانه برای حفاظت از داده‌ها ندارند. زمانی که یک مجموعه رمز که از یک الگوریتم AEAD استفاده می‌کند، شامل ارجاع به یک تابع هش می‌باشد، این تابع را تنها به عنوان یک جز برای تولید اعداد تصادفی استفاده می‌کند.

Length of RSA-keys	Status
At least 3072 bit	Good (2; 3)
2048 – 3071 bit	Sufficient (2)
Less than 2048 bit	Insufficient

در زمینه انتخاب **منحنی‌های بیضوی** به این نکته باید دقت کرد که تمام این الگوریتم‌ها مناسب استفاده در TLS نیستند. امنیت امضاهای دیجیتال ECDSA و EdDSA و تبادل کلید ECDHE بستگی به نوع **منحنی (پیاده‌سازی)** مورد استفاده دارد. جدول زیر رایج‌ترین آن‌ها را نمایش می‌دهد:

Elliptic curve	Status
secp384r1	Good (3; 4)
secp256r1	
curve 448	
curve 25519	
secp224r1	Phase out (3; 4)
Other curves	Insufficient

در زمینه **میدان‌های متناهی** نیز باید به مواردی دقت شود؛ امنیت تبادل کلید زودگذر دیفل-هلمن[۱] به طول کلید عمومی و خصوصی مورد استفاده در گروه میدان متناهی انتخاب شده بستگی دارد. کلیدهای بزرگ‌تری که برای الگوریتم DHE مورد نیازاند، با کاهش کارایی و سرعت همراه می‌شوند. از همین بابت پیشنهاد می‌شود که برای سرعت بیشتر از ECDHE استفاده شود. از سوی دیگر پیچیدگی‌هایی که با انتخاب آزاد *گروه‌های میدان متناهی* همراه است، در گذشته باعث رخ دادن آسیب‌پذیری شده است. TLS ۱.۳ تنها شامل شمار اندکی از **گروه‌های میدان متناهی** برای DHE می‌شود تا پیچیدگی مربوطه کاهش یابد. راهنمای زیر بر اساس آنچه برای نسخه ۱.۳ تعیین شده، آن دسته از **میدان‌های متناهی** که برای استفاده در تمام نسخه‌ها مناسبند را با درجه «کافی» تعیین می‌کند، دلیل عدم استفاده از درجه «خوب» تنها به دلیل سرعت پایین آن‌ها می‌باشد:

---

[۱] Dife-Hellman ephemeral key exchange (DHE) -

Finite field group	Status
ffdhe4096 (RFC 7919)	Sufficient[39] (4)
ffdhe3072 (RFC 7919)	
ffdhe2048 (RFC 7919)	Phase out
Other groups	Insufficient

۵-۸ سایر گزینه‌های پیکربندی

پروتکل TLS و کتابخانه‌های آن، گزینه‌هایی برای پیکربندی ارائه می‌دهند که برای حفظ امنیت بیشتر می‌بایست مد نظر قرار داده شوند. در ادامه به تشریح این گزینه‌ها می‌پردازیم:

۵-۸-۱ فشرده‌سازی

از قابلیت فشرده‌سازی برای کاهش حجم داده‌های ارسالی در این پروتکل پیش از رمزگذاری و ارسال آنها استفاده می‌شود. هدف از این‌کار افزایش سرعت با کاهش پهنای باند مصرفی است. با وجود مزایای فشرده‌سازی، از دیدگاه امنیتی اعمال این گزینه بی خطر نیست. استفاده از فشرده‌سازی می‌تواند به مهاجم اطلاعاتی درباره پیام رمزنگاری شده بدهد. مهاجمی که بتواند بخش‌هایی از داده‌های ارسالی را تعیین یا آنها را کنترل کند می‌تواند داده‌های اصلی ارسال شده را با اجرای حجم بالایی از درخواست‌ها بازسازی کند. داده‌ها زمانی که حاوی تکرارهای بیشتری باشند، نسبت به داده‌هایی که حاوی مقادیر تکراری نیستند از فشرده‌سازی بهتری برخوردارند. بر همین مبنا مهاجم می‌تواند یک بخش شناخته شده که بیشتر در داده‌های رمزنگاری شده ارسالی تکرار شده را شناسایی کند و بدین نحو استفاده از فشرده‌سازی می‌تواند امنیت پروتکل را به طور منفی تحت تاثیر قرار دهد.

قابلیت فشرده سازی TLS امروزه به اندازه‌ای به ندرت مورد استفاده قرار می‌گیرد که غیرفعال کردن آن عموما مشکلی ایجاد نمی‌کند. دقت کنید که فشرده سازی در این پروتکل متفاوت از فشرده‌سازی‌های در سطح برنامه است. برای نمونه فشرده‌سازی در پروتکل HTTP اغلب با استفاده از الگوریتم‌هایی مانند gzip برای استفاده بهینه‌تر از پهنای باند مورد استفاده قرار می‌گیرد. در صورتی که تصمیم گرفتید از فشرده‌سازی سطح برنامه استفاده کنید، پیش از استفاده اطمینان حاصل کنید که می‌توانید ریسک حملات مرتبط در سطح برنامه را کاهش دهید. یک نمونه از کارهایی که می‌توان در این زمینه انجام داد، محدود کردن میزانی است که یک مهاجم می‌تواند پاسخ سرور را تحت تاثیر و نفوذ خود قرار دهد. به جدول زیر برای تصمیم‌گیری دقت کنید، دلیل اینکه استفاده از فشرده‌سازی‌های سطح برنامه مانند فشرده‌سازی در سطح HTTP در درجه «کافی» و نه «خوب» قرار داده شده، این است که این نوع فشرده‌سازی باعث می‌شود تا پیکربندی TLS

نسبت به حمله BREACH آسیب‌پذیر باشد، اما همزمان غیرفعال کردن این ویژگی می‌تواند برروی کارایی و سرعت سیستم تاثیر منفی بگذارد. در حالی که فعال بودن فشرده‌سازی TLS باعث می‌شود پیکربندی TLS نسبت به حمله CRIME آسیب‌پذیر باشد.

Compression option	*Status*
No compression	Good
Application-level compression	Sufficient[40]
TLS compression	Insufficient[41]

۵-۸-۲ مذاکره مجدد

نسخه‌های قدیمی‌تر پروتکل TLS، یعنی نسخه‌های قدیمی‌تر از ۱.۳، اجازه می‌دهند عملیات دست‌تکانی جدید به اجبار انجام شود. این کار که **مذاکره مجدد**[۱] نامیده می‌شود، در طراحی اولیه و اصلی خود ناامن است. با توجه به این مسئله، استاندارد جدیدی طراحی و مکانیزم امن‌تری برای مذاکره مجدد اضافه شد. نسخه قدیمی، از این پس به نام **مذاکره مجدد نا امن**[۲] شناخته شده و می‌بایست غیرفعال شود. اما بطور کلی اجازه دادن به کلاینت‌ها برای انجام عملیات **مذاکره مجدد** ضروری نبوده و سرور را نسبت به حملات منع سرویس(DOS) در درون یک اتصال TLS آسیب‌پذیر می‌کند. البته یک مهاجم می‌تواند حملات مشابهی بدون انجام عملیات **مذاکره مجدد** از سمت کلاینت و با باز کردن مقدار زیادی اتصالات TLS موازی انجام دهد، اما تشخیص و دفاع در برابر این حملات با استفاده از روش‌های استاندارد کاهش ریسک ساده‌تر است.

Insecure renegotiation	*Status*
Off[42] (or N/A for TLS 1.3)	Good
On	Insufficient

[۱] Renegotiation

[۲] Insecure Renegotiation

Client-initiated renegotiation	Status
Off (or N/A for TLS 1.3)	Good
On	Sufficient

۵-۸-۳ ۰-RTT

نسخه‌های قدیمی‌تر TLS حداقل از دو سیکل رفت و برگشت[۱] پیام میان کلاینت و سرور پیش از آنکه داده‌های برنامه‌ها منتقل شوند، استفاده می‌کنند. نسخه ۱.۳ این مقدار از سربار را با نیاز به فقط یک سیکل به نصف کاهش می‌دهد. ۰-RRT گزینه‌ای در نسخه جدید TLS می‌باشد که می‌تواند سربار را از بیش از این کاهش دهد. با استفاده از این گزینه داده‌های برنامه در همان دست‌تکانی نخست انتقال می‌یابند. اما اینکار یک نکته منفی دارد و آن عدم حفاظت ۰-RTT در برابر حملات مجدد پاسخ‌دهی[۲] در سطح لایه TLS است و بنابراین استفاده از آن در محیطی که برنامه‌های متنوع و بسیاری وجود دارند دشوار خواهد بود، زیرا یک ساز و کار دفاعی جداگانه باید برای هر برنامه پیاده سازی شود.

Off (or N/A prior to TLS 1.3)	Good
On	Insufficient

۵-۸-٤ OCSP Stapling

کلاینت TLS می‌تواند اعتبار گواهی‌نامه X.۵۰۹ که سرور با تماس با ارائه‌دهنده گواهی‌نامه با استفاده از پروتکل OCSP ارائه می‌دهد را تأیید کند. پروتکل OCSP به ارائه دهنده گواهی‌نامه، اطلاعات مرتبط با کلاینت‌هایی که با سرور ارتباط برقرار می‌کنند را ارائه می‌دهد. این کار می‌تواند حریم خصوصی را به خطر بیندازد. یک سرور می‌تواند پاسخ‌های OCSP خودش را ارائه دهد (OCSP Stapling). این‌کار مشکل حریم خصوصی را حل کرده، نیاز به ارتباط مستقیم میان کلاینت و ارائه دهنده گواهی‌نامه نداشته و سریع‌تر است.

---

[۱] Round Trips

[۲] Replay

OCSP stapling	Status
On	Good
Off	Sufficient

۵-۹ تاملات دیگر

۵-۹-۱ رازداری رو به جلو

رازداری رو به جلو مکانیزمی است که محرمانگی ارتباطات قدیمی در پروتکل TLS را حتی در صورت به سرقت رفتن کلید خصوصی سرور حفظ می‌کند. همان‌طور که گفته شد، پیکربندی‌هایی که از الگوریتم‌های ECDHE یا DHE برای تبادل کلید استفاده می‌کنند، این قابلیت را فراهم می‌کنند. در صورت استفاده از رازداری رو به جلو، کلاینت و سرور به‌طور مستقیم از کلیدهای اصلی خود برای رمزنگاری داده‌ها استفاده نمی‌کنند. بلکه طبق توافق بین کلاینت و سرور، یک کلید موقتی و جداگانه برای رمزنگاری در نشست فعلی ایجاد می‌شود. این کلید موقتی فقط برای همان نشست معتبر است و پس از پایان ارتباط، تمامی مقادیر مربوط به آن به‌طور کامل حذف می‌شوند. بنابراین، کلید موقتی مورد استفاده نمی‌تواند با استفاده از کلید خصوصی گواهی‌نامه بازیابی شود.

بدون وجود قابلیت رازداری رو به جلو، کلیدهای سرور (کلیدهایی که با گواهی‌نامه مطابقت دارند) به‌طور مستقیم برای تبادل کلیدهای نشست استفاده می‌شوند. سناریویی که این قابلیت از آن محافظت می‌کند به دو مرحله تقسیم می‌شود: ابتدا مهاجم باید ارتباطات محافظت‌شده توسط TLS را شنود کند. سپس مهاجم باید به کلید خصوصی سرور که با کلید عمومی موجود در گواهی‌نامه همخوانی دارد دسترسی پیدا کند؛ این دسترسی می‌تواند از طریق نفوذ به سرور یا با دستور قانونی به دست آید. پس از به‌دست آوردن کلید خصوصی، مهاجم قادر خواهد بود کلید نشست را استخراج کرده و ترافیک رمزنگاری‌شده را رمزگشایی کند، که در نتیجه محرمانگی ارتباطات شکسته و از بین می‌رود.

۵-۹-۲ تیکت‌های نشست

در بسیاری از برنامه‌ها رایج است که کلاینت و سرور پس از آنکه اتصال اولیه TLS برقرار شد، اتصالات بیشتری را ایجاد کرده و یا دوباره از سر بگیرند. در TLS ساز و کاری وجود دارد که اجازه می‌دهد تا کلاینت و سرور نشست پیشین خود را هنگام اتصال دوباره حفظ کرده و به جای انجام دست تکانی TLS بطور مستقیم به فاز برنامه بروند. این‌کار هزینه راه‌اندازی نشست TLS برای اتصالات اضافه را کاهش می‌دهد. با

استفاده از شناسانگر نشست‌ها[۱]، وضعیت ذخیره‌شده نشست کلاینت و سرور به یک شناسانگر مشخص ارجاع داده می‌شود. هنگامی که کلاینت درخواست ازسرگیری نشست را ارسال می‌کند، این شناسانگر را به سرور ارائه می‌دهد. سرور نیز با استفاده از این شناسانگر، وضعیت نشست مرتبط را بازیابی می‌کند و بدین ترتیب کلاینت و سرور می‌توانند بدون انجام مجدد دست‌تکانی کامل TLS ، مستقیماً به ادامه ارتباط در فاز برنامه بپردازند.

تیکت‌های نشست مشابه شناسانگرهای نشست هستند، با این تفاوت که تیکت نشست در واقع یک نسخه رمزنگاری‌شده از وضعیت نشست است که در کلاینت ذخیره می‌شود. این روش باعث می‌شود که سرور نیازی به ذخیره شناسانگر و وضعیت نشست هر کلاینت نداشته باشد. در این حالت، کلاینت تیکت نشست را نگه می‌دارد و هنگام ازسرگیری نشست آن را به سرور ارائه می‌کند. سرور تیکت نشست را رمزگشایی کرده، وضعیت نشست را استخراج و اتصال TLS پس از انجام این عمل بطور مستقیم به فاز برنامه می‌رود. برای انجام این عملیات سرور می‌بایست یک «کلید رمزنگاری تیکت نشست» را در سرور ذخیره نماید. طراحی تیکت‌های نشست در نسخه‌های ۱.۰، ۱.۱ و ۱.۲ پروتکل TLS آسیب‌پذیر است. مهاجمی که بتواند کلید رمزنگاری تیکت نشست را به سرقت ببرد، می‌تواند بطور پسیو و غیرفعالانه تمام اتصالاتی که از تیکت نشست استفاده کرده و یا دست به تبادل آن با سرور می‌زنند را رمزگشایی کند و این‌کار حتی باعث از بین رفتن قابلیت *رازداری رو به جلو* می‌شود. این مشکل در نسخه ۱.۳ حل شده و NCSC به سازمان‌هایی که می‌خواهند فرآیند از سرگیری نشست‌ها را سرعت ببخشند، پیشنهاد می‌کند که از نسخه ۱.۳ استفاده کنند. در صورت استفاده از نسخه‌های قدیمی پروتکل TLS و تمایل به استفاده از قابلیت تیکت‌های نشست، نباید «کلید رمزنگاری تیکت نشست» را برروی دیسک ذخیره کرد و همچنین باید بطور مداومی کلید را تغییر داد تا در صورت افشا، اتصالات کمتری به خطر بیفتند.

۳-۹-۵ مولد اعداد تصادفی

برای اینکه از امنیت یک الگوریتم رمزنگاری اطمینان حاصل کنیم، به یک منبع خوب برای داده‌های تصادفی(آنتروپی[۲]) و یک مولد خوب اعداد تصادفی(PRNG[۳]) نیازمند خواهیم بود. منبع آنتروپی، داده‌های تصادفی اولیه را فراهم کرده و آن‌ها را به عنوان ورودی به مولد اعداد شبه‌تصادفی می‌دهد. سپس مولد، این

---

۱ Session IDs

۲ Entropy

۳ Pseudo Random Number Generator

داده‌ها را به اعداد تصادفی با توزیع یکنواخت تبدیل می‌کند. در پروتکل TLS، تولید اعداد تصادفی در زمینه‌های مختلفی کاربرد دارد که مهم‌ترین آن‌ها به شرح زیر است:

- تولید کلیدها و گواهی‌نامه‌ها
- تولید کلیدهای موقتی و اثبات در دست داشتن کلید مخفی برای قابلیت رازداری رو به جلو

تولید آنتروپی کافی برای اجرای عملیات مورد نیاز پروتکل TLS ممکن است در شرایط فشار زیاد روی سرور باعث ایجاد گلوگاه[۱] شود. استفاده از افزونه‌های سخت‌افزاری تولید اعداد تصادفی در سرور می‌تواند اطمینان حاصل کند که آنتروپی لازم به اندازه کافی در دسترس باشد. بسیاری از پردازنده‌های مدرن دارای واحدهای ویژه‌ای برای استخراج داده‌های تصادفی هستند. همچنین، بیشتر سیستم‌عامل‌ها و کتابخانه‌های TLS از مولدهای اعداد شبه‌تصادفی قابل اطمینانی استفاده می‌کنند. برای اطمینان بیشتر، پیشنهاد می‌شود از ارائه‌دهنده کتابخانه TLS یا سازنده محصول مربوطه درباره نوع مولد اعداد تصادفی و منبع آنتروپی استفاده شده پرس‌وجو کنید. همچنین لازم است بدانید که مولد اعداد تصادفی Dual EC DRBG به‌طور مشخص ناامن است؛ بنابراین مطمئن شوید که کتابخانه TLS شما از این مولد استفاده نمی‌کند.

۱- مدیریت گواهی‌نامه

مدیریت مناسب گواهی‌نامه‌ها یک شرط مهم برای برای پیاده‌سازی ایمن TLS می‌باشد . برخی از نکات مهم در همین رابطه در زیر آورده شده:

- **مولد کلید مخفی**: برای تولید کلید مخفی، از یک مولد اعداد تصادفی مطمئن استفاده کنید و اطمینان حاصل کنید که این کلید در سیستمی امن و مورد اعتماد تولید می‌شود. برای نمونه یک افزونه امنیتی سخت‌افزاری (HSM[۲]) یا رایانه‌ای که بطور فیزیکی به اینترنت متصل نیست، گزینه مناسبی می‌باشد. در این روش، کلید در سیستمی که به اینترنت متصل نیست تولید شده و سپس در سرور ارائه‌دهنده گواهی‌نامه پیاده‌سازی می‌شود.

---

[۱] Bottleneck

[۲] Hardware Security Module

- **ارائه دهنده گواهی‌نامه:** یک ارائه دهنده گواهی‌نامه قابل اعتماد برای ارائه و امضای گواهی‌نامه‌ها انتخاب کنید.
- **نام دامنه‌ها:** گواهی‌نامه حاوی فهرستی از نام دامنه‌هایی (FQDNs[۱]) می‌باشد که برای آن دامنه‌ها معتبر است. مطمئن شوید که گواهی‌نامه تمام دامنه‌ها و زیر دامنه‌هایی که قصد استفاده ار آن‌ها را دارید را پوشش می‌دهد. همچنین بررسی کنید که گواهی‌نامه در هر دو حالت با پیشوند www و بدون آن، محافظت کاملی از وب‌سایت شما فراهم می‌کند.
- **اعتبارسنجی افزوده:** بسیاری از صادرکنندگان گواهی‌نامه، گواهی‌نامه‌هایی با اعتبارسنجی افزوده (EV)نیز ارائه می‌کنند. این نوع گواهی‌نامه‌ها سطح اطمینان بیشتری درباره هویت صاحب گواهی‌نامه فراهم می‌کنند، زیرا صادرکننده پیش از صدور، سازمان مربوطه را به‌طور کامل بررسی و تأیید می‌کند تا ارتباط وب‌سایت با آن سازمان را تضمین کند. این گواهی‌نامه‌ها معمولاً هزینه بیشتری نسبت به گواهی‌نامه‌های عادی دارند. با این حال، باید توجه داشت که توسعه‌دهندگان مرورگرهای مختلف به مرور زمان اقدام به حذف یا پنهان‌سازی نشانگرهای مرتبط با گواهی‌نامه‌های EVکرده‌اند؛ پیش از این تغییرات، این نشانگرها در نوار آدرس نمایش داده می‌شدند، اما طبق گفته توسعه‌دهندگان، اکثر کاربران به این نشانه‌ها توجه چندانی نمی‌کردند. بنابراین، قبل از خرید گواهی‌نامه EV[۲] و صرف هزینه بیشتر، لازم است بررسی شود که آیا این نوع گواهی‌نامه واقعاً به بهبود امنیت سازمان کمک می‌کند یا خیر.
- **فایل‌های برروی سرور:** مدیر سرور TLS باید گواهی‌نامه‌های مراجع واسطه صدور گواهی‌نامه دیجیتال را بین گواهی‌نامه مرجع اصلی[۳] و گواهی‌نامه صادر شده برای سرور قرار دهد. این مراجع واسطه هنگام برقراری اتصال TLS توسط سرور به کلاینت ارائه می‌شوند که موجب افزایش امنیت و اعتمادپذیری ارتباط خواهد شد. همچنین، باید توجه ویژه‌ای به حفاظت از کلید خصوصی یا کلید پنهان داشت؛ زیرا مهاجمی که به این کلید دسترسی پیدا کند، قادر است ترافیک شنود شده را مشاهده یا دستکاری کند. برای افزایش امنیت، می‌توان کلید خصوصی را روی یک افزونه امنیتی سخت‌افزاری (HSM) ذخیره کرد که به‌گونه‌ای طراحی شده تا حفاظت فیزیکی مناسبی در برابر استخراج کلید فراهم کند. لازم به ذکر است که برخی مراجع صدور گواهی‌نامه پیشنهاد تولید

---

[۱] Fully Qualifed Domain Names

[۲] Extended Validation (EV) certifcates

[۳] Root CA

کلیدهای مخفی را به مشتریان خود می‌دهند؛ بهتر است از این مراجع خودداری کنید و کلیدهای خصوصی را خودتان تولید کنید.

- **مدیریت:** برای تمامی گواهی‌نامه‌هایی که در داخل سازمان استفاده می‌شوند، یک مدیر مسئول تعیین کنید. این کار فرآیند شناسایی محل استفاده هر گواهی‌نامه را در صورت نیاز به جایگزینی ساده‌تر می‌کند. همچنین، تاریخ انقضای همه گواهی‌نامه‌ها باید در دسترس باشد تا بتوان به موقع نسبت به تمدید یا تعویض آن‌ها اقدام کرد. استفاده از گواهی‌نامه‌های منقضی شده هرگز مجاز نیست. برخی ارائه‌دهندگان سرویس، مکانیزم‌هایی برای تمدید و استقرار خودکار گواهی‌نامه‌ها ارائه می‌دهند که می‌تواند احتمال بروز خطاهای انسانی را به‌طور قابل توجهی کاهش دهد.

۲- نقطه پایانی اتصال TLS

شیوه اتصال یک کلاینت به سرور با پیکربندی‌ای که در بسیاری از سازمان‌ها مورد استفاده است، همخوانی ندارد. برای نمونه، رمزگشایی ترافیک TLS ممکن است پیش از مسیریابی داخل شبکه داخلی به صورت متمرکز انجام شود. این نوع پیکربندی اجازه می‌دهد تا ترافیک شبکه پس‌پردازش شود، اما در این حالت، پروتکل TLS تنها تا نقطه پایان اتصال، از ترافیک محافظت می‌کند. اگر نیاز دارید که یکپارچگی و محرمانگی ترافیک در داخل شبکه سازمان حفظ شود، باید اقدامات بیشتری انجام دهید. یکی از راهکارها می‌تواند راه‌اندازی یک نشست TLS جدید برای بخش داخلی شبکه باشد.

برخی از سازمان‌هایی که راهکارهای امنیتی برای مقابله با حملات DDoS ارائه می‌دهند، از سازمان‌ها درخواست می‌کنند کلیدهای خصوصی گواهی‌نامه‌های TLS مورد استفاده را در اختیارشان قرار دهند. اگر قصد استفاده از چنین سرویس‌هایی را دارید، هرگز به‌سادگی کلید خصوصی خود را تحویل ندهید و ترجیحاً از ارائه‌دهندگانی استفاده کنید که نیازی به دریافت کلیدهای خصوصی شما ندارند. در صورتی که مجبور به استفاده از سرویسی هستید که کلیدهای شما را درخواست می‌کند، ابتدا ریسک‌های مرتبط با این اقدام را به‌دقت ارزیابی کنید، سپس در قرارداد خود مفادی برای جبران کاهش کنترل بر شبکه سازمان لحاظ نمایید. پس از آن، عملکرد ارائه‌دهنده را به‌صورت مستمر بررسی کنید تا اطمینان حاصل شود که مفاد توافق‌شده به درستی اجرا می‌شود.

علاوه بر نکات مطرح شده، هنگام استفاده از اتصال امن TLS باید مطمئن شوید که تمام داده‌ها به‌طور کامل رمزنگاری شده‌اند. صفحات وبی که از طریق TLS منتقل می‌شوند ممکن است حاوی منابعی مانند فایل‌های جاوااسکریپت، تصاویر یا CSS باشند که به صورت غیررمزنگاری‌شده بارگذاری می‌شوند. این موضوع باعث کاهش امنیت صفحه می‌شود. به عنوان مثال، مهاجم می‌تواند از طریق حمله مرد میانی با استفاده از یک منبع جاوااسکریپت محافظت‌نشده، اقدام به سرقت نشست کاربر کند. در مواقعی که از منابع خارجی مانند

فایل‌های جاوااسکریپت از منابع دیگر استفاده می‌کنید، به‌کارگیری مکانیزم‌هایی مانند سیاست امنیت محتوا (CSP[۱]) می‌تواند به بهبود امنیت کمک کند.

۳- امنیت پس کوانتومی

نسخه‌های عادی TLS در برابر حملات کوانتومی مقاوم نیستند، اما این امکان وجود دارد که پروتکل TLS را بگونه‌ای پیکربندی کرد که امنیت پس کوانتومی محدودی ارائه دهد. پیشنهاد می‌شود که برای چنین نیازی، سازمان‌ها به متخصصانی که می‌توانند چنین خدماتی ارائه کنند، رجوع کنند.

٤- احراز هویت کلاینت‌ها با استفاده از گواهی‌نامه

پروتکل TLS از احراز هویت دوطرفه با استفاده از گواهی‌نامه‌ها پشتیبانی می‌کند؛ به‌طوری که کلاینت با ارائه گواهی‌نامه خود، به سرور احراز هویت می‌شود و سرور نیز با گواهی‌نامه خود به کلاینت معرفی می‌شود. گواهی‌نامه‌های کلاینت معمولاً شامل اطلاعات حساس مانند نام کاربر هستند. پیش از نسخه ۱.۳، این گواهی‌نامه‌ها به عنوان بخشی از عملیات دست تکانی به صورت رمزنگاری نشده ارسال می‌شدند. بنابراین، اگر از گواهی‌نامه‌هایی با اطلاعات حساس برای احراز هویت استفاده می‌کنید و نیاز به حفظ محرمانگی دارید، توصیه می‌شود حتماً از نسخه ۱.۳ پروتکل TLS استفاده کنید.

٥- سنجاق کردن گواهی‌نامه و استفاده از DANE

زمانی که کلاینت یک جلسه TLS را با سرور آغاز می‌کند، گواهی‌نامه X.٥٠٩ سرور را بررسی می‌کند. کلاینت زنجیره‌ای از امضاهای دیجیتال را که گواهی‌نامه سرور را به مرجع صدور اصلی (CA) مرتبط می‌سازد، اعتبارسنجی می‌کند. سیستم PKI به دلیل اعتماد گسترده نرم‌افزارها به صدها مرجع صدور گواهی، آسیب‌پذیر است؛ زیرا اگر یکی از این مراجع گواهی‌نامه جعلی صادر کند، امنیت کل سیستم به خطر می‌افتد. در صورتی که کنترل کامل بر نرم‌افزار کلاینت و سرور داشته باشید، با استفاده از روش سنجاق کردن گواهی‌نامه[۲] می‌توانید اعتماد کلاینت را تنها به گواهی‌نامه‌های مشخص و مورد تایید محدود کنید.

---

[۱] Content Security Policy

[۲] Certifcate Pinning

کلاینت دیگر نیازی به بررسی زنجیره امضاهای دیجیتال ندارد، زیرا یا گواهی‌نامه را می‌شناسد و به آن اعتماد می‌کند یا خیر. در این حالت، اگر یک مرجع صدور گواهی‌نامه افشا شود، دیگر به عنوان ریسکی برای اتصال کلاینت مطرح نخواهد بود. به عنوان جایگزین، می‌توان با استفاده از سنجاق کردن گواهی‌نامه، تنها گواهی‌نامه‌هایی که توسط یک مرجع صدور خاص یا مراجع واسطه مشخص صادر شده‌اند را در لیست سفید قرار داد. به این ترتیب، افشای سایر مراجع صدور گواهی‌نامه دیگر تهدیدی برای امنیت نخواهد بود.

اتصال بین یک برنامه روی پلتفرم موبایل و سرور نمونه‌ای است که در آن سنجاق کردن گواهی‌نامه می‌تواند بسیار مؤثر باشد. با این حال، هنگام استفاده از این روش باید تاخیر انتشار تغییرات گواهی‌نامه را در نظر گرفت؛ زیرا در صورت تغییر سریع گواهی‌نامه و به‌روزرسانی نشدن کلاینت‌ها، آن‌ها قادر به اتصال به سرور نخواهند بود.

تکنیک احراز هویت موجودیت‌های نام‌دار مبتنی بر DNS (DANE[۱]) روشی است که به کلاینت‌ها امکان می‌دهد تا برای احراز هویت با استفاده از گواهی‌نامه‌ها، از سرویس DNS بهره‌مند شوند. در این روش، مدیر گواهی‌نامه اطلاعات مربوط به گواهی‌نامه را در قالب یک رکورد مخصوص DNS به نام TLSA منتشر می‌کند. این کار به کلاینت‌ها اجازه می‌دهد تا علاوه بر استفاده از سیستم PKI، اصالت گواهی‌نامه را با بررسی رکورد TLSA نیز تأیید کنند.

توجه داشته باشید که سیستم‌های سنتی DNS به تنهایی برای استفاده از DANE قابل اعتماد نیستند و به همین دلیل استفاده از DNSSEC[۲] ضروری است. یکی از کاربردهای رایج DANE، احراز هویت اتصالات TLS بین سرورهای ایمیل می‌باشد.

DNSSEC به مجموعه‌ای از فناوری‌ها اشاره دارد که تضمین یکپارچگی داده‌های DNS را فراهم می‌کنند. امروزه مهاجمان به‌راحتی می‌توانند درخواست‌های DNS را رهگیری کرده و پاسخ‌های جعلی ارسال کنند. با استفاده از DNSSEC، تمام درخواست‌ها تا سطح DNS root به صورت رمزنگاری‌شده و قابل تأیید پیگیری می‌شوند.

۶- استفاده از DNS CAA

---

[۱] DNS-based Authentication of Named Entities

[۲] Domain Name System Security Extensions

رکوردهای CAA[۱] در DNS به صاحبان دامنه این امکان را می‌دهند که با تعیین محدودیت‌هایی، فقط گروه خاصی از مراجع صدور گواهی‌نامه، اجازه صدور گواهی برای دامنه‌های خود را داشته باشند. این محدودیت باعث می‌شود که مراجع صدور گواهی‌نامه مخرب نتوانند به‌صورت غیرمجاز گواهی‌نامه‌ای برای دامنه صادر کنند و در نتیجه خطر صدور گواهی‌های جعلی—که می‌توانند برای جعل هویت یا حملات مرد میانی استفاده شوند—کاهش یابد.

۷- پیاده‌سازی HSTS

قابلیت HSTS[۲] با افزودن یک هدر پاسخ به وب‌سایت فعال می‌شود و پس از آن، تمامی مرورگرهای مدرن که از HSTS پشتیبانی می‌کنند، این قابلیت را به صورت اجباری اجرا می‌کنند. این ویژگی از برقراری هرگونه اتصال ناامن بین کاربر و وب‌سایت جلوگیری می‌کند. HSTS با الزام استفاده از HTTPS به جای HTTP، جلوی برخی حملات از جمله حملات مرد میانی را می‌گیرد. همچنین، در صورت وجود گواهی‌نامه‌های نامعتبر، سایت به‌طور کلی بارگذاری نمی‌شود؛ در حالی که در حالت عادی مرورگرها هشدار خطا نمایش می‌دهند که اغلب کاربران آن را نادیده گرفته و وارد سایت می‌شوند. برای افزایش امنیت، توصیه می‌شود از قابلیت HSTS Preloading استفاده شود که حتی اولین اتصال کاربر به وب‌سایت را نیز ایمن می‌کند.

۱- راهنمای سناریو محور

سند سازمان NCSC راهنمایی مبتنی بر سناریو برای پیکربندی سرورها و کلاینت‌ها ارائه کرده است. هدف این سناریوها، ارائه دید کلی و نقشه راهی جامع برای نحوه پیکربندی سرورها و کلاینت‌هایی است که از پروتکل TLS پشتیبانی می‌کنند. برای اطلاعات بیشتر درباره سطوح امنیتی مختلف هر سناریو، به جداول مربوطه در بخش‌های پیشین مراجعه کنید.

۸- سناریو نخست: کنترل برروی سرور و کلاینت

در برخی از موارد، مدیری که برروی پیکربندی سرور کنترل دارد، می‌تواند برروی پیکربندی کلاینت نیز کنترل داشته باشد. برای مثال، یک وب‌سرور که برنامه وب داخلی را فقط در دسترس گروه محدودی از مشتریان سازمان قرار می‌دهد. سازمان NCSC در چنین شرایطی توصیه می‌کند از تنظیماتی با سطح امنیتی

---

[۱] Certification Authority Authorization

[۲] HTTP Strict Transport Security

«خوب» استفاده شود. در ادامه، راهنمای گام‌به‌گام پیشنهادی این سازمان برای پیکربندی در چنین موقعیتی آورده شده است:

۱. تهیه فهرستی از گزینه‌های در دسترس TLS که برروی سرور قابل فعال سازی می‌باشند.

۲. تهیه فهرستی از کلاینت‌های مختلفی که قرار است به سرور متصل شوند.

۳. برای هر نوع کلاینت، مجموعه تنظیمات پشتیبانی شده را فهرست کنید.

٤. برای موارد زیر از پیکربندی با درجه «خوب» استفاده کنید:

الف. نسخه TLS برای سرور: اطمینان حاصل کنید که هر مشتری از نسخه‌ای پشتیبانی می‌کند که توسط سرور نیز پشتیبانی می‌شود.

ب. الگوریتم‌های انتخابی برای سرور: اطمینان حاصل کنید که هر کلاینت از الگوریتمی پشتیبانی می‌کند که توسط سرور نیز پشتیبانی می‌شود.

ج. طول کلید برای سرور: اطمینان حاصل کنید که هر کلاینت از طول کلید انتخاب شده پشتیبانی می‌کند.

د. منحنی‌های بیضوی پشتیبانی شده برای سرور: اطمینان حاصل کنید که هر کلاینت از منحنی بیضوی‌ای که توسط سرور پشتیبانی می‌شود، پشتیبانی کند. اگر کلاینت‌های قدیمی‌تر قادر به پشتیبانی از منحنی‌های بیضوی نیستند، استفاده از گروه‌های میدان متناهی را مدنظر قرار دهید. در این صورت، مطمئن شوید که هر کلاینت از یک گروه میدان متناهی با سطح امنیتی «کافی» پشتیبانی می‌کند و این گروه با سرور نیز سازگار باشد.

ه. سایر گزینه‌ها برای سرور: این گزینه‌ها برای زمانی هستند که دلایل قوی برای انتخاب تنظیمات با درجه ایمنی «کافی» وجود داشته باشد. این دلایل از فهرست کلاینت‌های تنظیم شده و توانایی پشتیبانی آن‌ها از گزینه‌های گوناگون منشا می‌گیرد.

۵. آیا سرور از پشتیبانی از یک پیکربندی خاص با درجه «خوب» که کلاینت‌ها توانایی پشتیبانی از آن را دارند، برخوردار نیست؟ سه گزینه دارید:

۱- الف. کلاینت‌ها را با نوعی دیگر که از یک پیکربندی متفاوت با درجه ایمنی «خوب» پشتیبانی کنند، جایگزین نمایید.

ب. سرور را با نوعی دیگر که از یک پیکربندی متفاوت با درجه ایمنی «خوب» پشتیبانی کند، جایگزین نمایید.

ج. از یک پیکربندی با درجه ایمنی «مناسب» که توسط کلاینت و سرور پشتیبانی می‌شود، استفاده کنید.

۶ .آیا سرور از پشتیبانی یک پیکربندی با درجه ایمنی «خوب» و «کافی» که کلاینت‌ها از آن پشتیبانی می‌کنند، پشتیبانی نمی‌کند؟ سه گزینه دارید:

الف. کلاینت‌ها را با نوعی دیگر که از یک پیکربندی متفاوت با درجه ایمنی «خوب» و «کافی» پشتیبانی کنند، جایگزین نمایید.

ب. سرور را با نوعی دیگر که از یک پیکربندی متفاوت با درجه ایمنی «خوب» و «کافی» پشتیبانی کند، جایگزین نمایید.

ج. یک گزینه با درجه ایمنی «فاز خروج» که توسط سرور و کلاینت پشتیبانی می‌شود را انتخاب نمایید. این گزینه از روی ریسک‌های امنیتی و نیاز به تغییر در آینده باید کمتر ترجیح داده شود.

۷ .سرور را با تنظیمات انتخاب‌شده پیکربندی کنید. پیکربندی TLS بخشی از نرم‌افزاری است که از ارتباط TLS استفاده می‌کند. به عنوان مثال، اگر می‌خواهید HTTPS را به کلاینت‌ها ارائه دهید، TLS به عنوان بخشی از نرم‌افزار وب سرور پیکربندی می‌شود.

۸ . با آزمایش مطمئن شوید که این پیکربندی برای تمام کلاینت‌ها کار می‌کند، آیا به مشکل سازگاری در اتصالات کلاینت‌ها و سرور برخوردید؟ پس به قدم پنجم بازگردید.

۹. پیکربندی انتخاب‌شده را مستندسازی کنید و دست کم برروی موارد زیر زمان بگذارید:

الف. شرایط حذف برنامه‌ریزی‌شده هر گزینه «در فاز خروج» که انتخاب کرده‌اید را مشخص کنید.

ب. دلایلی که باعث شد تا از یک گزینه با درجه ایمنی «کافی» به جای درجه ایمنی «خوب» استفاده کنید، مستند سازی کنید.

۹- سناریو دوم: کنترل فقط برروی سرور

در مواردی دیگر، مدیر سیستم تنها کنترل سرور را در اختیار دارد و بر کلاینت‌ها—یا حداقل همه آن‌ها— کنترلی ندارد. به عنوان مثال، وب‌سروری که یک سایت عمومی را ارائه می‌دهد، در این دسته قرار می‌گیرد.

سازمان NCSC توصیه می‌کند از ترکیب پیکربندی‌هایی با سطح امنیتی «خوب» و «کافی» استفاده شود. در برخی پیاده‌سازی‌ها ممکن است لازم باشد تنظیماتی با درجه امنیتی «در فاز خروج» نیز به کار گرفته شود.

۱. فهرستی از نوع کلاینت‌هایی که نیاز به برقراری ارتباط با سرور دارند (یا باید داشته باشند) تهیه کنید. این نیاز معمولاً در مشورت با مالک کسب‌وکار تعیین می‌شود.

الف. نمایش تعادل ریسک و ارزش: ارزش سازگاری در مقابل ریسک و هزینه‌های پشتیبانی مرتبط با پیکربندی‌های آسیب‌پذیرتر را مشخص کنید. فهرست گزینه‌های TLS موجود بر روی سرور را تهیه کنید.

۲ .برای سرور TLS از نسخه‌هایی با درجه ایمنی «خوب» و «کافی» استفاده کنید.

۳ .الگوریتم‌هایی با درجه ایمنی «خوب» و «کافی» برای سرور انتخاب کنید. طیف گسترده‌ای از الگوریتم‌هایی با درجه «کافی» را پشتیبانی کنید تا با کلاینت‌های قدیمی‌تر سازگار باشید. در عین حال، بیش از حد نیاز از الگوریتم‌هایی با درجه «کافی» پشتیبانی نکنید تا ریسک و هزینه‌های پشتیبانی کاهش یابد.

۴ .طول کلیدهایی با درجه «کافی» را انتخاب کنید. در صورت اطمینان از پشتیبانی تمامی کلاینت‌ها، از طول کلیدهایی با درجه ایمنی «خوب» استفاده کنید.

۵ .منحنی‌های بیضوی با درجه «خوب» را برای سرور را انتخاب کنید. اگر گمان می‌کنید همه مشتریان از این منحنی‌ها پشتیبانی نمی‌کنند، منحنی‌ها با درجه «کافی» را نیز اضافه کنید. بیش از حد منحنی‌های لازم را پشتیبانی نکنید.

۶ .برخی مشتریان قدیمی از منحنی‌های بیضوی پشتیبانی نمی‌کنند. اگر نیاز دارید از این مشتریان پشتیبانی کنید، میدان‌های متناهی مناسب را انتخاب کنید. بیش از حد لازم از این میدان‌ها پشتیبانی نکنید.

۷ .سرور را با تنظیمات انتخاب شده پیکربندی کنید. پیکربندی TLS بخشی از نرم‌افزاری است که از اتصال TLSاستفاده می‌کند، برای نمونه در صورت ارائه HTTPS، TLS در نرم‌افزار سرور وب پیکربندی می‌شود.

۸ .درباره نرم‌افزارهای مورد استفاده توسط کلاینت‌ها تفکر کنید. بررسی کنید که آیا مشتریانی که از این نرم‌افزارها استفاده می‌کنند، می‌توانند اتصال برقرار کنند یا خیر.

۹ .آیا مشکلات سازگاری دارید؟ گزینه‌هایی که باعث این مشکلات می‌شوند را شناسایی کنید.

الف. معمولاً این مشکلات با جایگزینی یا افزودن تنظیمات «خوب» به تنظیمات «کافی» حل می‌شوند.

ب. اگر در فهرست ساخته شده در قدم‌های ۱ و ۹ کلاینت‌هایی با نرم‌افزارهای بسیار قدیمی وجود دارند، ممکن است نیاز باشد تا گزینه‌های «در فاز خروج» را نیز بطور موقتی در پیکربندی خود اضافه کنید. از آنجایی که گزینه‌های «در فاز خروج» آسیب‌پذیر بوده و امنیت بسیار اندکی ارائه می‌کنند، از پشتیبانی از این گزینه‌ها بیش از حد نیاز خودداری کنید.

۱۱ .معیارهای واضح برای حذف برنامه‌ریزی شده تنظیماتی با درجه «در فاز خروج» را تعیین کرده و آن‌ها را مستند سازی کنید.

۲- مجموعه رمزگذاری

اشاره شد که الگوریتم رمزنگاری داده‌ها و هشینگ در نسخه ۱.۳ به عنوان «مجموعه رمزگذاری» شناخته و جهت حفاظت از رکوردها بکار گرفته می‌شوند. در نسخه‌های پیشین تا نسخه ۱.۲، این عبارت به الگوریتم

تبادل کلید و الگوریتم امضای دیجیتال اشاره می‌کرد. در اینجا مجموعه رمزهای مختلف با درجه امنیت آن‌ها آورده شده است:

۱۰- درجه خوب

TLS_AES_۲۵۶_GCM_SHA۳۸٤

TLS_CHACHA۲۰_POLY۱۳۰۵_SHA۲۵٦

TLS_AES_۱۲۸_GCM_SHA۲۵٦

۱۱- درجه کافی

TLS_ECDHE_ECDSA_WITH_AES_۲۵٦_GCM_SHA۳۸٤

TLS_ECDHE_ECDSA_WITH_CHACHA۲۰_POLY۱۳۰۵_SHA۲۵٦

TLS_ECDHE_ECDSA_WITH_AES_۱۲۸_GCM_SHA۲۵٦

TLS_ECDHE_RSA_WITH_AES_۲۵٦_GCM_SHA۳۸٤

TLS_ECDHE_RSA_WITH_CHACHA۲۰_POLY۱۳۰۵_SHA۲۵٦

TLS_ECDHE_RSA_WITH_AES_۱۲۸_GCM_SHA۲۵٦

TLS_ECDHE_ECDSA_WITH_AES_۲۵٦_CBC_SHA۳۸٤

TLS_ECDHE_ECDSA_WITH_AES_۲۵٦_CBC_SHA

TLS_ECDHE_ECDSA_WITH_AES_۱۲۸_CBC_SHA۲۵٦

TLS_ECDHE_ECDSA_WITH_AES_۱۲۸_CBC_SHA

TLS_ECDHE_RSA_WITH_AES_۲۵٦_CBC_SHA۳۸٤

TLS_ECDHE_RSA_WITH_AES_۲۵٦_CBC_SHA

TLS_ECDHE_RSA_WITH_AES_۱۲۸_CBC_SHA۲۵٦

TLS_ECDHE_RSA_WITH_AES_۱۲۸_CBC_SHA

TLS_DHE_RSA_WITH_AES_۲۵٦_GCM_SHA۳۸٤

TLS_DHE_RSA_WITH_CHACHA۲۰_POLY۱۳۰۵_SHA۲۵٦

TLS_DHE_RSA_WITH_AES_۱۲۸_GCM_SHA۲۵٦

TLS_DHE_RSA_WITH_AES_۲۵٦_CBC_SHA۲۵٦

TLS_DHE_RSA_WITH_AES_۲۵٦_CBC_SHA

TLS_DHE_RSA_WITH_AES_۱۲۸_CBC_SHA۲۵٦

TLS_DHE_RSA_WITH_AES_۱۲۸_CBC_SHA

۱۲- درجه در فاز خروج

TLS_ECDHE_ECDSA_WITH_۳DES_EDE_CBC_SHA

TLS_ECDHE_RSA_WITH_۳DES_EDE_CBC_SHA

TLS_DHE_RSA_WITH_۳DES_EDE_CBC_SHA

TLS_RSA_WITH_AES_۲۵۶_GCM_SHA۳۸۴

TLS_RSA_WITH_AES_۱۲۸_GCM_SHA۲۵۶

TLS_RSA_WITH_AES_۲۵۶_CBC_SHA۲۵۶

TLS_RSA_WITH_AES_۲۵۶_CBC_SHA

TLS_RSA_WITH_AES_۱۲۸_CBC_SHA۲۵۶

TLS_RSA_WITH_AES_۱۲۸_CBC_SHA

TLS_RSA_WITH_۳DES_EDE_CBC_SHA

انتخاب نسخه پروتکل	فقط از TLS ۱٫۳ یا TLS ۱٫۲ استفاده شود.
	TLS ۱٫۰ و ۱.۱ در حالت فاز خروج (Phase out) هستند؛ فقط جهت پشتیبانی از سیستم‌های قدیمی در شرایط خاص استفاده شوند.
	تمام نسخه‌های SSL در وضعیت ناکافی قرار دارند و باید کاملاً غیرفعال شوند.
الگوریتم‌های تبادل کلید (Key Exchange)	استفاده از ECDHE توصیه شده با پشتیبانی از رازداری به جلو
	در صورت محدودیت استفاده از ECDHE استفاده از DHE
	عدم استفاده از الگوریتم‌های PSK، KRB۵، SRP
الگوریتم‌های هش	استفاده فقط از SHA-۲۵۶، SHA-۳۸۴، SHA-۵۱۲
	عدم استفاده از الگوریتم‌های SHA-۱، MD۵
	پشتیبانی از SHA۲ الزامی است تا نشست TLS سطح امنیتی «خوب» داشته باشد.
الگوریتم‌های رمزنگاری متقارن	استفاده از الگوریتم‌های AEAD مانند AES-۲۵۶-GCM, AES-۱۲۸-GCM, ChaCha۲۰-Poly۱۳۰۵
	عدم استفاده از ۳DES, RC۴, NULL, DES, IDEA

طول کلید RSA	۳۰۷۲ بیت یا بیشتر
منحنی‌های بیضوی	استفاده از secp۲۵۶r۱, secp۳۸٤r۱, curve۲۵۵۱۹, curve٤٤۸
گروه‌های میدان متناهی – (Finite Field Groups برای DHE)	استفاده از ffdhe٤۰۹٦، ffdhe۳۰۷۲
	در صورت امکان، از ECDHE به جای DHE استفاده شود.
انتخاب کتابخانه رمزنگاری (TLS Libraries)	استفاده از OpenSSL, SChannel, NSS, mbedTLS
فشرده‌سازی	غیرفعال کردن فشرده‌سازی TLS
مذاکره مجدد (Renegotiation)	غیرفعال‌سازی کامل مذاکره مجدد ناامن در TLS ۱،۳
	غیرفعال کردن ویژگی مذاکره مجدد از سمت کلاینت
۰-RTT در TLS ۱،۳	غیرفعال کردن ۰-RTT در صورت عدم نیاز
OCSP Stapling	فعال‌سازی OCSP Stapling برای حفظ حریم خصوصی کاربران و افزایش سرعت پاسخ TLS
تیکت‌های نشست	در TLS ۱،۰/۱،۱/۱،۲ استفاده از Session Ticket با احتیاط و همراه با اقدامات امنیتی انجام شود
	در صورت استفاده از Session Ticket در نسخه‌های قدیمی TLS از ذخیره "کلید رمزنگاری تیکت نشست" روی دیسک خودداری شود
	ترجیح به استفاده از TLS ۱،۳ برای پیاده‌سازی امن فرآیند ازسرگیری نشست‌ها
مولد اعداد تصادفی و آنتروپی	اطمینان حاصل کنید که سیستم به یک منبع قوی آنتروپی مجهز است
	از مولدهای شبه‌تصادفی (PRNG) امن و با توزیع یکنواخت استفاده شود
	در صورت امکان، از قابلیت تولید تصادفی سخت‌افزاری موجود در پردازنده‌های مدرن استفاده

	کنید
مدیریت گواهی‌نامه	از یک مرجع صدور گواهی‌نامه معتبر و قابل اعتماد استفاده کنید.
	اطمینان حاصل کنید که گواهی‌نامه تمام دامنه‌ها و زیردامنه‌های مورد استفاده را پوشش می‌دهد.
	حالت‌های با www و بدون www را نیز در گواهی‌نامه لحاظ کنید.
	تاریخ انقضای گواهی‌نامه‌ها را پیگیری و ثبت کنید.
	پیش از خرید گواهی‌نامه EV بررسی کنید که آیا واقعاً ارزش افزوده امنیتی برای سازمان شما ایجاد می‌کند یا خیر. در نظر داشته باشید که مرورگرها دیگر نشان‌های EV را برجسته نمایش نمی‌دهند.
نقطه پایانی اتصال TLS	در صورت نیاز به حفظ محرمانگی و یکپارچگی داخل شبکه، یک نشست TLS جدید در لایه داخلی راه‌اندازی کنید.
	اطمینان حاصل کنید که تمام منابع وب‌سایت JS، CSS، تصاویر و .. نیز از طریق HTTPS بارگذاری می‌شوند.
	در استفاده از منابع خارجی از سیاست امنیت محتوا (CSP) برای کنترل دقیق منابع استفاده کنید.
احراز هویت کلاینت با گواهی‌نامه	در صورت استفاده از گواهی‌نامه‌های حساس کلاینت، از TLS ۱،۳ استفاده کنید.
	در نسخه‌های پیش از TLS ۱،۳، گواهی کلاینت بدون رمزنگاری منتقل می‌شود و ممکن است افشا شود.
	در صورت کنترل کامل بر کلاینت و سرور، از روش سنجاق کردن گواهی‌نامه (Certificate Pinning)

سنجاق کردن گواهی‌نامه و استفاده از DANE	استفاده کنید
	در Pinning، تأخیر در به‌روزرسانی کلاینت‌ها در صورت تغییر گواهی‌نامه را در نظر بگیرید.
	برای افزایش امنیت، فقط گواهی‌نامه‌هایی از مراجع خاص را مجاز کنید (لیست سفید).
	برای احراز اصالت گواهی از طریق DNS، از DANE استفاده کنید و رکورد TLSA را پیکربندی نمایید. توجه داشته باشید استفاده از DANE فقط با DNSSEC معتبر است؛ بدون آن قابل اطمینان نیست.
	برای ایمن‌سازی رکوردهای DNS، از DNSSEC جهت تضمین یکپارچگی و جلوگیری از جعل استفاده کنید.
استفاده از DNS CAA	در DNS رکورد CAA تنظیم کنید تا فقط مراجع صدور خاص بتوانند برای دامنه شما گواهی صادر کنند.
پیاده‌سازی HSTS	با استفاده از هدر HSTS، مرورگر را وادار به استفاده از HTTPS کنید.
	برای امنیت بیشتر، از قابلیت HSTS Preloading استفاده کنید

۶ پیکربندی Nginx

فایل پیش‌فرض پیکربندی وب‌سرور معمولاً در مسیر:
/etc/nginx/conf.d/default.conf

و فایل پیش‌فرض پیکربندی پراکسی در مسیر:
/etc/nginx/nginx.conf

کد پیکربندی NGINX برای ریدایرکت کردن HTTP به HTTPS

```
{server
    listen ۸۰;
    server_name cisecurity.org;
    return ۳۰۱ https://$host$request_uri;}
```

- **گواهی‌ها و زنجیره‌های اعتماد (Certificates and Trust Chains)**

باز کردن فایل گواهی SSL:

cat /etc/nginx/cert.pem

اجرای دستور زیر تا کلید خصوصی ۳۰۷۲ بیتی و فایل درخواست گواهی ایجاد شود:

openssl req -new -newkey rsa:۳۰۷۲ -keyout nginx.key -out nginx.csr

- **محافظت کلید خصوصی سرور از دسترسی غیرمجاز**

اجرای دستور زیر برای بررسی سطح دسترسی:

find /etc/nginx/ -name '*.key' -exec stat -Lc "%n %a" {} +

خروجی مورد انتظار:

/etc/nginx/nginx.key ٤٠٠

اگر عددی بیشتر از ۴۰۰ باشد (مثل ۶۴۴ یا ۶۰۰)، یعنی سطح دسترسی بیش از حد مجاز است.

برای حذف سطح دسترسی اضافی از فایل‌های کلید خصوصی در مسیر /etc/nginx/، دستور زیر را اجرا کنید:

find /etc/nginx/ -name '*.key' -exec chmod u-wx,go-rwx {} +

- **استفاده از نسخه TLS:**

برای فعال کردن فقط TLS ۱،۲ وTLS ۱،۳ ، پیکربندی NGINX باید شامل خط زیر باشد:

ssl_protocols TLSv۱،۲ TLSv۱،۳;

این خط را در بلاک server یا در فایل تنظیمات کلی /etc/nginx/nginx.conf یا فایل‌های include شده قرار دهید.

- **پیکربندی الگوریتم‌های رمزنگاری:**

برای بررسی پیکربندی cipher در NGINX، از دستورات زیر استفاده کنید:

```
grep -ir ssl_ciphers /etc/nginx/
grep -ir proxy_ssl_ciphers /etc/nginx/
```

مثال پیکربندی در بلاک سرور برای ارتباط با کلاینت:

```
server {
   ssl_ciphers ALL:!EXP:!NULL:!ADH:!LOW:!SSLv۲:!SSLv۳:!MD۵:!RC٤;
}
```

- **الگوریتم‌های تبادل کلید:**

برای بررسی اینکه آیا پارامتر سفارشی DH تنظیم شده یا نه، از دستور زیر استفاده کنید:

```
grep ssl_dhparam /etc/nginx/nginx.conf
```

اگر چیزی نمایش داده نشد، یعنی این تنظیم انجام نشده است.

تولید پارامترهای قویDiffie-Hellman

```
mkdir /etc/nginx/ssl
openssl dhparam -out /etc/nginx/ssl/dhparam.pem ۲۰٤۸
chmod ٤۰۰ /etc/nginx/ssl/dhparam.pem
```

این فایل DH با طول ۲۰۴۸ بیت تولید شده و فقط توسط کاربر root قابل خواندن است.

برای اعمال در سرور در بلاک server داخل بلاک http، خط زیر را اضافه کنید:

```
http {
   server {
      ssl_dhparam /etc/nginx/ssl/dhparam.pem;
   }
}
```

- **فعال کردن OCSP stapling**

برای بررسی فعال بودن OCSP stapling، دستور زیر را اجرا کنید:

```
grep -ir ssl_stapling /etc/nginx
```

اگر در فایل پیکربندی شما دو خط زیر وجود نداشته باشند، یعنی این قابلیت فعال نشده:

```
ssl_stapling on;
```

```
ssl_stapling_verify on;
```

آدرس OCSP سرور CA خود را از وب‌سایت شرکت صادرکننده گواهی پیدا کنید و با دستور زیر بررسی کنید که سرور شما به آن دسترسی دارد:

curl -I "https://ocsp.example-ca.com"

اگر پاسخ HTTP/۱،۱ ۲۰۰ OK دریافت کردید، یعنی دسترسی برقرار است.

در بلاک serverمربوط به پیکربندیHTTPS ، این دو خط را اضافه کنید:

```
server {
   ssl_stapling on;
   ssl_stapling_verify on;
}
```

- **هدر HSTS**

برای بررسی فعال بودنHSTS ، دستور زیر را اجرا کنید:

grep -ir Strict-Transport-Security /etc/nginx

اگر در خروجی، خط زیر یا مشابه آن دیده نشود، یعنی این قابلیت پیاده‌سازی نشده است:

add_header Strict-Transport-Security "max-age=۱۵۷۶۸۰۰۰;" always;

توجه : مقدار ۱۵۷۶۸۰۰۰معادل ۶ ماه است.

اطمینان حاصل کنید که در فایل پیکربندی سرور (وب‌سرور یا پراکسی)، هدر HSTS با مدت اعتبار مناسب تعریف شده باشد. مثالی از این پیکربندی:

```
server {
   add_header Strict-Transport-Security "max-age=۱۵۷۶۸۰۰۰;" always;
}
```

اگر می‌خواهید پشتیبانی از ساب‌دامنه‌ها نیز شامل شود و امکان افزودن به لیست preload مرورگرها را داشته باشید، می‌توانید مقدار هدر را به صورت زیر گسترش دهید:

add_header Strict-Transport-Security "max-age=۱۵۷۶۸۰۰۰; includeSubDomains; preload" always;

فقط در صورتی preload را فعال کنید که کاملاً مطمئن هستید تمام بخش‌های دامنه‌تان از HTTPS پشتیبانی می‌کنند.

- **احراز هویت گواهی کلاینت به سرور بالادستی**

برای بررسی اینکه این قابلیت پیکربندی شده یا نه، از دستور زیر استفاده کنید:

```
grep -ir proxy_ssl_certificate /etc/nginx
```

اگر پیکربندی به‌درستی انجام شده باشد، باید خروجی‌ای مشابه موارد زیر داشته باشید:

```
proxy_ssl_certificate /etc/nginx/ssl/nginx.pem;
proxy_ssl_certificate_key /etc/nginx/ssl/nginx.key;
```

همچنین باید بررسی کنید که این دستورها در بلاک locationکه ترافیک به سمت upstream هدایت می‌شود قرار گرفته باشند.

فرض کنید گواهی و کلید خصوصی را در مسیر زیر قرار می‌دهید:

```
/etc/nginx/ssl/nginx.pem
/etc/nginx/ssl/nginx.key
```

افزودن پیکربندی به بلاک location مربوط بهupstream

```
location /upstream-path/ {
    proxy_pass https://your-upstream-server;
    proxy_ssl_certificate /etc/nginx/ssl/nginx.pem;
    proxy_ssl_certificate_key /etc/nginx/ssl/nginx.key;
}
```

سرور بالادستی شما (upstream) نیز باید برای بررسی گواهی کلاینت پیکربندی شود.

اگر سرور upstream شما نیز NGINX است، باید در بلاک server پیکربندی زیر را اضافه کنید:

```
ssl_client_certificate /etc/nginx/ssl/ca.cert;
ssl_verify_client on;
```

ssl_client_certificate مسیر گواهی CA است که گواهی کلاینت باید توسط آن امضا شده باشد.

ssl_verify_client on سرور را ملزم می‌کند که حتماً کلاینت گواهی معتبر ارائه دهد.

- **اعتبارسنجی هویت سرور بالادستی (upstream)**

در بلاک location که درخواست‌ها را به سرور upstream ارسال می‌کند، بررسی کنید که دو دستور زیر وجود داشته باشند:

```
proxy_ssl_trusted_certificate /etc/nginx/trusted_ca_cert.crt;
proxy_ssl_verify on;
```

برای بررسی با خط فرمان، از دستورات زیر استفاده کنید:

```
grep -ir proxy_ssl_trusted_certificate /etc/nginx
grep -ir proxy_ssl_verify /etc/nginx
```

اطمینان حاصل کنید که گواهی‌های موجود در فایل اشاره‌شده در دستور proxy_ssl_trusted_certificate معتبر هستند و زنجیره اعتماد (Trust Chain) کامل است.

زنجیره گواهی سرور upstream را به‌صورت کامل در قالب pem. تهیه کنید. این فایل معمولاً شامل گواهی root و intermediate از صادرکننده گواهی (CA) است.

سپس، این فایل را در یک مسیر مناسب مثلاً /etc/nginx/trusted_ca_cert.crt قرار دهید.

در بلاک location مربوط به proxy ، دستورات زیر را اضافه کنید:

```
location / {
    proxy_pass https://upstream-server.example.com;
    proxy_ssl_trusted_certificate /etc/nginx/trusted_ca_cert.crt;
    proxy_ssl_verify on;
}
```

- **قابلیت Preloading**

برای بررسی اینکه دامنه شما preloaded شده یا نه، به سایت زیر بروید:

https://hstspreload.org

نام دامنه اصلی را وارد کنید و نتیجه را بررسی کنید.

پیکربندی هدر HSTS در دامنه اصلی به صورت زیر:

add_header Strict-Transport-Security "max-age=۳۱۵۳٦۰۰۰; includeSubDomains; preload" always;

max-age=۳۱۵۳٦۰۰۰ معادل ۱ سال

includeSubDomains تمام زیردامنه‌ها را نیز شامل می‌شود

preload اعلام می‌کند که دامنه آماده اضافه‌شدن به لیست preload مرورگرهاست

- **غیرفعال‌سازی session resumption**

برای بررسی اینکه آیا ssl_session_tickets غیرفعال شده یا نه، دستور زیر را اجرا کنید:

```
grep -ir ssl_session_tickets /etc/nginx
```

خروجی صحیح باید شامل این خط باشد:

```
ssl_session_tickets off;
```

توجه: فایل‌های include شده در تنظیمات NGINX خارج از مسیر /etc/nginx را نیز بررسی کنید تا مطمئن شوید همه بلاک‌های server بررسی شده‌اند.

برای غیرفعال‌سازی session resumption، در هر بلاک server در پیکربندی NGINX، خط زیر را اضافه کنید:

```
ssl_session_tickets off;
```

- **استفاده از رمزنگاری با Perfect Forward Secrecy (PFS)**

برای بررسی اینکه فقط الگوریتم‌هایی استفاده شده‌اند که از PFS پشتیبانی می‌کنند، دستورهای زیر را اجرا کنید:

```
grep -ir ssl_ciphers /etc/nginx/
grep -ir proxy_ssl_ciphers /etc/nginx/
```

در خروجی، باید فقط رمزنگارهایی از نوع زیر مشاهده شود:

- ECDHE یا EECDH
- DHE یا EDH

این‌ها الگوریتم‌هایی هستند که از PFS پشتیبانی می‌کنند.

اطمینان حاصل کنید که فقط رمزنگارهایی که با PFS سازگار هستند، در پیکربندی استفاده شده‌اند.

- **غیرفعال کردن ۰-RTT**

اگر از nginx-quic یا پشتیبانی از TLS ۱،۳ استفاده می‌کنید در این حالت، برای غیرفعال کردن ۰-RTT باید این گزینه را در فایل پیکربندی Nginx اضافه کنید:

```
ssl_early_data off;
```

- **اضافه کردن هدر CSP:**

برای این کار دستور زیر به بلوک سرور اضافه کنید:

```
add_header Content-Security-Policy “script-src ‘self’; object-src ‘none’; base-uri ‘self’; frame-ancestors ‘none’;”;
```

۷ پیکربندی Apache

ماژول mod_ssl پرکاربردترین ماژول استاندارد برای پیاده‌سازی SSL/TLS در Apache است.

مراحل زیر را برای اطمینان از پیاده‌سازی وضعیت پیشنهادی انجام دهید:
بررسی کنید که mod_ssl و /یا mod_nss در پیکربندی Apache بارگذاری شده‌اند یا خیر:

```
# httpd -M | egrep 'ssl_module|nss_module'
```

نتیجه باید شامل "Syntax OK" همراه با یکی یا هر دوی این ماژول‌ها باشد.

از گزینه --with-ssl= برای مشخص کردن مسیر openssl و گزینه --enable-ssl برای افزودن ماژول‌های SSL به ساخت استفاده کنید. ممکن است استفاده از گزینه --with-included-apr نیز لازم باشد.

```
# ./configure --with-included-apr --with-ssl=$OPENSSL_DIR --enable-ssl
```

کافی است مطمئن شوید بسته mod_ssl نصب شده است. بسته mod_nss نیز ممکن است نصب شود.
برای Red Hat می‌توانید از دستور زیر استفاده کنید:

```
# yum install mod_ssl
```

- **اطمینان از نصب گواهی معتبر و مورد اعتماد**

بررسی وضعیت با استفاده از بسته گواهی‌های CA معتبر:

```
$ openssl verify -CAfile /etc/ssl/certs/ca-bundle.crt -purpose sslserver /etc/ssl/certs/example.com.crt
```

نام میزبان باید با URL در مرورگر مطابقت داشته باشد، مثل www.example.com.

استفاده از طول کلید ۳۰۷۲ بیتی (با گذرواژه):

```
# cd /etc/pki/tls/certs
```

```
# umask ۰۷۷
# openssl genrsa -aes۱۲۸ ۳۰۷۲ > example.com.key
```

فایل تنظیمات CSR باید شامل فیلد subjectAltName برای چند hostname باشد.

cp /etc/ssl/openssl.cnf ex۱.cnf

تنظیم فیلد subjectAltName

```
[ req ]
distinguished_name = req_distinguished_name
req_extensions = req_ext
[ req_ext ]
subjectAltName = @alt_names
[ alt_names ]
DNS.۱ = www.example.com
DNS.۲ = example.com
DNS.۳ = app.example.com
DNS.٤ = service.example.com
```

تنظیم بخش اطلاعات گواهی

```
[ req_distinguished_name ]
countryName_default = GB
stateOrProvinceName_default = Scotland
localityName_default = Glasgow
۰.organizationName_default = Example Company Ltd
organizationalUnitName_default = ICT
commonName_default = www.example.com
```

تولید درخواست امضای گواهی(CSR)

openssl req -new -config ex۱.cnf -out example.com.csr -key example.com.key

بررسی صحت CSR

openssl req -in example.com.csr -text | more

انتقال کلید خصوصی به مسیر نهایی:

mv www.example.com.key /etc/ssl/private/

CA گواهی امضا شده را برمی‌گرداند. گواهی نهایی را در مسیر مناسب قرار دهید.

/etc/ssl/certs/www.example.com.crt

ویرایش فایل پیکربندی Apache

```
SSLCertificateFile /etc/ssl/certs/example.com.crt
SSLCertificateKeyFile /etc/ssl/private/example.com.key
SSLCertificateChainFile /etc/ssl/certs/server-chain.crt
```

- **اطمینان از محافظت کلید خصوصی سرور**

برای هر فایل گواهی که در فایل‌های پیکربندی Apache با دستور SSLCertificateFile مشخص شده، بررسی کنید که آیا داخل فایل عبارت -----PRIVATE KEY وجود دارد (نشان‌دهنده وجود کلید خصوصی در آن فایل است).

برای هر مسیر مشخص‌شده با SSLCertificateKeyFile بررسی کنید که:

- مالک فایل باید root:root باشد.
- سطح دسترسی فایل باید ۰۴۰۰باشد فقط root اجازه خواندن دارد

کلیدهای خصوصی باید جدا از گواهی‌های عمومی ذخیره شوند.
مسیر تمام دستورات SSLCertificateFile را در فایل‌های پیکربندی Apache پیدا کنید.
اگر در کنار آن‌ها دستور SSLCertificateKeyFile جداگانه‌ای وجود ندارد، کلید را از گواهی جدا کرده و در فایل مجزا ذخیره کنید. سپس دستور SSLCertificateKeyFile را برای اشاره به این فایل کلید اضافه کنید.

برای تمام مسیرهای مشخص‌شده در SSLCertificateKeyFile

- مالکیت فایل را به root:root تغییر دهید.
- سطح دسترسی را روی ۰۴۰۰ تنظیم کنید:

```
chown root:root /path/to/private.key
chmod ۰٤۰۰ /path/to/private.key
```

- **اطمینان از غیرفعال بودن پروتکل‌های ضعیف SSL**

برای بررسی اینکه وضعیت توصیه‌شده پیاده‌سازی شده است:

۱. در فایل‌های پیکربندی Apache بررسی کنید که دستور SSLProtocol در سطح سرور و تمام VirtualHostهایی که SSL فعال دارند، وجود داشته باشد.

۲. برای هر یک از این دستورات، بررسی کنید که یکی از موارد زیر رعایت شده باشد:

  - علامت -SSLv۲ و -SSLv۳ صراحتاً درج شده باشد.

SSLProtocol all -SSLv۲ -SSLv۳

یا فقط پروتکل‌های TLS به‌صورت صریح و بدون هیچ علامتی (مثل + یا -) لیست شده باشند:

SSLProtocol TLSv۱،۳

- **اطمینان از غیرفعال بودن رمزنگاری‌های ضعیف SSL/TLS**

برای بررسی وضعیت پیکربندی:

۱. از ابزار sslscan موجود در Kali Linux یا GitHub: https://github.com/rbsec/sslscan استفاده کنید. رنگ‌ها به صورت زیر نشانه‌گذاری می‌شوند:

  - قرمز: الگوریتم NULL (بدون رمزنگاری)، پروتکل‌های شکسته مثل SSLv۲/SSLv۳، امضای ضعیف مثل MD۵
  - زرد: الگوریتم‌های ضعیف مثل RC٤ یا امضاهای SHA-۱
  - بنفش: الگوریتم‌های ناشناس مثل ADH یا AECDH

۲. همچنین می‌توانید از ابزار آنلاین SSL Labs استفاده کنید:
https://www.ssllabs.com/

۳. در فایل پیکربندیApache ، بررسی کنید که تنظیمات زیر وجود داشته باشند:

```
SSLHonorCipherOrder On
SSLCipherSuite ALL:!EXP:!NULL:!LOW:!SSLv۲:!RC٤:!aNULL
```

برای اعمال تنظیمات پیشنهادی:

۱. اطمینان حاصل کنید که SSLCipherSuite شامل موارد زیر باشد:

  - !NULL, !SSLv۲, !RC٤, !aNULL

۲. این تنظیمات را در سطح سرور و تمام VirtualHostهایی که TLS دارند، اضافه یا به‌روزرسانی کنید:

برای تقویت بیشتر از این مقدار استفاده کنید:

```
SSLHonorCipherOrder On
SSLCipherSuite EECDH:EDH:!NULL:!SSLv۲:!RC٤:!aNULL:!۳DES:!IDEA
```

مقدار پیش‌فرض SSLCipherSuite به نسخه OpenSSL بستگی دارد.

مقدار پیش‌فرض SSLHonorCipherOrder برابر با Off است.

- **اطمینان از غیرفعال بودن مذاکره مجدد ناامن SSL**

مراحل زیر را انجام دهید:

- در فایل‌های پیکربندی Apache جستجو کنید که آیا دستور SSLInsecureRenegotiation وجود دارد یا نه.
- اگر وجود دارد، بررسی کنید که مقدار آن off باشد.
- اگر این دستور اصلاً وجود ندارد، به معنای آن است که حالت پیش‌فرض (یعنی غیرفعال) فعال است و مشکلی نیست.

اگر دستور SSLInsecureRenegotiation در پیکربندی وجود دارد و مقدار آن on است، آن را به صورت زیر تغییر دهید:

SSLInsecureRenegotiation off

اگر این دستور وجود ندارد، نیازی به هیچ اقدامی نیست، زیرا مقدار پیش‌فرض آن برابر با off است.

- **اطمینان از غیرفعال بودن فشرده‌سازی SSL**

برای Apache نسخه ۲.۲.۲۶ و بالاتر:

۱. فایل‌های پیکربندی Apache را بررسی کنید:
   - به دنبال دستور SSLCompression بگردید.
۲. اطمینان حاصل کنید که:

- یا این دستور اصلاً وجود ندارد که یعنی پیش‌فرض offاست
- یا اگر وجود دارد، مقدار آن off باشد:

SSLCompression off

برای Apache نسخه ۲.۲.۲۴ و ۲.۲.۲۵

- حتماً دستور SSLCompression باید وجود داشته باشد و مقدار آن off باشد، چون پیش‌فرض در این نسخه‌ها on بوده است.

نسخه‌های قبل از ۲.۲.۲۴

- این نسخه‌ها امکان غیرفعال‌سازی فشرده‌سازی SSL را ندارند و بنابراین غیراستاندارد محسوب می‌شوند.

این موارد را در فایل /etc/httpd انجام دهید

- **اطمینان از غیرفعال بودن الگوریتم‌های رمزنگاری با قدرت متوسط در SSL/TLS**

از ابزار sslscan برای بررسی وجود الگوریتم‌های ۳DES و IDEA استفاده کنید:

sslscan --no-colour www.example.com | egrep 'IDEA|DES'

اگر خروجی‌ای شامل الگوریتم‌های ۳DES یا IDEA دریافت کردید مثلاً DES-CBC۳-SHA یعنی هنوز فعال هستند و باید غیرفعال شوند.

برای غیرفعال کردن الگوریتم‌های با قدرت متوسط، مراحل زیر را انجام دهید:

۱. در سطح سرور و در تمامی VirtualHostهایی که SSL/TLS دارند، دستورات زیر را اضافه یا ویرایش کنید:

```
SSLHonorCipherOrder On
SSLCipherSuite ALL:!EXP:!NULL:!LOW:!SSLv۲:!RC٤:!aNULL:!۳DES:!IDEA
```

برای پیروی کامل از همه توصیه‌های امنیتی، از مقدار پیشنهادی زیر استفاده کنید که هم الگوریتم‌های ضعیف و هم متوسط را حذف می‌کند:

```
SSLHonorCipherOrder On
SSLCipherSuite EECDH:EDH:!NULL:!SSLv۲:!RC٤:!aNULL:!۳DES:!IDEA
```

- **اطمینان از در دسترس بودن تمامی محتوای وب از طریق HTTPS**

برای بررسی اینکه آیا حالت پیشنهادی پیاده‌سازی شده است، مراحل زیر را انجام دهید:

۱. دریافت لیست آدرس‌های IP که در فایل‌های پیکربندی Apache گوش می‌دهند:

از دستور زیر استفاده کنید (ابتدا مسیر فایل‌های پیکربندی را تنظیم کنید):

```
CONF_DIRS="/etc/httpd/conf /etc/httpd/conf.d /etc/httpd/conf_dir۲ ..."
CONFS=$(find $CONF_DIRS -type f -name '*.conf')
IPS=$(egrep -ih '^\s*Listen ' $CONFS | egrep -iv '(:٤٤٣\b)|https' | cut -d' ' -f۲)
```

۲. دریافت لیست نام‌های میزبان مجازی (Virtual Hosts)

```
VHOSTS=$(egrep -iho '^\s*<VirtualHost .*>' $CONFS | egrep -io '\s+[A-Z:.۰-۹]+>$' | tr -d '
>')
```

۳. تولید URLهای HTTP برای تست:

```
URLS=$(for h in $LIPADDR $VHOSTS ; do echo "http://$h/"; done)
```

۴. بررسی اینکه آیا محتوای مهم از طریق HTTP ارائه می‌شود:

می‌توانید آدرس‌ها را در مرورگر باز کنید یا از ابزار خط فرمان curl استفاده کنید:

```
for u in $URLS ; do echo -e "\n\n\n=== $u ==="; curl -fSs $u | head -c ۳۰۰ ; done
```

نشانه‌های عدم انطباق (غیراستاندارد بودن):

اگر URL موردنظر محتوای HTML واقعی برگرداند (به‌جای خطا یا ریدایرکت)، این URL غیراستاندارد است.

مثال استاندارد (ریدایرکت):

```
=== http://www.cisecurity.org/ ===
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML ۲,۰//EN">
<html><head><title>۳۰۱ Moved Permanently</title></head><body>
<h۱>Moved Permanently</h۱>
<p>The document has moved <a href="https://www.cisecurity.org/">here</a>.</p>
</body></html>
```

برای پیاده‌سازی وضعیت پیشنهادی:

- محتوای وب‌سایت را به یک سرور دارای TLS پروتکل HTTPS منتقل کنید.
- در فایل پیکربندی Apache یک دستور ریدایرکت دائمی (Permanent Redirect) اضافه کنید مانند نمونه زیر:

```
Redirect permanent / https://www.cisecurity.org/
```

TLS به صورت پیش‌فرض فعال نیست.

- **غیرفعال‌سازی پروتکل‌های TLSv۱،۰ و TLSv۱،۱**

برای بررسی اینکه آیا حالت پیشنهادی پیاده‌سازی شده است:

- فایل‌های پیکربندی Apache را جست‌وجو کنید و مطمئن شوید که دستور SSLProtocol یکی از مقادیر زیر را دارد:

SSLProtocol TLSv۱،۲ TLSv۱،۳

برای پیاده‌سازی وضعیت پیشنهادی، مراحل زیر را انجام دهید:

۱. بررسی کنید که آیا Apache از TLSv۱،۳ پشتیبانی می‌کند یا خیر.
برای این کار یا بررسی کنید که نسخه OpenSSL برابر یا جدیدتر از ۱.۱.۱ باشد:

```
$ openssl version
OpenSSL ۱،۱،۱a ۲۰ Nov ۲۰۱۸
```

یا مقدار TLSv۱،۳ را به دستور SSLProtocol در فایل پیکربندی اضافه کرده و با دستور زیر صحت آن را بررسی کنید:

```
# httpd -t
Syntax OK
```

۲. در فایل‌های پیکربندی Apache، دستور SSLProtocol را بیابید.
اگر وجود ندارد، آن را اضافه کنید، یا مقدار آن را به یکی از گزینه‌های زیر تغییر دهید:

SSLProtocol TLSv۱،۲

یا اگر TLSv۱،۳ پشتیبانی می‌شود:

SSLProtocol TLSv۱،۲ TLSv۱،۳

- **اطمینان از فعال بودن HTTP Strict Transport Security (HSTS)**

برای بررسی فعال بودن HSTS یکی از روش‌های زیر را اجرا کنید:

۱. بررسی در فایل پیکربندی Apache

برای هر میزبان مجازی که SSL فعال دارد، بررسی کنید که دستور زیر موجود باشد و مقدار max-age حداقل ۴۸۰ ثانیه (۸ دقیقه) یا بیشتر باشد:

Header always set Strict-Transport-Security "max-age=٦٠٠"

۲. بررسی با ابزار OpenSSL

می‌توانید به سرور HTTPS متصل شوید و بررسی کنید که هدر HSTS در پاسخ وجود دارد:

openssl s_client -connect www.example.com:٤٤٣

سپس در خروجی به دنبال هدر زیر باشید:

Strict-Transport-Security: max-age=٦٠٠

برای پیاده‌سازی حالت پیشنهادی در سطح پیکربندی سرور Apache و در هر میزبان مجازی که SSL فعال دارد، یکی از دستورات زیر را اضافه کنید:

با گزینه‌های اختیاری:

Header always set Strict-Transport-Security "max-age=٦٠٠; includeSubDomains; preload"

یا فقط مقدار پایه:

Header always set Strict-Transport-Security "max-age=٦٠٠"

- **اطمینان از فعال بودن فقط مجموعه رمزنگاری‌هایی که از Forward Secrecy پشتیبانی می‌کنند**

برای بررسی اینکه فقط مجموعه رمزهای FS فعال هستند، یکی از روش‌های زیر را انجام دهید:

۱. استفاده از ابزار sslscan

می‌توانید با استفاده از ابزار sslscan به سرور متصل شوید و لیست cipherهای مجاز را ببینید.

$ sslscan --no-colour --no-failed www.example.com | egrep '(^Accepted)|(^Preferred)' | egrep -v '(ECDHE-)|(DHE-)'

اگر خروجی نمایش داده نشد، یعنی فقط رمزنگاری‌های FS فعال هستند.

اگر خروجی داشت، یعنی برخی الگوریتم‌های غیراستاندارد مانند RSA فعال هستند.

۲. بررسی پیکربندی Apache

فایل پیکربندی Apache و هر میزبان مجازی دارای SSL را بررسی کنید و بخش SSLCipherSuite را پیدا کنید. با دستور زیر می‌توانید لیستcipher های مجاز را بررسی و مطمئن شوید که فقط الگوریتم‌های FS مجاز هستند:

$ openssl ciphers -v 'EECDH:EDH:!NULL:!SSLv٢:!RC٤:!٣DES:!IDEA:!aNULL:!SHA١'

در خروجی، فقط مواردی با ECDHE- یا DHE- باید نمایش داده شوند.

برای پیاده‌سازی تنظیمات پیشنهادی در پیکربندی سطح سرور Apache و در همه میزبان‌های SSL/TLS ، خط زیر را اضافه یا ویرایش کنید:

SSLCipherSuite EECDH:EDH:!NULL:!SSLv۲:!RC٤:!aNULL:!۳DES:!IDEA

در نسخه‌های جدید OpenSSL (مثل ۱.۰.۲ و بالاتر)، از ECDHE به‌جای EECDH و DHE به‌جای EDH استفاده می‌شود. استفاده از ECDHE و DHE توصیه می‌شود تا با خروجی ابزارهای بررسی تطابق داشته باشد.

- **فعال‌سازی OCSP Stapling**

فایل پیکربندی اصلی SSL در مسیر:
conf/extra/httpd-ssl.conf

این فایل شامل تنظیمات کلی SSL و همچنین VirtualHost پیش‌فرض SSL است. این دو خط را قبل از بخش SSL Virtual Host Context ## قرار دهید:

```
SSLUseStapling On
SSLStaplingCache shmcb:logs/ssl_stapling(۳۲۷٦۸)
```

از نسخه ۲.۴.۱۱ به بعد، این دستورات به‌صورت کامنت‌شده در فایل وجود دارند. کافیست آن‌ها را از حالت کامنت خارج کنید.

- **اضافه کردن CSP**

برای اعمال CSP در Apache، خط زیر را به فایل /etc/apache۲/sites-enabled/example.conf اضافه کنید:

```
Header always set Content-Security-Policy "default-src 'self'; font-src *; img-src * data:;
script-src *; style-src *;"
```

۸ مراجع

- https://duo.com/decipher/chrome-and-firefox-removing-ev-certificate-indicators

- IT Security Guidelines for Transport Layer Security (TLS), V۲،۱, National Cyber Security Centre.
- https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices
- CIS Benchmarks