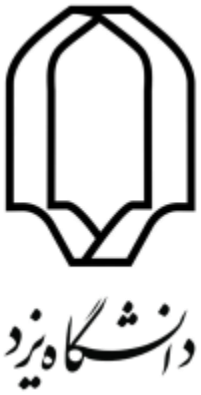


دانشکده مهندســی کامپیوتر

گزارش پروژه دوره کارشناسی در رشته مهندسی کامپیوتر (مهندسی نرم‌افزار)

بررسی امنیت برنامه‌های تحت وب

نگارش:

عرفانه فلاح‌پور

استاد راهنما:

دکتر فضل‌الله ادیب نیا

شهریور ۱۴۰۱

اظهارنامه

این‌جانب عرفانه فلاح‌پور دانشجوی رشته‌ی مهندسی کامپیوتر گرایش مهندسی نرم‌افزار دانشکده‌ی مهندسی اظهار می‌کنم که این گزارش پروژه حاصل تلاش‌های این‌جانب بوده و درجاهایی که از منابع دیگران استفاده کرده‌ام، نشانی دقیق و مشخصات کامل آن را نوشته‌ام. همچنین اظهار می‌کنم که تحقیق و موضوع پایان‌نامه تکراری نیست و تعهد می‌نمایم که بدون مجوز دانشگاه دستاوردهای آن را منتشر ننموده و یا در اختیار غیر قرار ندهم. کلیه حقوق این اثر مطابق با آیین‌نامه مالکیت فکری و معنوی متعلق به دانشگاه یزد است.

نام و نام خانوادگی:

عرفانه فلاح‌پور

تاریخ و امضا:

۱۴۰۱/۰۶/۱۵

سپاسگزاری

پیش از هر چیز از خداوند متعال به خاطر تمامی نعمت‌هایی که هدیه کرده سپاسگزارم.

از اساتید ارجمند، که در طول دوره کارشناسی همواره مشوق و راهنمای من بوده‌اند، صمیمانه تشکر و قدردانی می‌نمایم. همچنین از اعضای خانواده‌ام که در تمام دوران تحصیل با روی همواره گشاده مرا یاری کرده‌اند نیز سپاسگزارم.

همچنین از استاد ارجمند، **جناب دکتر فضل‌الله ادیب نیا** برای راهنمایی و همکاری با این‌جانب در این مدت و صبر بی‌کران ایشان برای تحقیق و توسعه این پروژه بسیار ممنونم.

چکیده

به کوشش
عرفانه فلاح‌پور

در بهار و تابستان سال ۱۴۰۱ با همیاری دکتر فضل‌الله ادیب نیا، استادیار و راهنمای بنده در این پروژه، تلاش کردم ضمن بررسی و معرفی آسیب‌پذیری‌های موجود برای برنامه‌های تحت وب، ابزاری برای پیدا کردن آسیب‌پذیری XSS پیاده‌سازی کنم. هدف از این گزارش معرفی OWASP و BUG Bounty، نحوه کار و کسب درآمد در حوزه امنیت و بررسی آسیب‌پذیری‌های مطرح در دنیای وب و نحوه تأثیر آن بر برنامه‌های تحت وب، است.

OWASP Top 10 فهرستی از آسیب‌پذیرهای مربوط به برنامه‌های تحت وب است. این ۱۰ آسیب‌پذیری به‌طور کامل بررسی‌شده و مثال‌هایی از آن‌ها، در دنیای واقعی مطرح می‌شود. همچنین راهکار و ابزارهای نوشته‌شده در سال‌های اخیر برای پیدا کردن این آسیب‌پذیری‌ها و گزارش‌هایی که به‌طور واقعی از آن‌ها ارائه‌شده است، مطرح می‌شود و برای جلوگیری از حمله مهاجمان راهکارهایی معرفی می‌شود.

کلیدواژه‌ها: OWASP, SQL injection, Command injection, Code injection, Broken Authentication, Cross-Site Script, Security Misconfiguration, XML External Entity, Sensitive Data Exposure, Server-Side Request Forgery, Broken Access Control, Security Logging and Monitoring Failures, Cross-Site Request Forgery

فهرست مطالب

فصل اول ۲

فصل ۱- تعریف پروژه و مفاهیم کلیدی ۲

۱-۱- مقدمه ۲

۱-۲- تعریف «OWASP» ۲

۱-۳- تعریف «Bug Bounty» ۳

۱-۴- نحوه کسب درآمد در دنیای امنیت ۴

فصل دوم ۸

فصل ۲- بررسی آسیب‌پذیری‌های «OWASP Top۱۰» ۸

۲-۱- Injection ۸

۲-۲- Cross Site Script ۱۴

۲-۳- Cross-Site Request Forgery ۲۱

۲-۴- Broken Authentication ۲۲

۲-۵- Sensitive Data Exposure ۳۱

۲-۶- Server-Side Request Forgery ۳۴

۲-۷- XML External Entity ۳۸

۲-۸- Broken Access Control ۴۱

۲-۹- Security Misconfiguration ۴۴

۲-۱۰- Security Logging and Monitoring Failures ۴۶

فصل سوم ۴۹

فصل ۳- توضیحات مرتبط با ابزارهای پیاده‌سازی شده ۴۹

۳-۱- ابزار «dozd» ۴۹

۳-۲- ابزار «fallparams-master» ۴۹

۳-۳- ابزار «ayine» ۵۰

۳-۴- نمونه های ورودی و خروجی ۵۰

فصل ۴- فهرست مراجع ۵۳

فهرست شکل‌ها

شکل ۱–۱ - ترتیب آسیب‌پذیری‌های «*OWAP*» در سال ۲۰۱۷ و سال ۲۰۲۱..... ۳
شکل ۲–۱ - نمونه‌ای از گزارش «SQL Injection»..... ۹
شکل ۲–۲ - ابزار «SQLmap»..... ۱۰
شکل ۲–۳ - سناریو «Command Injection»..... ۱۰
شکل ۲–۴ - نمونه‌ای از گزارش «Command Injection»..... ۱۱
شکل ۲–۵ - ابزار «Commix»..... ۱۱
شکل ۲–۶ - ابزار TPLmap..... ۱٤
شکل ۲–۷ - نمونه‌ای از کدHTML و DOM مربوط به آن..... ۱٤
شکل ۲–۸ - روند تغییر «DOM» توسط جاوا اسکریپت..... ۱٥
شکل ۲–۹ - سناریو «Same Origin Policy»..... ۱٦
شکل ۲–۱۰ - Cross-Origin Resource Sharing..... ۱٦
شکل ۲–۱۱ - نمونه‌ای از درخواست «simple» در جاوا اسکریپت..... ۱۷
شکل ۲–۱۲ - سناریو «Cross-site Script»..... ۲۰
شکل ۲–۱۳ - سناریو «Cross Site Request Forgery»..... ۲۲
شکل ۲–۱۴ - سناریو مربوط به تعریف کوکی احراز شده..... ۲۳
شکل ۲–۱۵ - تعدادی از «Claim» های مربوط به «JWT»..... ۲٥
شکل ۲–۱۶ - اجزای «JWT»..... ۲٦
شکل ۲–۱۷ - ابزار «JWT»..... ۲۷
شکل ۲–۱۸ – سناریو فرآیند «OAuth»..... ۲۷
شکل ۲–۱۹ - سناریو «SSO»..... ۲۹
شکل ۲–۲۰ - سناریو «SSO» با «JASONP» یا «AjaxCall»..... ۲۹
شکل ۲–۲۱ - سناریو حمله «JASONP»..... ۳۰
شکل ۲–۲۲ - سناریو «Sensitive Data Exposure»..... ۳۲
شکل ۲–۲۳ - ابزار «Wfuzz»..... ۳٤
شکل ۲–۲۴ - نمونه‌ای از گزارش یک «SSRF»..... ۳٥
شکل ۲–۲٥ - سناریوی حمله «SSRF»..... ۳٥
شکل ۲–۲۶ - XML external entity..... ۳۸
شکل ۲–۲۷ - XXE + SSRF..... ٤۰
شکل ۲–۲۸ - ابزار «Burp Suite»..... ٤۰
شکل ۲–۲۹ - سناریو «Broken Access Control»..... ٤۲
شکل ۲–۳۰ - نمونهای از «Security Misconfiguration»..... ٤٥
شکل ۲–۳۱ - چگونگی تاثیر ثبت و نظارت به شناسایی الگوهای یک سیستم..... ٤٦
شکل ۳–۱ - ورودی ابزار «fallparams»..... ٥۰
شکل ۳–۲ - ورودی ابزار «ayine»..... ٥۱
شکل ۳–۳ - خروجی ابزار «ayine»..... ٥۱

فهرست جدول‌ها

جدول ۲–۱ - بررسی سیاست دسترسی به «origins» دیگر ۱۵
جدول ۲–۲ - انواع الگوریتم برای محاسبه «Signatures»..... ۲۵
جدول ۲–۳ - خلاصه‌ای از توضیحات«Claim» مربوط به نمونه«JWT» ۲۶

فصل اول

فصل ۱- تعریف پروژه و مفاهیم کلیدی

۱-۱- مقدمه

همان‌طور که از عنوان این پروژه پیداست، این پروژه از مقاله‌ای در ارتباط با بررسی امنیت برنامه‌های تحت وب و یک ابزار پیاده‌سازی شده برای تشخیص نوعی آسیب‌پذیری تشکیل شده‌است. این گزارش با قصد معرفی، شناسایی و برطرف‌کردن آسیب‌پذیری‌های برنامه‌های تحت وب طراحی شده‌است و در فصل پایانی ابزار تشخیص یک نوع از آسیب‌پذیری (XSS) به عنوان نمونه معرفی شده‌است.

به‌طورکلی آسیب‌پذیری‌های ده‌گانه (که توسط اجمن «OWASP[1]» معرفی شده‌اند)، در طبقه‌بندی آسیب‌پذیری‌هایی با خطر بالقوه بالا قرار دارند. در نتیجه کشف و جلوگیری از آن‌ها برای برقراری امنیت برنامه‌های تحت وب ضرروی است.

در بخش‌های آتی، علاوه بر معرفی ده آسیب‌پذیری به سناریوهای حمله، روش‌های جلوگیری از هر کدام و ابزارهای تشخیص آن‌ها پرداخته می‌شود. یک فرد متخصص امنیت باید شناخت کافی نسبت به آسیب‌پذیری‌های معرفی شده و نحوه جلوگیری از آن‌ها را داشته باشد.

ابزار پیاده‌سازی شده می‌تواند با بدست‌آوردن تمامی مقادیری از یک وب‌سایت و زیر دامنه‌های آن، که احتمال آسیب‌پذیری XSS در آن‌ها وجود دارد، نتیجه را به صورت لیستی از مقادیر آسیب‌پذیر برگرداند.

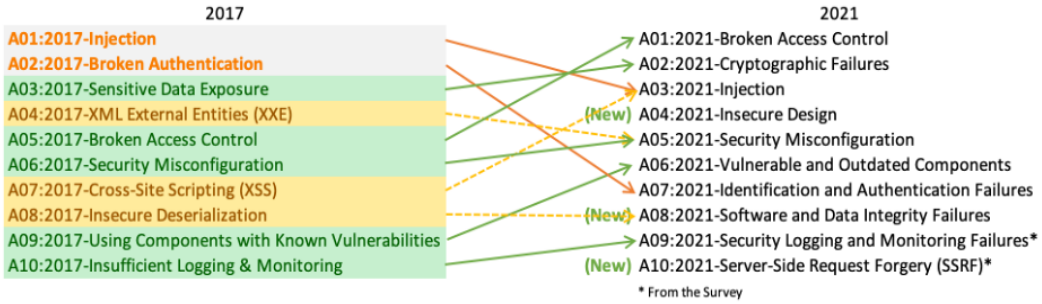
۱-۲- تعریف «*OWASP*»

«*OWASP*» یک انجمن آنلاین است که مقالات، روش‌شناسی‌ها[2]، اسناد، ابزارها و فن‌آوری‌های رایگان و در دسترس را در زمینه امنیت برنامه‌های کاربردی وب تولید می‌کند. «*OWASP*» به هیچ‌یک از شرکت‌های فناوری اطلاعات وابسته نیست. از طرفی تقریباً همه‌ی افرادی که با «*OWASP*» در ارتباط هستند، به‌صورت داوطلبانه این کار را انجام می‌دهند.

«*OWASP Top 10*» فهرستی از آسیب‌پذیری‌های مربوط به برنامه‌های تحت وب[3]، همراه با خطر، تأثیر و اقدامات متقابل است. این فهرست معمولاً در هر ۳ یا ۴ سال تجدید می‌شود. این آسیب‌پذیری‌ها، یک آگاهی

[1] Open Web Application Security Project
[2] Methodology
[3] Web application

عمومی و استاندارد برای توسعه‌دهندگان تحت وب بوده که می‌تواند دید وسیعی را در ارتباط با رایج‌ترین آسیب‌پذیری‌هایی که وب‌سایت یک شرکت را تهدید می‌کند در اختیار آن سازمان قرار دهد. [1]



شکل ۱–۱ - ترتیب آسیب‌پذیری‌های «*OWAP*» در سال ۲۰۱۷ و سال ۲۰۲۱

ترتیب این آسیب‌پذیری‌ها همان‌گونه که در شکل ۱-۱ نشان داده‌شده است در سال ۲۰۲۱ به‌صورت زیر هست:

1. *Broken Access Control*
2. *Cryptographic Failures*
3. *Injection*
4. *Insecure Design*
5. *Security Misconfiguration*
6. *Vulnerable and Outdated Components*
7. *Identification and Authentication Failures*
8. *Software and Data Integrity Failures*
9. *Security Logging and Monitoring Failures*
10. *Sever-Side Request Forgery (SSRF)*

در ادامه به بررسی این آسیب‌پذیری‌ها و آسیب‌پذیری‌هایی که درگذشته در لیست « *OWASP Top 10*» وجود داشته و امروزه نیز در بسیاری از موارد امنیتی برنامه‌های تحت وب نقش مهمی دارند، پرداخته می‌شود.

۱-۳- تعریف «*Bug Bounty*»

«*Bug Bounty*» به‌طورکلی برنامه‌ای برای یافتن باگ‌ها و آسیب‌پذیری وب‌سایت‌ها و برنامه‌ها است که توسط متخصصان امنیتی و هکرهای کلاه‌سفید[1] به افزایش امنیت یک سایت یا برنامه کمک می‌کند. آن‌ها با

[1] White Hat Hacker

یافتن باگ‌هایی که زمینه سوءاستفاده و نفوذ را فراهم می‌کند، و گزارش آن به توسعه‌دهندگان پاداش دریافت می‌کنند. البته این یک معامله دو سر برد است، متخصص در ازای یافتن اشکالات پاداش می‌گیرد و توسعه‌دهنده امنیت وب‌سایت یا برنامه تحت وب خود را بالا می‌برد و مشکلات را قبل از عمومی شدن و ایجاد مشکلی بزرگ برطرف می‌کند. در مواردی که یک سایت یا نرم‌افزار یک مشکل جدی دارد که در آینده باعث از دست رفتن اطلاعات مهم برای سازنده خواهد شد، می‌توان با برقراری ارتباط با مدیر سایت یا نرم‌افزار آن‌ها را مطلع کرد و پاداش خود را دریافت کرد و قبل از اینکه مشکل فراگیر شود و مورد سوءاستفاده قرار بگیرد مشکل را برطرف کرد و سایت یا نرم‌افزار را به‌روزرسانی کرد.

برنامه‌های «*Bug Bounty*» معمولاً به‌عنوان بخشی از استراتژی‌های مدیریت آسیب‌پذیری و آزمودن نفوذ و آزمودن امنیت کدها نیز، استفاده می‌شوند. به متخصصان «*Bug Bounty*»، «*Bounty hunter*» گفته می‌شود. برنامه‌های«*Bug Bounty*» توسط تعداد زیادی از سازمان‌ها، ازجمله «*Mozilla*»، «*Facebook*»، «*Yahoo*»، «*Google*»، «*Reddit*»،«*Square*» و«*Microsoft*» اجرا می‌شوند.

۱-٤- نحوه کسب درآمد در دنیای امنیت

هک و امنیت یکی از حوزه‌های کامپیوتر و «IT» است که در دنیای امروز بسیار پرطرفدار و همچنین بسیار کاربردی است. هرکسی اگر سیستمی داشته‌باشد، فایل‌های با ارزشی داشته‌باشد و به‌طورکلی اگر نیازمند امنیت باشد با توجه به پیشرفت فنّاوری و پیشرفت هکرها به یک متخصص امنیتی نیاز خواهد داشت. متخصصان امنیت و هکرها باید آزمون‌های امنیتی و شبکه و هک را پشت سر بگذارند و مطالعات زیادی داشته‌باشند تا بتوانند دانش مورد نیاز را کسب کنند و از سیستم‌های مختلف حفاظت کنند. حوزه هک و امنیت یک حوزه نوپا است که در ایران و تقریباً در خارج از کشور هنوز در آن جایگاهی که باید باشد نیست اما با این‌حال می‌توان درآمد قابل‌توجهی از این حوزه کسب کرد.

برخی از روش‌هایی که می‌توان در این حوزه درآمد کسب کرد در ادامه معرفی می‌شوند:

۱) تدریس هک و امنیت

شاید بتوان گفت مهم‌ترین بخش و سریع‌ترین راهی که می‌توان در هک و امنیت به درآمد رسید راه تدریس است. در ایران بیشتر نوجوانان و جوانان به هک و امنیت علاقه‌مند هستند و خیلی مصمم هستند که این مسیر را دنبال کنند و فردی که در این زمینه فعالیت کرده و دانش خوبی دارد، به‌عنوان مدرس دانش و تجربه‌ای را که کسب کرده را می‌تواند به آن‌ها انتقال دهد و درازای آن پول دریافت کند. برای آموزش می‌توان در آموزشگاه‌های حضوری کارکرد یا در خانه برای وب‌سایت‌ها دوره آموزشی آنلاین تهیه کرد، یا فرد مدرس برای خود یک بسته آموزشی درست کند و بفروشد. بسته به اینکه کدام راه انتخاب می‌شود سختی‌ها و درآمدهای مختلفی خواهد

داشت. به‌عنوان‌مثال وب‌سایت‌های آموزشی معمولاً ساعتی کار می‌کنند یعنی فرد یک دوره آموزشی ویدیویی تهیه می‌کند و به ازای هر ساعت دوره، مبلغی را دریافت می‌کند.

۲) استخدام در شرکت‌ها و سازمان‌ها

این راه سختی‌های خاص خودش را با توجه به‌جایی که فرد می‌خواهد آنجا استخدام شود دارد. یک فردی که در زمینه امنیت فعالیت می‌کند باید شرایط خاصی را داشته باشد، در موارد خاصی تسلط داشته باشد و بتواند نیازهای شرکت یا فردی که می‌خواهد او را استخدام کند را رفع کند. می‌توان در سازمان‌های دولتی و یا خصوصی استخدام شود و مشغول کار شود که با این روش درآمد حدودی ۱۳ تا ۱۵ میلیون تومان (در زمان نگارش این گزارش) را برای شروع دارد، که درآمد نسبتاً خوبی به شمار می‌آید. همچنین یک راه کسب درآمد دیگر برای هکرها را که می‌توان در محدوده استخدام قرارداد، استخدام برای محافظت از صفحه‌های اینستاگرام، فایل‌ها و تأمین امنیت وب‌سایت‌ها است، که البته زیاد به دانش خیلی پیشرفته‌ای نیاز ندارد اما حتی بسیاری از شرکت‌های بزرگ نیز برای چک کردن سرورها و حملاتی که به وب‌سایت‌هایشان وارد می‌شود از این افراد کمک می‌گیرند و در ازای آن هزینه‌ای پرداخت می‌کنند.

۳) *Bug Bounty*

فردی که در این حوزه کار می‌کند به‌عنوان یک متخصص امنیتی و هکر پیشرفته یا برنامه‌نویس حرفه‌ای می‌تواند نقص سیستم‌ها را پیدا کند و در ازای آن هزینه‌ای دریافت کند. برای این کار وب‌سایت‌هایی مانند «*Hackerone*[1]» و «*Bugcrowd*[2]» وجود دارند که فرد می‌تواند باگ‌هایی که پیدا می‌کند را به آن‌ها گزارش بدهد و در ازای آن مقداری بیت کوین یا دلار دریافت کند. همچنین شرکت‌های خصوصی از پیشرفته‌ترین‌ها مثل مایکروسافت و اپل و .. تا شرکت‌های نوپا و حتی شرکت‌های ایرانی نیز گاهی اوقات برای برنامه‌ای که تازه منتشر می‌کنند جایزه‌ای برای پیدا کردن مشکلات آن قرار می‌دهند و می‌توان در این راه نیز کسب درآمد کرد. که البته کار سختی است زیرا پیدا کردن باگ‌های برنامه‌های معروفِ شرکت‌های بزرگ نیازمند دانش فراوان و ابزارهای پیشرفته‌ای است. اما مثال‌هایی در دنیای امروزی وجود دارد که افرادی باتجربه کم و سن پایین و بدون هیچ ابزار پیشرفته‌ای، توانستند ایرادات اساسی از برنامه‌های بزرگی را پیدا کنند و در ازای گزارش آن، مبلغ زیادی دریافت کنند و یا به استخدام همان شرکت دربیایند. درنتیجه نباید از پیشرفت و کسب درآمد در این حوزه ناامید شد.

۴) ساخت و فروش ابزار امنیتی و هک

این بخش یکی از راه‌های بحث‌برانگیز در کسب درآمد هکرها شناخته می‌شود. درواقع می‌توان یک ابزار امنیتی را یک چاقو دید که می‌توان با آن هم به انسان‌ها آسیب رساند و هم می‌توان با آن میوه پوست کند یا کارهای روزمره‌ای که خیلی می‌تواند مفید باشد انجام داد. اما در کل ساخت ابزار کار غیرقانونی نیست و بستگی

[1] www.hackerone.com
[2] www.bugcrowd.com

به فرد دارد که چه استفاده‌ای از این ابزار می‌برد. اگر فردی برنامه‌نویس و هکر خوبی باشد می‌تواند ابزارهای امنیتی را بنویسد و در سایت‌های مربوطه بفروشد و درآمد خود را داشته باشد.

۵) مدیر امنیت شبکه

فردی که دانش‌پایه را در این حوزه دارد، به‌عنوان یک متخصص امنیتی می‌تواند از شبکه‌ها و سیستم‌های ادارات مختلف محافظت کند و از این طریق درآمد داشته باشد. در این راه فرد دائماً با انواع حملات هکرها روبه‌رو می‌شود و دائماً باید ذهن خود را به چالش بکشد و با خود فکر کند که یک هکر چگونه می‌تواند به شبکه آسیب بزند و آن حفره امنیتی را برطرف کند. همان‌طور که گفته شد فرد با یک شغل پُر از چالش و هیجان روبه‌رو هست اما ممکن است برای او سخت هم باشد. بااین‌حال حقوق پایه برای انجام این کار حدود ۴ هزار دلار (در زمان نگارش این متن حدود ۱ میلیارد و ۲۰۰ میلیون ریال) است و بسته به توانایی‌ها و سابقه کاری فرد می‌تواند بیشتر هم بشود.

۶) مهندس امنیت برنامه

این شاخه یکی دیگر از پُر درآمدترین مشاغل متخصص‌های امنیت سایبری است. مهندسان امنیت برنامه، به‌طور متوسط بین ۱۲۰،۰۰۰ تا ۱۸۰،۰۰۰ دلار (در زمان نگارش این متن بین ۳ میلیارد تا ۵ میلیارد ریال) درآمد دارند. اگر شرکتی از راه‌حل‌های نرم‌افزاری ارائه‌شده یا میزبانی‌شده توسط سازمان‌های شخص ثالث مانند «*AWS*» یا «*Microsoft's Azure*» استفاده می‌کند یا حتی اگر راه‌حل‌های خود را سفارشی می‌کند، استخدام یک مهندس امنیت برنامه برای این شرکت بسیار مهم است.

این متخصصان وظیفه دارند از ایمنی تمامی نرم‌افزارها و برنامه‌های کاربردی که توسط تمامی نیروهای کار شرکت استفاده می‌شوند، اطمینان حاصل کنند. تمام محدودیت‌های حریم خصوصی و انطباق در نرم‌افزار باید تعبیه شود و از آن‌ها پیروی شود.

۷) بررسی و آنالیز وب‌سایت

یکی از راه‌هایی که در خارج از کشور می‌توان در حوزه هک و امنیت درآمد داشت بررسی و آنالیز وب‌سایت‌ها است که در ایران، هرچند به میزان قابل‌توجهی کمتر، وجود دارد. به‌عنوان مثال اگر فردی سابقه کاری خوبی داشته باشد و تقریباً در سطح حرفه‌ای باشد برای آنالیز و بررسی یک وب‌سایت می‌تواند ساعتی ۶۰ دلار (در زمان نگارش این متن ۱۸ میلیون ریال) حقوق دریافت کند. حتی اگر سطح فرد متوسط و رو به پایین باشد نیز ساعتی ۳۰ دلار (در زمان نگارش این متن ۹ میلیون ریال) درآمد خواهد داشت که همچنان درآمد بالایی محسوب می‌شود. همچنین باید توجه داشت درآمدهایی که گفته شد ثابت نیستند و با توجه به توانایی‌های فرد ممکن است بیشتر شوند.

۸) تستر نفوذ

تسترهای نفوذ که معمولاً «*Pen Testers*» یا هکر اخلاقی نامیده می‌شوند به‌طور متوسط بین ۸۰۰۰۰ تا ۱۳۰۰۰۰ دلار (در زمان نگارش این متن بین ۲ میلیارد تا ۴ میلیارد ریال) درآمد دارند. یک نظرسنجی «*McAfee*» نشان داد که مدیران امنیتی معتقدند استخدام هکرهای اخلاقی، درک ارزشمندی از منطق استفاده از هکرها و مهارت‌های حیاتی آن‌ها برای امنیت سایبری به شرکت ارائه می‌دهد.

اگر شرکتی تست‌های امنیتی سه‌ماهه، ماهانه یا روزانه انجام می‌دهد، پس نیازمند افراد حرفه‌ای هستند که باید روی جذب و حفظ آن‌ها سرمایه‌گذاری کند. تسترهای نفوذ تست‌های مختلف و عمیق را در سیستم‌های کامپیوتری، شبکه‌ها و حتی برنامه‌های کاربردی وب شرکت انجام می‌دهند تا آسیب‌پذیری‌هایی را که می‌توانند توسط مجرمان سایبری مورد سوءاستفاده قرار گیرند را، شناسایی کنند.

۹) مهندس امنیت سایبری

شغل مهندس امنیت سایبری نیز یکی از بالاترین درآمدها را در صنعت متخصص‌های امنیت به همراه دارد که میانگین حقوق متخصص امنیت سایبری از ۱۲۰۰۰۰ تا ۲۰۰۰۰۰ دلار (در زمان نگارش این متن بین ۳ میلیارد تا ۶ میلیارد ریال) است. شرکت‌ها برای مجموعه مهارت‌ها و تجربه‌هایشان روی این متخصصان سرمایه‌گذاری می‌کنند، زیرا آن‌ها در درجه اول مسئول چندین کارکرد مهندس امنیت هستند، ازجمله این کارکردها می‌توان به طراحی، توسعه و پیاده‌سازی راه‌حل‌های شبکه امن برای دفاع در برابر حملات سایبری پیشرفته، تلاش‌های مقابله با هک و تهدیدهای مداوم نام برد.

۱۰) مدیر امنیت اطلاعات

طبق راهنمای حقوق و دستمزد صنعت فناوری و دیجیتال مارکتینگ در سال ۲۰۲۱، این شغل در صدر لیست پردرآمدترین مشاغل در حوزه متخصص‌های امنیت سایبری با بازه متوسط دستمزد ۱۲۵۰۰۰ تا ۲۱۵۰۰۰ دلار (در زمان نگارش این متن بین ۳ میلیارد تا ۷ میلیارد ریال) قرار دارد. مدیران امنیت اطلاعات با شناسایی مناطقی که ممکن است سیستم‌های اطلاعاتی یک شرکت را آسیب‌پذیر کند، نقش کلیدی در جلوگیری از بلایای امنیتی ایفا می‌کنند.

این افراد حرفه‌ای هستند که وظیفه ارزیابی اقدامات امنیتی فعلی و کاهش حملات آینده علیه رایانه‌ها، شبکه‌ها و داده‌های شرکت را بر عهده‌دارند.

۱۱) استخدام شرکت‌های دولتی و خصوصی

یکی از راه‌های کسب درآمد از هک در خارج از کشور استخدام در شرکت‌های خصوصی و یا دولتی است به‌طوری‌که حداقل درآمد یک فرد که در شرکتی استخدام می‌شود ماهیانه ۵ هزار دلار (در زمان نگارش این متن ۱ میلیارد و ۵۰۰ میلیون ریال) است و همین‌طور که فرد و توانایی‌هایش پیشرفت می‌کنند حقوقش نیز بالاتر می‌رود که همان‌طور که مشخص است یک‌راه درآمدی خیلی خوب و عالی است.

فصل دوم

فصل ۲- بررسی آسیب‌پذیری‌های «OWASP Top10»

۲-۱- *Injection*

این آسیب‌پذیری در سال ۲۰۲۱ در جایگاه سوم لیست «*OWASP*» قرار دارد. ۹۴ درصد از ۲۷۴هزار اپلیکیشن آزمایش‌شده، در برابر نوعی از «*Injection*» (تزریق) با میزان بروز ۱۹ درصد و میانگین نرخ بروز ۳ درصد، آسیب‌پذیر هستند.

یک برنامه در برابر این حمله آسیب‌پذیر است درصورتی‌که:

۱) .

۲) .

۳) .

۴) .

مثال: یک اپلیکیشن از داده‌های نامعتبر در فراخوانی «*SQL*» آسیب‌پذیر زیر استفاده می‌کند:

.

به‌طور مشابه...

.

مثلاً:

.

این امر معنای هر دو پرس‌وجو را تغییر می‌دهد تا تمام داده‌ها از جدول حساب‌ها برگردد. حملات خطرناک‌تر می‌توانند داده‌ها را تغییر داده یا حذف کنند یا حتی رویه‌های ذخیره‌شده را فراخوانی کنند.

انواع «*Injection*» شامل موارد «*SQL injection*»، «*Command injection*»، «*Code injection*» و «*Server-side template injection*» می‌شود.

SQL injection

یک حمله «*SQL injection*» شامل درج یا تزریق یک «*SQL query*» از طریق داده‌های ورودی از مشتری به برنامه است. یک استخراج داده موفق «*SQL Injection*» می‌تواند داده‌های حساس را از پایگاه داده بخواند، داده‌های پایگاه داده را تغییر دهد که این تغییر می‌تواند یکی از عملیات‌های درج، به‌روزرسانی یا حذف باشد، مدیریت بر روی پایگاه داده (مانند خاموش کردن سیستم مدیریت پایگاه داده[1]) را اجرا کند، محتوای یک فایل معین موجود در سیستم مدیریت پایگاه داده را بازیابی کند و در برخی موارد دستوراتی را به سیستم‌عامل ارسال می‌کند.

.

.

.

شدت حملات «*SQL injection*» به مهارت و تخیل مهاجم و تا حدی کمتر، دفاع در اقدامات متقابل اساسی، مانند اتصالات با امتیاز پایین به سرور پایگاه داده و غیره محدود می‌شود. به‌طورکلی، «*SQL injection*» را یک سری حملات با شدت تأثیرگذاری بالا باید در نظر گرفت.

.

.

.

شکل ۲–۱ – نمونه‌ای از گزارش «SQL Injection»

به طورکلی پایگاه داده با وب اپلیکیشن در سه حالت« *Interaction*« برقرار میکند:

.

.

.

[1] Database Management Systems (DBMS)

شکل ۲–۲ - ابزار «SQLmap»

Command Injection

«*Command Injection*» حمله‌ای است که هدف آن اجرای دستورات دلخواه بر روی سیستم‌عامل میزبان از طریق یک برنامه آسیب‌پذیر است. این حملات زمانی امکان‌پذیر است که یک برنامه داده‌های ناامن ارائه‌شده توسط کاربر (فرم‌ها، کوکی‌ها، سرتیترهای «*HTTP*» و غیره) را به پوسته سیستم ارسال کند. تزریق فرمان به دستورات سیستم‌عامل ارائه‌شده توسط مهاجم معمولاً با امتیازات برنامه آسیب‌پذیر اجرا می‌شود. حملات تزریق فرمان عمدتاً به دلیل اعتبار سنجی ناکافی ورودی امکان‌پذیر است.

.

.

.

شکل ۲–۳ - سناریو «Command Injection»

مثال: قطعه کد «*PHP*» زیر در برابر حمله تزریق دستور آسیب‌پذیر است:

```
.
```

درخواست و پاسخ زیر نمونه‌ای از یک حمله موفق است:

درخواست:

```
.
```

پاسخ:

.

.

.

.

شکل ۲–۴ - نمونه‌ای از گزارش «Command Injection»

یکی از ابزارهای شناخته‌شده برای «*Command Injection*» ابزار «*Commix*» است. «*Commix*[۱]» یک ابزار رایگان و منبع باز است که در «*GitHub*» موجود است. این ابزار قدرتمند برای بهره‌برداری از آسیب‌پذیری‌های تزریق دستور در وب‌سایت‌ها و برنامه‌های کاربردی وب است. «*Commix*» به زبان پایتون نوشته‌شده است. ابزار «*Commix*» با ماژول‌های مختلف نصب‌شده در داخل آن عرضه می‌شود که به کاربر اجازه می‌دهد آسیب‌پذیری در برنامه هدف را پیدا کند. با استفاده از رشته‌های داده یا سرتیتر[۲] «*HTTP*» یا کوکی‌ها، همچنین روی پارامترهای احراز هویت، حمله به «*URL*» هدف را می‌توان ترکیب کرد. با استفاده از «*Commix*» کاربر می‌تواند دو نوع تزریق دستور را انجام دهد. اولی تکنیک تزریق فرمان مبتنی بر نتیجه[۳] و دومی تکنیک تزریق دستور کور [۴] است.

شکل ۲–۵ - ابزار «Commix»

Code Injection

«*Code Injection*» یک اصطلاح کلی برای انواع حملاتی است که شامل تزریق کد است و سپس توسط برنامه تفسیر یا اجرا می‌شود. این نوع حملات از مدیریت ضعیف داده‌های نامعتبر سوءاستفاده می‌کند و معمولاً به دلیل عدم اعتبارسنجی صحیح داده‌های ورودی/خروجی ممکن می‌شود. به‌عنوان مثال: کاراکترهای مجاز[۵]، فرمت داده و مقدار داده‌های مورد انتظار.

---

[۱] مخفف [comm]and و [i]njection و e[x]ploiter

[2] Header
[3] The result-based command injection technique
[4] The blind command injection technique
[5] Standard regular expressions classes or custom

.

.

.

زمانی که یک توسعه‌دهنده از تابع «*eval()*» در «*PHP*» استفاده می‌کند و داده‌های غیرقابل‌اعتمادی را که مهاجم می‌تواند تغییر دهد به آن ارسال می‌کند، تزریق کد ممکن است. مثال زیر یک روش خطرناک برای استفاده از تابع «*eval()*» را نشان می‌دهد:

.

ازآنجایی‌که اعتبار ورودی وجود ندارد، کد بالا در برابر حمله تزریق کد آسیب‌پذیر است. برای مثال:

.

در حین سوءاستفاده از اشکالاتی مانند مورد ذکرشده، یک مهاجم ممکن است بخواهد دستورات سیستم را اجرا کند. در این مورد، یک باگ تزریق کد نیز می‌تواند برای تزریق دستور استفاده شود، به‌عنوان مثال:

.

Server-side Template Injection

این حمله زمانی اتفاق می‌افتد که ورودی کاربر به‌طور ناامن در قالب سمت سرور[1] جاسازی شود و به کاربران امکان تزریق دستورالعمل‌های الگو را می‌دهد. با استفاده از دستورالعمل‌های الگوی مخرب، یک مهاجم ممکن است بتواند کد دلخواه را اجرا کند و کنترل کامل وب سرور را در دست بگیرد. شدت این موضوع بسته به نوع موتور قالبِ مورد استفاده متفاوت است. موتورهای قالب از میزان درصد بالایی بهره‌برداری[2]، تا تقریباً غیرممکن بودن بهره‌برداری متغیر هستند. هنگام تلاش برای توسعه یک استخراج داده باید از مراحل زیر استفاده کرد:

.

.

.

آسیب‌پذیری‌های تزریق قالب سمت سرور زمانی به وجود می‌آیند که ورودی کاربر به‌جای اینکه به‌عنوان داده ارسال شود، به قالب‌ها متصل می‌شود. قالب‌های ثابت[3] که به‌سادگی مکان‌هایی را فراهم می‌کنند که محتوای

[1] Server-side Template
[2] Exploit
[3] Static templates

پویا در آن‌ها تولید[1] می‌شود، معمولاً در برابر تزریق قالب سمت سرور آسیب‌پذیر نیستند. مثالی از این مبحث ایمیلی است که به هر کاربر با نام خودش خوش‌آمد می‌گوید، مانند کد زیر از یک الگوی *Twig*:

.

این کد نسبت به تزریق قالب سمت سرور آسیب‌پذیر نیست زیرا نام کاربر صرفاً به‌عنوان داده در قالب ارسال می‌شود.

بااین‌حال، ازآنجایی‌که قالب‌ها رشته‌های ساده هستند، توسعه‌دهندگان وب گاهی مستقیماً ورودی کاربر را قبل از تولید شدن به قالب‌ها متصل می‌کنند. در ادامه مثالی مشابه با مثال بالا آورده شده است، اما این بار، کاربران می‌توانند قسمت‌هایی از ایمیل را قبل از ارسال، سفارشی‌سازی[2] کنند. به‌عنوان‌مثال، آن‌ها ممکن است بتوانند نام مورداستفاده را انتخاب کنند:

.

در این مثال....

.

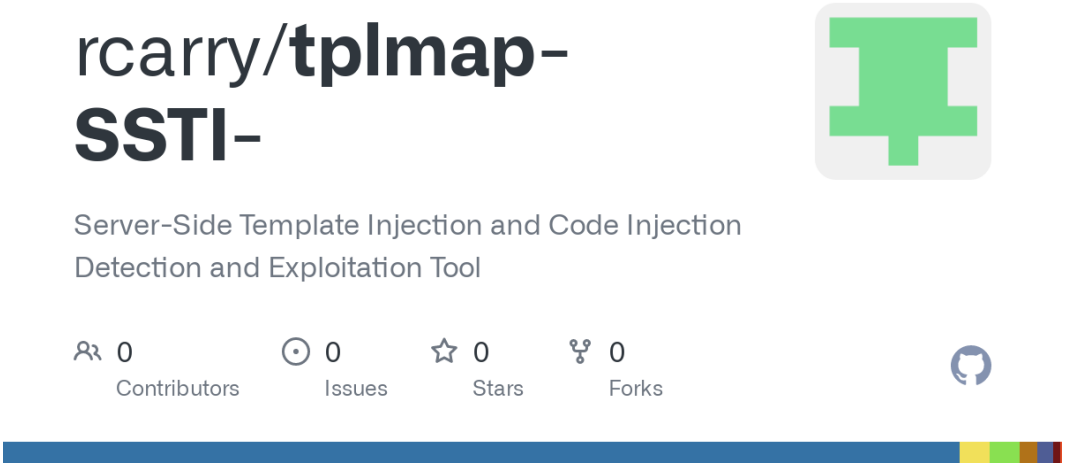
.

.

.

«*Tplmap*» یک ابزار امنیتی است که می‌تواند آسیب پذیری «*SSTI*» (تزریق قالب سمت سرور) را بررسی و بهره‌برداری کند همچنین برای استفاده به عنوان ابزار امنیتی تهاجمی در طول تست‌های نفوذ برنامه‌های وب توسعه داده شده‌است. ابزار «*Tplmap*» از بسیاری از موتورهای قالب مانند «*PHP*»، «*Ruby*»، «*Python*»، «*Jinja2*» و «*Tornado*» پشتیبانی می‌کند. این ابزار به زبان پایتون توسعه یافته و در پلتفرم «*GitHub*» نیز موجود است.

[1] Render
[2] Customize



شکل ۲–۶ – ابزار TPLmap

۲-۲- *Cross Site Script*

برای توضیح و بررسی این آسیب‌پذیری ابتدا نیاز است برخی مفاهیم توضیح داده شود.

۱. Document Object Model

«*Document Object Model*» (به‌اختصار «*DOM*») یک واسط برنامه‌نویسی کاربردی برای اسناد «*HTML*» و «*XML*» است. «*DOM*» در هنگام بارگذاری صفحه «*HTML*» ساخته می‌شود. (در حقیقت، مرورگر «*DOM*» را ایجاد می‌کند)

.

.

.

شکل ۲–۷ – نمونه‌ای از کد HTML و DOM مربوط به آن

منابع «*HTML*» از «*CRP*[1]» برای نمایش سبک صفحه[2] استفاده می‌کنند؛ ....

[1] Critical Rendering Path
[2] Style page

.

.

.

شکل ۲–۸ - روند تغییر «DOM» توسط جاوا اسکریپت

۲. Same Origin Policy

مفهوم دیگری که باید به توضیح آن پرداخته شود، «*Same Origin Policy*» (به‌اختصار «*SOP*») یا سیاست منبع مشترک است.

«*SOP*» یک فرآیند امنیتی است که نحوه بارگذاری کد یک منبع را محدود می‌کند تا منبع موردنظر با منابع مختلف دادوستد کند. یک «*Origin*» شامل طرح سایت[۱]، دامنه[۲] و شماره خروجی[۳] است. ریشه درخواست در هنگام اجرای کد تعیین می‌شود. [3]

سیاست مبدأ مشترک در زمانی اهمیت پیدا می‌کند که کد «*JavaScript*» درخواست دسترسی به «*DOM*» یک منبع دیگر را دارد. در این مرحله، «*SOP*» به‌وسیله تشخیص منبع درخواست، تصمیم به محدود کردن یا نکردن درخواست می‌گیرد. برای مثال «*URL*» زیر را در نظر بگیرید:

.

در مثال بالا ....

جدول ۲–۱ - بررسی سیاست دسترسی به «origins» دیگر

دسترسی اجازه داده خواهد شد؟	درخواست دسترسی به
بله؛ زیرا...	
بله؛ زیرا...	
خیر؛ زیرا...	
خیر؛ زیرا...	
خیر؛ زیرا...	
خیر؛ زیرا...	

هنگامی‌که یک مرورگر یک درخواست «*HTTP*» را از یک منبع به منبع دیگر ارسال می‌کند، ....

[1] URI scheme
[2] Domain
[3] Port

حال برای درک بهتر این مفاهیم سناریو[1] که در شکل ۱۰-۲ نمایش داده‌شده، مطرح می‌شود:

.

.

.

شکل ۲–۹ - سناریو «Same Origin Policy»

۱) کاربر وارد وب‌سایتی (به‌طور مثال سیستم نامه‌رسان خود *Website A*) می‌شود

۲) کاربر در همان مرورگر یک زبانه جدید باز می‌کند و وارد سایت دیگری می‌شود (*Website B*)

۳) سایت جدیدی که کاربر وارد آن شده .....

۴) ....

۵) ....

پس می‌توان نتیجه گرفت که ....

۳. Cross-Origin Resource Sharing

در این روش سرور به کد جاوا اسکریپت دسترسی به برخی از منابع ریشه‌ها را می‌دهد. برخی از سرتیترها توسط سرور تعیین می‌شوند. با توجه به سرتیترهای پاسخ، سرور تصمیم به اجازه دادن یا رد درخواست دسترسی به «*DOM*» را می‌دهد.

به بیانی دیگر اشتراک‌گذاری منابع متقاطع (*CORS*) یک سازوکار[2] مرورگر است که دسترسی کنترل‌شده به منابع واقع در خارج از یک دامنه مشخص را امکان‌پذیر می‌سازد. آن را گسترش می‌دهد و انعطاف‌پذیری را به سیاست مبدأ یکسان (*SOP*) اضافه می‌کند. بااین‌حال، اگر خط‌مشی «*CORS*» یک وب‌سایت به‌درستی پیکربندی و اجرانشده باشد، پتانسیل حملات بین دامنه‌ای را نیز فراهم می‌کند. [4]

.

.

.

شکل ۲–۱۰ - Cross-Origin Resource Sharing

بسیاری از وب‌سایت‌های مدرن از «*CORS*» برای اجازه دسترسی از زیر دامنه‌ها و اشخاص ثالث قابل‌اعتماد استفاده می‌کنند. اجرای «*CORS*» آن‌ها ممکن است حاوی اشتباهات باشد یا برای اطمینان از اینکه همه‌چیز کار می‌کند بسیار ملایم باشد و این می‌تواند منجر به آسیب‌پذیری‌های قابل بهره‌برداری شود.

[1] Scenario
[2] Mechanism

٤. انواع درخواست مرورگر

اکنون این موضوع مطرح می‌شود که مرورگر همواره می‌تواند دو نوع درخواست «*Simple*» یا «*Preflight*» ارسال کند.

۱) درخواست «*Simple*»

درخواست‌های ....

نمونه‌ای از درخواست «*simple*» در زیر نشان داده‌شده است:

```
.
```

.

.

.

شکل ۲–۱۱ – نمونه‌ای از درخواست «simple» در جاوا اسکریپت

۲) درخواست «*Preflight*»

مرورگر یک درخواست «*HTTP*» توسط «*option*» ارسال می‌کند، پاسخ را برای سطح دسترسی «*pars*» می‌کند، در صورت اجازه سرور به «*CORS*»، درخواست ارسال می‌شود.

.

.

.

مثال:

```
.
```

برای قطعه کد بالا می‌بایست ....

تعریف «*Cross-site Script*»

اکنون بعد از توضیح مفاهیم مطرح‌شده که برای تعریف و درک بهتر این آسیب‌پذیری نیاز بود، به بررسی «*Cross-site Script(XSS)*» پرداخته می‌شود.

«*Cross-site Script*» (همچنین به‌عنوان «*XSS*» شناخته می‌شود) یک آسیب‌پذیری امنیتی وب است که به مهاجم اجازه می‌دهد تا تعاملات کاربران با یک برنامه آسیب‌پذیر را به خطر بیندازد. این آسیب‌پذیری به مهاجم اجازه می‌دهد تا سیاست مبدأ یکسانی را که برای جداسازی وب‌سایت‌های مختلف از یکدیگر طراحی‌شده است، دور بزند. آسیب‌پذیری‌های اسکریپت بین سایتی معمولاً به مهاجم این امکان را می‌دهند که خود را به‌عنوان یک کاربر قربانی درآورد، هر اقدامی را که کاربر قادر به انجام آن است انجام دهد و به هر یک از داده‌های کاربر دسترسی پیدا کند. اگر کاربر قربانی دسترسی ممتازی به برنامه داشته باشد، ممکن است مهاجم بتواند کنترل کاملی بر تمام عملکردها و داده‌های برنامه داشته باشد. [6]

«*Cross site Script*» با دست‌کاری یک وب‌سایت آسیب‌پذیر عمل می‌کند تا جاوا اسکریپت مخرب را به کاربران برگرداند. هنگامی‌که کد مخرب در داخل مرورگر قربانی اجرا می‌شود، مهاجم می‌تواند به‌طور کامل تعامل آن‌ها با برنامه را به خطر بیندازد.

اهداف این آسیب‌پذیری:

.

.

.

سه نوع اصلی از حملات «*XSS*» وجود دارد. که شامل موارد زیر هستند:

۱. ....

۲. ....

۳. ....

حال به توضیح دقیق‌تری از انواع «*XSS*» پرداخته می‌شود:

۱. *Reflected cross-site scripting*

«*Reflected XSS*» ساده‌ترین نوع اسکریپت‌نویسی بین‌سایتی است.

.

.

.

در ادامه یک مثال ساده از آسیب پذیری «*XSS*» اورده شده است:

.

برنامه هیچ پردازش دیگری روی داده‌ها را انجام نمی‌دهد، بنابراین یک مهاجم می‌تواند به راحتی حمله‌ای مانند زیر را طراحی کند:

.

اگر کاربر ....

۲. *Stored cross-site scripting*

«*XSS*» ذخیره شده (همچنین به عنوان «*XSS*» دائمی یا مرتبه دوم شناخته می‌شود) زمانی ایجاد می‌شود که یک برنامه داده‌ها را از یک منبع نامعتبر دریافت می‌کند و آن داده‌ها شامل پاسخ‌های «*HTTP*» بعدی خود به روشی ناامن هستند.

داده‌های مورد نظر ممکن است از طریق درخواست‌های «*HTTP*» به برنامه ارسال شوند. به عنوان مثال، نظرات در مورد یک پست وبلاگ، نام مستعار کاربر در اتاق گفتگو، یا جزئیات تماس در مورد سفارش مشتری. در موارد دیگر، داده‌ها ممکن است از منابع غیرقابل اعتماد دیگری به دست آیند.

در ادامه یک مثال ساده از آسیب‌پذیری «*XSS*» ذخیره‌شده بیان شده‌است. فرض می‌شود یک برنامه پیام رسان به کاربران امکان می‌دهد پیام‌هایی را ارسال کنند که برای سایر کاربران نمایش داده می‌شود:

.

برنامه هیچ پردازش دیگری از داده‌ها انجام نمی‌دهد، بنابراین یک مهاجم می‌تواند به راحتی پیامی ارسال کند که به سایر کاربران حمله می‌کند:

.

۳. *DOM-based cross-site scripting*

«*XSS*» مبتنی بر «*DOM*» (همچنین به عنوان «*DOM XSS*» شناخته می‌شود) زمانی ایجاد می‌شود که یک برنامه حاوی جاوا اسکریپت سمت سرویس گیرنده باشد که معمولاً با نوشتن داده‌ها در «*DOM*» آنها را از یک منبع نامعتبر به روشی ناامن پردازش می‌کند.

در مثال زیر، یک برنامه از جاوا اسکریپت برای خواندن مقدار از یک فیلد ورودی و نوشتن آن مقدار در یک عنصر[1] در «*HTML*» استفاده می‌کند:

.

اگر مهاجم بتواند مقدار فیلد ورودی را کنترل کند، می‌تواند به راحتی یک مقدار مخرب بسازد که باعث می‌شود اسکریپت خودش اجرا شود:

.

[1] Element

.

.

.

مهاجمی که بتواند آسیب‌پذیری اسکریپت بین سایتی را پیدا کند معمولاً قادر به انجام اعمال زیر است:

۱. .

۲. .

۳. .

۴. .

۵. .

۶. .

حال یک سناریو در ارتباط با این آسیب‌پذیری مطرح می‌شود:

.

.

.

شکل ۲–۱۲ – سناریو «Cross-site Script»

۱) مهاجم ....

۲) .

۳) .

۴) .

۵) .

۶) .

روش‌های جلوگیری از حملات «Cross-site Script»

به طور کلی، پیشگیری موثر از آسیب‌پذیری‌های «*XSS*» شامل ترکیبی از اقدامات زیر است:

.

.

.

۲-۳- *Cross-Site Request Forgery*

«*CSRF*» یک نوع حمله است که با استفاده از درخواست «*HTTP*» کاربر را مجبور می‌کند تا اقدامات ناخواسته‌ای را روی یک برنامه وب که در حال حاضر در آن احراز هویت شده است، انجام دهد.

این درخواست «*HTTP*» را می‌توان به روش‌های زیر ارسال کرد:

- .
- .
- .
- .

در یک حمله موفقیت‌آمیز «*CSRF*»، مهاجم باعث می‌شود....[5]

برای اینکه حمله «*CSRF*» امکان پذیر باشد، سه شرط کلیدی باید وجود داشته باشد:

.

.

.

به عنوان مثال، یک برنامه دارای تابعی است که به کاربر اجازه می‌دهد آدرس ایمیل حساب خود را تغییر دهد. هنگامی که یک کاربر این عمل را انجام می‌دهد، یک درخواست «*HTTP*» مانند زیر ارائه می‌دهد:

.

این درخواست شرایط لازم برای «*CSRF*» را برآورده می‌کند زیرا:

- .
- .
- .

با وجود این شرایط، مهاجم می‌تواند یک صفحه وب حاوی «*HTML*» زیر بسازد:

.

اگر کاربر قربانی از صفحه وب مهاجم بازدید کند، موارد زیر رخ می‌دهد:

- .
- .
- .

اکنون سناریو زیر برای این آسیب پذیری مطرح می‌شود:

.

.

.

شکل ۲–۱۳ - سناریو «Cross Site Request Forgery»

۱) .

۲) .

۳) .

۴) .

۵) .

روش‌های جلوگیری از حملات «CSRF»:

۱) .

۲) .

۳) .

۲-٤- *Broken Authentication*

«*Authentication*» فرآیند تأیید هویت یک کاربر یا مشتری معین است. به عبارت دیگر، این شامل اطمینان از این است که آنها واقعاً همان چیزی هستند که ادعا می‌کنند هستند. حداقل تا حدی، وب‌سایت‌ها در معرض دید هر کسی که به اینترنت متصل است، قرار می‌گیرد. بنابراین، رویکردهای احراز هویت قوی یک جنبه جدایی ناپذیر از امنیت وب است.

تفاوت میان«*authentication*» و «*authorization*» در این است که ....

به طور کلی، بیشتر آسیب‌پذیری‌ها در رویکردهای احراز هویت به یکی از دو دلیل زیر ایجاد می‌شوند:

۱) .

۲) .

تأثیر آسیب‌پذیری‌های احراز هویت می‌تواند بسیار شدید باشد. هنگامی که یک مهاجم....

سیستم احراز هویت یک وب سایت معمولاً از چندین رویکرد مجزا تشکیل می‌شود که در آن آسیب‌پذیری ممکن است رخ دهد. برخی از آسیب‌پذیری‌ها به طور گسترده در همه این زمینه‌ها قابل اجرا هستند، در حالی که برخی دیگر به عملکرد ارائه‌شده بستگی دارند. بطور کلی یک مدل احراز هویت شامل بخش‌های زیر است:

- .
- .
- .
- .

نکته قابل توجه این است که وقتی کاربر اطلاعات خودش را وارد می‌کند برنامه این اطلاعات را با اطلاعات قبلی که در پایگاه داده ذخیره شده است مقایسه می‌کند و اگر درست بود کوکی احراز هویت بروزرسانی می‌شود. این بروزرسانی می‌تواند در کوکی یا «*session*» یا هر دوی ایشان ذخیره‌شود.

برای مثال سناریو ورود یک کاربر به سایت اصلی و زیردامنه‌ای از آن سایت در ادامه مطرح شده است:

.

.

.

شکل ۲–۱۴ – سناریو مربوط به تعریف کوکی احراز شده

۱) .

۲) .

۳) .

۴) .

در نتیجه میتوان گفت عاملی که وضعیت کاربر را در سایت به صورت احراز هویت شده یا همان «*Logging*» شده نگه‌می‌دارد، کوکی احرازهویت است.

تفاوت «*Cookie*» و «*Session*»

.

.

.

مزیت‌های استفاده از توکن شامل موارد زیر است:

۱) .

توکن‌ها ....

۲) Security

توکن، در هر درخواست ارسال می‌شود و ....

۳) Extensibility (Friend of A Friend and Permissions)

توکن‌ها به ما این امکان را می‌دهند که ....

۴) Multiple Platforms and Domains

هنگامی که برنامه و سرویسی گسترش می‌یابد، ....

JSON Web Token

«*JWT*» توکن‌های «*JSON*» با امضای رمزنگاری شده هستند که برای به اشتراک گذاشتن «*claims*» بین سیستم‌ها طراحی شده‌اند. آن‌ها اغلب به عنوان تأیید اعتبار یا نشانه‌های «*session*»، به ویژه در «*API*»های «*REST*» استفاده می‌شوند. «*JWT*»ها، هم در نحوه پیاده‌سازی آن‌ها در برنامه‌ها و هم در کتابخانه‌های زیربنایی، منبع مشترکی از آسیب‌پذیری‌ها هستند. از آنجایی که از آن‌ها برای احراز هویت استفاده می‌شود، یک آسیب‌پذیری به راحتی می‌تواند منجر به به خطر افتادن کامل برنامه شود.

«*JWT*»ها از سه جزء تشکیل شده‌اند:

- *The header*
- *The payload (or body)*
- *The signature*

هر جزء از این نوع توکن با «*Base64*» کدگذاری شده‌است و با نقطه (.) از هم جدا می‌شوند. حال به بررسی دقیق‌تری از هر قسمت نام برده‌شده، پرداخته می‌شود.

- *The header*

.

.

.

یک نمونه «*header*» رمزگشایی شده در زیر نشان داده شده‌است:

.

سه نوع الگوریتم اصلی برای محاسبه امضا وجود دارد:

جدول ۲–۲ - انواع الگوریتم برای محاسبه «Signatures»

توضیحات	الگوریتم
.	.
.	.
.	.

همچنین طیف گسترده‌ای از الگوریتم‌های دیگر وجود دارد که ممکن است برای توکن‌های رمزگذاری شده «*JWT*» استفاده شود، اگرچه این الگوریتم‌ها کمتر رایج هستند.

- *The payload*

«*Payload*» حاوی داده‌های واقعی است.

.

.

.

نمونه‌ای از «*Payload*» در زیر نشان داده‌شده‌است:

.

.

.

.

ادعاهایی که اجباری نیستند و نام آنها محفوظ است، در ادامه معرفی شده‌اند:

.

.

.

شکل ۲–۱۵ - تعدادی از «Claim» های مربوط به «JWT»

.

.

.

این ادعاها در «*RFC 5719*[1]» تعریف شده‌اند که خلاصه‌ای از آنها در جدول زیر آمده است:

جدول ۲–۳ – خلاصه‌ای از توضیحات«Claim» مربوط به نمونه«JWT»

Claim	**نام کامل**	**توضیحات**
.	.	.
.	.	.
.	.	.
.	.	.

- *The signature*

امضا با استفاده از الگوریتم تعریف‌شده در ....

.

.

.

شکل ۲–۱۶ – اجزای «JWT»

«*JWT tool*» ابزاری است که برای نفوذگرها نوشته شده‌است، که می‌تواند قدرت توکن‌های در حال استفاده و حساسیت آنها را در برابر حملات شناخته‌شده بررسی کند. در این ابزار طیف وسیعی از گزینه‌های دستکاری، امضا و تأیید برای کمک به عمیق‌تر کردن نقاط ضعف احتمالی موجود در برخی از کتابخانه‌های «*JWT*» در دسترس است. همچنین ممکن است برای توسعه‌دهندگانی که از «*JWT*» در پروژه‌ها استفاده می‌کنند مفید باشد، زیرا می‌توانند هنگام استفاده از توکن‌های جعلی، پایداری و آسیب‌پذیری‌های شناخته شده را آزمایش کنند.

این ابزار به صورت بومی در پایتون ۳ (نسخه ۳.۶+) با استفاده از کتابخانه‌های رایج نوشته شده‌است، با این حال توابع رمزنگاری مختلف نیاز به نصب چند کتابخانه رایج پایتون دارند.

[1] برای مشاهده اطلاعات بیشتر به سایت https://tools.ietf.org/html/rfc7519#section-4.1 مراجعه کنید.

شکل ۲–۱۷ - ابزار «JWT»

Open Authorization

«*OAuth*» یک فریم‌ورک «*Authorization*» است که به وب‌سایت‌ها و برنامه‌های کاربردی وب امکان می‌دهد تا دسترسی محدودی به حساب کاربر در برنامه دیگری درخواست کنند. مهمتر از همه، «*OAuth*» به کاربر اجازه می‌دهد تا این دسترسی را بدون افشای اعتبار ورود خود به برنامه درخواست‌کننده اعطا کند. به این معنا که کاربران می‌توانند به جای اینکه کنترل کامل حساب خود را به شخص ثالث بسپارند، داده‌هایی را که می‌خواهند به اشتراک بگذارند تنظیم کنند.

فرآیند اولیه «*OAuth*» به طور گسترده برای ادغام عملکرد «*third-party*» استفاده می‌شود که نیاز به دسترسی به داده‌های خاصی از حساب کاربر دارد. به عنوان مثال، یک برنامه ممکن است از «*OAuth*» برای درخواست دسترسی به لیست مخاطبین برنامه نامه‌رسان الکترونیکی فردی استفاده کند تا بتواند افراد را برای ارتباط با آنها پیشنهاد دهد. با این حال، از همین رویکرد برای ارائه خدمات احراز هویت شخص‌ثالث نیز استفاده می‌شود و به کاربران اجازه می‌دهد با حسابی که در وب‌سایت دیگری دارند وارد شوند.

.

.

.

ساده‌ترین سناریو این فرآیند به شکل زیر است:

.

.

.

شکل ۲–۱۸ – سناریو فرآیند «OAuth»

۱) .

۲) .

۳) .

۴) .

۵) .

۶) .

هر «*URL*» از چهار بخش متفاوت تشکیل شده‌است که دسترسی به آن‌ها نیز بصورت مجزا، نیازمند مجوز دسترسی می‌باشد. برای مثال «*URL*» زیر و بخش‌های متفاوت آن آورده شده‌است:

.

- .
- .
- .
- .

برای مثال، ....

.

.

.

برای ادامه دادن این مبحث ابتدا باید «*SSO*» را تعریف کرد.

Single Sign-On

«*Single Sign-on (SSO)*» یک روش شناسایی است که به کاربران امکان می‌دهد با یک مجموعه از «*credentials*» به چندین برنامه و وب‌سایت وارد شوند. «*SSO*» که فرآیند احراز هویت را برای کاربران ساده می‌کند، زمانی اتفاق می‌افتد که کاربر به برنامه‌ای وارد می‌شود و بدون در نظر گرفتن دامنه، پلتفرم یا فناوری که استفاده می‌کند، به‌طور خودکار به برنامه‌های متصل دیگر نیز وارد می‌شود. این امر مدیریت چندین نام کاربری و رمز عبور را در حساب‌ها و سرویس‌های مختلف آسان می‌کند. مثالی از این فرآیند: زمانی که کاربر وارد «*Google*» می‌شود و اعتبار آنها به طور خودکار در سرویس‌های پیوندی مانند «*Gmail*» و «*YouTube*» احراز هویت می‌شود، بدون اینکه نیازی به ورود جداگانه به هر یک به صورت جداگانه باشد. [14]

.

.

.

شکل ۲–۱۹ - سناریو «SSO»

سرویس‌های «*SSO*» اطلاعات یا هویت کاربر را ذخیره نمی‌کنند. در عوض، آنها معمولاً با بررسی و تطبیق اعتبار ورود کاربر با اطلاعات ذخیره‌شده در سرویس مدیریت هویت یا پایگاه‌داده کار می‌کنند.

«*Single sign-on*» از مراحل زیر برای اطمینان از هدایت اعتبار کاربر از یک ارائه‌دهنده خدمات (*SP*[1]) به یک ارائه دهنده هویت (*IdP*[2]) استفاده می‌کند:

۱. .
۲. .
۳. .
۴. .
۵. .

وقتی کاربر بعد از گذراندن مراحل بالا توانست وارد یکی از زیر دامنه‌ها یا وب‌سایت‌های دیگر شود....

.

.

.

شکل ۲–۲۰ - سناریو «SSO» با «JASONP» یا «AjaxCall»

۱) فرآنید اول
   - ○ .
   - ○ .
   - ○ .
   - ○ .
۲) فرآیند دوم
   - ○ .

[1] Service Provider
[2] Identity Provider

- .

۳) فرآیند سوم

- .
- .
- .
- .

۴) فرآیند چهارم

- .
- .
- .
- .
- .
- .
- .

آسیب‌پذیری‌های احراز هویت «*OAuth*» تا حدی به این دلیل به وجود می‌آیند که ....

باتوجه به توضیحات داده‌شده مراحلی که یک مهاجم طی می‌کند تا با استفاده از «*JSONP*» به توکن احراز هویت کاربر دسترسی پیدا کند در شکل ۲۳-۲ نشان داده شده‌است:

.

.

.

شکل ۲–۲۱ - سناریو حمله «JASONP»

۱) .

۲) .

۳) .

۴) .

۵) .

برای جلوگیری از آسیب‌پذیری‌های احراز هویت «*OAuth*»، ....

۲-۵- *Sensitive Data Exposure*

این آسیب‌پذیری در بروزرسانی جدید «*OWASP top10*» قرار ندارد؛ با این وجود یکی از مهم‌ترین آسیب‌پذیری‌های موجود در اپلیکیشن‌های تحت وب است.

داده‌های مهمی همواره در پایگاه‌داده اپلیکیشن‌ها ذخیره می‌شود؛ بطور مثال این داده‌ها شامل: «*Passwords*»، «*Session tokens*»، «*Credit cards*»، «*Private health data*» و « *Private tokens*» هستند. این آسیب‌پذیری زمانی رخ می‌دهد که یک برنامه به طور تصادفی داده‌های حساس را در معرض دید قرار دهد. این فرآیند با نقص اطلاعات، که در آن مهاجم به اطلاعات دسترسی پیدا کرده و آنها را سرقت می‌کند، متفاوت است. قرار گرفتن در معرض داده‌های حساس معمولا زمانی اتفاق می‌افتد که محافظت کافی از اطلاعات موجود در پایگاه داده صورت نگیرد.

مواردی که باعث این آسیب‌پذیری می‌شوند:

1) .

2) .

3) .

4) .

5) .

6) .

آسیب‌پذیری قرار گرفتن در معرض داده‌ها، به نحوه مدیریت اطلاعات بستگی دارد.

.

.

.

سناریو زیر برای درک بهتر این مفهوم مطرح می‌شود:

.

.

.

.

.

.

شکل ۲–۲۲ – سناریو «Sensitive Data Exposure»

بنابراین ممکن است مهاجم به داده‌های خصوصی کاربر دسترسی داشته‌باشد یا آنها را تغییر دهد. مهاجم حتی می‌تواند داده‌های منتقل شده را تغییر دهد، به عنوان مثال، گیرنده انتقال پول را حساب کاربری خود قرار دهد.

همانطور که مطرح شد یکی از راه‌های دسترسی به این آسیب‌پذیری چک کردن فایل‌های مشخصی با انواع تایپ‌های ذکر شده است. که برای این کار بهترین روش «*fuzz*» کردن فایل‌ها می‌باشد.

تعریف «Fuzzing»

«*Fuzz Testing*» یا «*Fuzzing*» یک تکنیک و نوعی نرم‌افزار «*Black Box*» است که اساساً شامل یافتن اشکالات پیاده‌سازی با استفاده از تزریق داده‌های نادرست یا نیمه‌درست به روش خودکار است.

بطور مثال یک عدد صحیح در یک برنامه در نظر گرفته می‌شود که نتیجه انتخاب کاربر بین ۳ سوال را ذخیره می‌کند. هنگامی که کاربر یکی را انتخاب می‌کند، انتخاب ۰، ۱ یا ۲ خواهد بود. که سه مورد عملی را ایجاد می‌کند. اما اگر عدد ۳ یا ۲۵۵ ارسال شود هیچ تغییری رُخ نمی‌دهد، زیرا اعداد صحیح یک متغیر به اندازه ثابت ذخیره می‌شوند. در نتیجه می‌توان دنباله‌ای از اعداد را به عنوان ورودی امتحان کرد تا به جواب درست رسید.

«*Fuzzing*» هنر یافتن خودکار اشکال است و نقش آن یافتن خطاهای پیاده‌سازی نرم‌افزار و شناسایی آن‌ها در صورت امکان است.

«*fuzzer*» برنامه‌ای است که به طور خودکار داده‌های نیمه‌تصادفی را به یک برنامه یا پشته تزریق می‌کند و ایرادات را شناسایی می‌کند.

بخش تولید داده از ژنراتورها ساخته شده‌است و شناسایی آسیب‌پذیری به ابزارهای اشکال‌زدایی متکی است. ژنراتورها معمولاً از ترکیبی از بردارهای فازی ثابت (مقادیر خطرناک شناخته‌شده) یا داده‌های کاملاً تصادفی استفاده می‌کنند. فازرهای نسل جدید از الگوریتم‌های ژنتیک برای پیوند داده‌های تزریق شده و تاثیر مشاهده شده استفاده می‌کنند. چنین ابزارهایی هنوز عمومی نیستند.

یک «*fuzzer*» ترکیبی از حملات را در موارد زیر امتحان می‌کند:

- اعداد («*signed*» / بدون علامت / «*float*» / ...)

- کاراکترها («*URL*»، ورودی های خط فرمان[1])
- ابرداده[2]: متن ورودی کاربر (تگ «*id3*»)
- دنباله‌های باینری خالص

یک رویکرد رایج برای فازبندی، تعریف فهرست‌هایی از «مقادیر خطرناک شناخته‌شده» (بردارهای فاز) برای هر نوع، و تزریق آن‌ها یا ترکیب‌های مجدد است.

- برای اعداد صحیح: صفر، اعداد منفی یا بسیار بزرگ
- برای کاراکترها: «*escaped*»، دستورالعمل‌های قابل تفسیر، «*instructions*» (مثلاً برای درخواست‌های «*SQL*»، نقل قول‌ها [3]/ دستورات[4]...)
- برای باینری: تصادفی

پروتکل‌ها و فرمت‌های فایل به هنجارهایی دلالت دارند که گاهی مبهم، بسیار پیچیده یا بد پیاده‌سازی می‌شوند، به همین دلیل است که توسعه‌دهندگان گاهی اوقات در فرآیند پیاده‌سازی اختلال ایجاد می‌کنند. (به دلیل محدودیت‌های زمان یا هزینه) در نتیجه می‌توان رویکرد مخالف را در پیش گرفت. یعنی یک هنجار در نظر گرفته شود، به همه ویژگی‌ها و محدودیت‌های اجباری نگاه شود و همه آنها امتحان شود. مانند: مقادیر ممنوع یا رزرو شده، پارامترهای مرتبط و اندازه فیلدها. این فرآیند یک فاز مبتنی بر تست انطباق خواهد بود.

یکی از ابزارهای «*fuzzer*» که استفاده‌ی زیادی برای مهاجمین دارد «*Wfuzz*» است.

«*Wfuzz*» برای تسهیل کار در ارزیابی برنامه‌های کاربردی وب ایجاد شده است و بر یک مفهوم ساده استوار است: هر ارجاعی به کلمه کلیدی «*FUZZ*» را با مقدار «*payload*» مشخص جایگزین می‌کند.

یک «*payload*» در «*Wfuzz*» منبع داده‌است. این مفهوم ساده اجازه می‌دهد تا هر ورودی در هر زمینه‌ای از یک درخواست «*HTTP*» تزریق شود و امکان انجام حملات امنیتی پیچیده وب در اجزای مختلف برنامه وب مانند پارامترها، احراز هویت، فرم‌ها، پوشه‌ها یا فایل‌ها، هدرها و غیره را فراهم می‌کند.

«*Wfuzz*» چیزی بیش از یک اسکنر محتوای وب است، «*Wfuzz*» می‌تواند به فرد کمک کند تا با یافتن و بهره‌برداری از آسیب‌پذیری‌های برنامه‌های وب، برنامه‌های وب خود را ایمن کند. اسکنر آسیب‌پذیری برنامه وب «*Wfuzz*» توسط افزونه‌ها پشتیبانی می‌شود.

«*Wfuzz*» یک رابط زبان ساده را در معرض درخواست‌ها یا پاسخ‌های «*HTTP*» قبلی که با استفاده از «*Wfuzz*» یا ابزارهای دیگر مانند «*Burp*» انجام می‌شد، نشان می‌دهد. این به فرد امکان می‌دهد تا تست‌های

[1] Command-line inputs
[2] MetaData
[3] Quotes
[4] Commands

دستی و نیمه خودکار را با زمینه کامل و درک اقدامات خود انجام دهد، بدون اینکه به یک اسکنر برنامه کاربردی وب زیربنای پیاده سازی تکیه کند.

این ابزار برای تسهیل کار در ارزیابی برنامه‌های کاربردی وب ایجاد شده‌است، و ابزاری است که توسط نفوذگران برای نفوذ استفاده زیادی می‌شود.

شکل ۲–۲۳ - ابزار «Wfuzz»

روش‌های جلوگیری از آسیب‌پذیری «Sensitive Data Exposure»

در این بخش مراحل مختلفی برای جلوگیری از قرار گرفتن در معرض آسیب‌پذیری « *Sensitive Data Exposure*» مورد بحث قرار داده می‌شود.

.

.

.

۲-٦- *Server-Side Request Forgery*

جعل درخواست سمت سرور (همچنین به عنوان «*SSRF*» شناخته می شود) یک آسیب‌پذیری امنیتی وب است که به مهاجم اجازه می‌دهد تا برنامه سمت سرور را وادار کند تا درخواست‌هایی را به یک مکان ناخواسته ارسال کند و در جواب این درخواست، اطلاعات حساسی از سرور دریافت می‌شود. به وسیله این آسیب‌پذیری، می‌توان به اطلاعات «*Cloud Server Metadata*» دسترسی پیدا کرد.

در هنگام ورود به فضای وب، «*Database HTTP Interface*» به صورت پیش‌فرض در خروجی ۹۲۰۰ به صورت محلی بارگذاری می‌شود اما از طریق وب قابل مشاهده نیست؛ در نتیجه این فرآیند چنانچه سرور مجبور به ارسال درخواست به این خروجی شود، آسیب پذیری «*SSRF*» رخ می‌دهد.

این آسیب‌پذیری منجر به خواندن یک سند یا تحلیل «*IP*» یا خروجی سرور[1] می‌شود.

.

.

.

شکل ۲–۲۴ – نمونه‌ای از گزارش یک «SSRF»

در یک حمله معمولی «*SSRF*»، مهاجم ممکن است باعث شود سرور به سرویس‌های داخلی در زیرساخت سازمان متصل شود. در موارد دیگر، ممکن است بتواند سرور را مجبور به اتصال به سیستم‌های خارجی دلخواه کند و به طور بالقوه داده‌های حساس مانند اعتبارنامه‌های مجوز [2] را نشت دهند.

یک حمله موفقیت آمیز «*SSRF*» اغلب می‌تواند منجر به....

.

.

.

شکل ۲–۲۵ – سناریوی حمله «SSRF»

۱) .

۲) .

۳) .

۴) .

۵) .

انواع حملات «SSRF»

۱) *SSRF attacks against the server itself*

۲) *SSRF attacks against other back-end systems*

که به توضیح بیشتر آنها خواهیم پرداخت.

[1] Port
[2] Authorization Credentials

۱- حملات «*SSRF*» به خود سرور

در یک حمله «*SSRF*» علیه خود سرور...

.

.

.

بنابراین هنگامی که یک کاربر وضعیت موجودی یک کالا را مشاهده می‌کند، مرورگر او درخواستی مانند زیر را ارائه می‌دهد:

.

درخواست یاد شده باعث می‌شود ....

به طور مثال:

.

.

.

.

۲- حملات «*SSRF*» علیه سایر سیستم‌های «*Back-end*»

نوع دیگری از رابطه اعتماد که اغلب با جعل درخواست سمت سرور ایجاد می‌شود، جایی است که سرور برنامه قادر به تعامل با سایر سیستم‌های پشتیبان است که مستقیماً توسط کاربران قابل دسترسی نیستند. این سیستم‌ها اغلب دارای آدرس‌های «*IP*» خصوصی غیر قابل مسیریابی هستند. از آن‌جایی که سیستم‌های پشت صحنه معمولاً توسط توپولوژی شبکه محافظت می‌شوند، اغلب وضعیت امنیتی ضعیف‌تری دارند. در بسیاری از موارد، سیستم‌های پشت صحنه داخلی دارای قابلیت‌های حساسی هستند که می‌توانند بدون احراز هویت توسط هر کسی که قادر به تعامل با سیستم‌ها باشد، به آن دسترسی داشته‌باشد.

در اینجا، یک مهاجم می‌تواند با ارسال درخواست زیر از آسیب‌پذیری «*SSRF*» برای دسترسی به رابط اداری سوء استفاده کند:

.

روش‌های رایج دور زدن [1] دفاع در مقابل «SSRF»

مشاهده برنامه‌های کاربردی حاوی رفتار «*SSRF*» همراه با دفاع‌هایی با هدف جلوگیری از بهره‌برداری مخرب بسیار رایج است اما اغلب، این دفاع‌ها را می‌توان با راه‌حل‌هایی که در ادامه توضیح داده‌می‌شود دور زد.

۱) .

۲) .

.

برای مثال، فرض شود برنامه حاوی یک آسیب‌پذیری «*open redirection*» است که در آن «*URL*» زیر وجود دارد:

.

یک تغییر مسیر به:

.

می‌تواند از آسیب‌پذیری «*open redirection*» برای دور زدن فیلتر «*URL*» استفاده کرد و از آسیب‌پذیری «*SSRF*» به صورت زیر استفاده کرد:

.

این بهره‌برداری «*SSRF*» به این دلیل کار می‌کند که ....

روش‌های جلوگیری از حملات «SSRF»

در ادامه چند راهکار وجود دارد که برای کاهش حملات «*SSRF*» موثر است. طبق معمول، این روش‌ها ۱۰۰ درصد در برابر این آسیب پذیری از برنامه محافظت نمی‌کنند، اما شانس دفاع را بیشتر می‌کنند.

۱) .

۲) .

۳) .

.

۴) .

۵) .

۶) .

[1] Circumvent

۷) .

۸) .

۹) .

۲-۷- *XML External Entity*

تزریق موجودیت خارجی «*XML*» (همچنین به عنوان «*XXE*» شناخته می‌شود) یک آسیب‌پذیری امنیتی وب است که به مهاجم اجازه می‌دهد در پردازش داده‌های «*XML*» برنامه دخالت کند. «*XML*» (*Extensible Markup Language*) نوعی قالب داده، است که امروزه بسیار استفاده می‌شود. این آسیب‌پذیری اغلب به مهاجم اجازه می‌دهد تا فایل‌ها را در سیستم فایل سرور برنامه مشاهده کند و با هر سیستم پشتیبان یا خارجی که خود برنامه می‌تواند به آن دسترسی داشته باشد، تعامل برقرار کند. [21]

.

.

.

انواع مختلف حملات «*XXE*»

۱) .

۲) .

۳) .

۴) .

در ادمه سناریو ساده‌ای از نحوه کار «*XXE*» مطرح شده‌است:

.

.

.

شکل ۲–۲۶ - XML external entity

۱) .

۲) .

۳) .

۴) .

برای انجام یک حمله تزریق «*XXE*» که یک فایل دلخواه را از سیستم فایل سرور بازیابی می‌کند، باید «*XML*» ارسال شده را به دو روش تغییر داد:

- .
- .

به عنوان مثال، فرض می‌شود یک برنامه خرید با ارسال «*XML*» زیر به سرور، سطح موجودی یک محصول را بررسی می‌کند:

```
.
```

این برنامه هیچ دفاع خاصی در برابر حملات «*XXE*» انجام نمی‌دهد، بنابراین می‌توان از آسیب‌پذیری «*XXE*» برای بازیابی فایل «*etc/passwd/*» با ارسال بار «*XXE*» زیر سوء استفاده کرد:

```
.
```

این محموله «*XXE*» از یک موجودیت خارجی «*&xxe;*» که مقدار آن محتویات فایل «*etc/passwd/*» است و از موجودیت درون مقدار «*productid*» استفاده می‌کند تشکیل شده‌است. این محموله باعث می‌شود که پاسخ برنامه شامل محتویات فایل باشد:

```
.
```

استفاده از «*XXE*» برای انجام حملات «*SSRF*»

جدای از بازیابی داده‌های حساس، تاثیر اصلی دیگر حملات «*XXE*» این است که می‌توان از آنها برای جعل درخواست‌های سمت سرور (*SSRF*) استفاده کرد. این یک آسیب‌پذیری بالقوه جدی است که در آن برنامه سمت سرور می‌تواند درخواست‌های «*HTTP*» را به هر «*URL*»ی که سرور می‌تواند به آن دسترسی داشته باشد، وارد کند.

.

.

.

در مثال «*XXE*» زیر، موجودیت خارجی باعث می‌شود سرور یک درخواست «*HTTP*» پشتیبان را به یک سیستم داخلی در زیرساخت سازمان ارسال کند:

```
.
```

.

.

.

شکل ۲–۲۷ - XXE + SSRF

اکثر مواقع آسیب‌پذیری‌های «*XXE*» را می‌توان به سرعت با استفاده از اسکنر آسیب‌پذیری وب «*Burp Suite*» پیدا کرد.

«*Burp Suite*» یک پلتفرم و ابزار گرافیکی یکپارچه برای انجام تست امنیتی برنامه‌های کاربردی وب است که از کل فرآیند آزمایش، از نقشه برداری اولیه و تجزیه و تحلیل سطح حمله یک برنامه، تا یافتن و بهره‌برداری از آسیب پذیری‌های امنیتی را پشتیبانی می‌کند.

این ابزار به زبان جاوا نوشته شده و توسط «*PortSwigger Web Security*» توسعه یافته است. این ابزار دارای سه نسخه است؛ یک نسخه «*Community*» که می‌توان رایگان دانلود کرد، یک نسخه «*Professional*» و یک نسخه «*Enterprise*» که می‌توان پس از یک دوره آزمایشی خریداری کرد. نسخه «*Community*» به طور قابل توجهی عملکرد را کاهش داده‌است زیرا در نظر دارد یک راه حل جامع برای بررسی‌های امنیتی برنامه‌های وب ارائه کند. این ابزار علاوه بر عملکردهای اساسی مانند سرور پروکسی، اسکنر و نفوذگر، دارای گزینه‌های پیشرفته‌تری مانند «*spider*»، «*a repeater*»، «*a decoder*»، «*a comparer*»، «*an extender*» و «*sequencer*» است.

شکل ۲–۲۸ - ابزار «Burp Suite»

آزمایش دستی برای آسیب‌پذیری‌های «*XXE*» معمولاً شامل موارد زیر است:

- .
- .

- .

روش‌های جلوگیری از آسیب پذیری «XXE»

تقریباً همه آسیب‌پذیری‌های «*XXE*» به این دلیل به وجود می‌آیند که کتابخانه تجزیه «*XML*» برنامه از ویژگی‌های بالقوه خطرناک «*XML*» پشتیبانی می‌کند که برنامه به آن نیازی ندارد یا قصد استفاده از آن‌ها را ندارد. ساده‌ترین و موثرترین راه برای جلوگیری از حملات «*XXE*» غیرفعال کردن آن ویژگی‌ها است.

به طور کلی، غیرفعال کردن وضوح موجودیت‌های خارجی و غیرفعال کردن پشتیبانی برای «*XInclude*» کافی است. این کار معمولاً می‌تواند از طریق گزینه‌های پیکربندی یا با نادیده گرفتن رفتار پیش‌فرض از نظر برنامه‌ریزی انجام شود.

۲-۸- *Broken Access Control*

کنترل دسترسی یا صدور مجوز اعمال محدودیت‌هایی است که چه کسی (یا چه چیزی) می‌تواند اقدامات انجام شده یا دسترسی به منابعی را که درخواست کرده‌است را انجام دهد. در زمینه برنامه‌های کاربردی وب، کنترل دسترسی به احراز هویت و مدیریت جلسه بستگی دارد:

- **احراز هویت** (***Authentication***): احراز هویت، کاربر را شناسایی می‌کند و تأیید می‌کند که او همان چیزی است که می‌گوید.
- **مدیریت جلسه** (***Session management***): مشخص می‌کند که کدام درخواست‌های «*HTTP*» بعدی توسط همان کاربر انجام می‌شود.
- **کنترل دسترسی** (***Access control***): تعیین می‌کند که آیا کاربر مجاز به انجام عملی است که می‌خواهد انجام دهد یا خیر.

نقص کنترل دسترسی یک آسیب‌پذیری امنیتی است که معمولاً یک برنامه تحت‌وب با آن مواجه می‌شود و اغلب بسیار مهم است. طراحی و مدیریت کنترل‌های دسترسی یک مشکل پیچیده و پویا است که محدودیت‌های تجاری، سازمانی و قانونی را برای پیاده‌سازی فنی اعمال می‌کند. تصمیمات طراحی کنترل دسترسی باید توسط انسان گرفته شود نه فناوری، و احتمال خطا زیاد است.

از دیدگاه کاربر، کنترل‌های دسترسی را می‌توان به دسته‌های زیر تقسیم کرد:

- کنترل دسترسی عمودی (*Vertical access controls*)
- کنترل دسترسی افقی (*Horizontal access controls*)

اکنون به توضیح هرکدام از کنترل‌های دسترسی پرداخته می‌شود.

کنترل دسترسی عمودی

.

.

.

کنترل دسترسی افقی

.

.

.

نمونه‌هایی از نقص کنترل دسترسی[1]

نقص‌های کنترل دسترسی زمانی وجود دارند که کاربر در واقع می‌تواند به منابعی دسترسی پیدا کند یا اقداماتی را انجام دهد که قرار نیست به آن دسترسی داشته باشند.

.

.

.

شکل ۲–۲۹ - سناریو «Broken Access Control»

روش‌های ارتقای سطح کنترل عمودی

اگر کاربر بتواند به عملکردی دسترسی پیدا کند که ...

۱) عملکرد محافظت نشده

.

.

.

به عنوان مثال، یک وب سایت ممکن است دارای عملکرد حساس در «*URL*» زیر باشد:

.

این در واقع ممکن است ....

[1] Broken access controls

.

.

.

.

به عنوان مثال، برنامه ای را در نظر بگیرید که توابع مدیریتی را در «*URL*» زیر میزبانی می‌کند:

.

.

.

.

.

.

.

.

.

.

۲) کنترل دسترسی مبتنی بر پارامتر

برخی از برنامه‌ها حقوق دسترسی یا نقش کاربر را در هنگام ورود تعیین می‌کنند و سپس این اطلاعات را در مکانی قابل کنترل توسط کاربر ذخیره می‌کنند، مانند یک فیلد پنهان، کوکی یا پارامتر رشته پرس و جو از پیش تعیین شده. برنامه بر اساس مقدار ارسال شده تصمیمات بعدی کنترل دسترسی را می گیرد. مثلا:

.

این رویکرد اساساً ناامن است زیرا کاربر می‌تواند به سادگی مقدار را تغییر دهد و به عملکردهایی که مجاز به آن نیستند، مانند عملکردهای اداری، دسترسی پیدا کند.

۳) نقص کنترل دسترسی ناشی از پیکربندی نادرست پلتفرم

برخی از برنامه‌ها با محدود کردن دسترسی به «*URL*»های خاص و روش‌های «*HTTP*» بر اساس نقش کاربر، کنترل‌های دسترسی را در لایه پلتفرم اعمال می‌کنند. به عنوان مثال یک برنامه ممکن است قوانینی مانند زیر را پیکربندی کند:

.

این قانون دسترسی به ....

ممکن است با استفاده از درخواستی مانند زیر، کنترل‌های دسترسی را بتوان دور زد:

.

یک حمله جایگزین می‌تواند در رابطه با روش «*HTTP*» مورد استفاده در درخواست ایجاد شود....

ارتقای سطح کنترل افقی

افزایش کنترل افقی زمانی ایجاد می‌شود که ....

به عنوان مثال، یک کاربر معمولاً ممکن است با استفاده از «*URL*» مانند زیر به صفحه حساب خود دسترسی داشته باشد:

.

حال، اگر یک مهاجم ....

آسیب‌پذیری‌های کنترل دسترسی در فرآیندهای چند مرحله‌ای

.

.

.

۹-۲- *Security Misconfiguration*

پیکربندی نادرست امنیتی زمانی رخ می‌دهد که تنظیمات امنیتی به اندازه کافی در فرآیند پیکربندی تعریف نشده باشند یا با تنظیمات پیش‌فرض نگهداری و مستقر شوند. این آسیب پذیری ممکن است بر هر لایه از پشته برنامه، ابر پیکربندی یا شبکه تأثیر بگذارد. ابرهای پیکربندی نادرست یکی از دلایل اصلی نقض داده‌ها هستند که میلیون‌ها دلار برای سازمان‌ها هزینه دارد.

آسیب‌پذیری‌ها معمولاً در طول پیکربندی معرفی می‌شوند. آسیب‌پذیری‌های پیکربندی نادرست با استفاده از موارد زیر رخ می‌دهند:

- .
- .
- .
- .
- .

- .
- .

دلیل رُخ دادن پیکربندی نادرست امنیتی

یک پیکربندی نادرست ممکن است ....

تاثیر حملات پیکربندی نادرست امنیتی

پیکربندی نادرست امنیتی می‌تواند ....

انواع رایج پیکربندی نادرست امنیتی

موارد زیر رخدادهای رایج در یک محیط «IT» است که می‌تواند منجر به پیکربندی نادرست امنیتی شود:

۱) .

۲) .

۳) .

۴) .

۵) .

۶) .

۷) .

۸) .

۹) .

.

.

.

شکل ۲–۳۰ - نمونه‌ای از «Security Misconfiguration»

روش‌های جلوگیری از پیکربندی نادرست امنیتی

گام اولی که باید برای جلوگیری از این آسیب‌پذیری برداشت، این است که ....

۲-۱۰- *Security Logging and Monitoring Failures*

ثبت و نظارت[1]، داده‌های خامی را ارائه می‌دهد که به شناسایی تهدیدهای احتمالی کمک می‌کند. این امر زمانی اتفاق می‌افتد که مدیریت سیستم عمیقاً به داده‌ها نگاه می‌کند و الگوهای غیر معمول را شناسایی می‌کند. این فرآیندها به عنوان ستون‌هایی عمل می‌کنند که پایه و اساس یک چارچوب امنیتی قوی هستند.

در صورت بروز حوادث امنیتی یا از دست رفتن داده‌ها در یک سیستم، ثبت و نظارت به یافتن علت واقعی هر گونه خرابی کمک می‌کند. با این حال، گاهی اوقات نمی‌توان عمیق‌تر مشکل را کاوش کرد و موارد را ردیابی کرد زیرا هیچ گزارش نظارتی وجود ندارد.

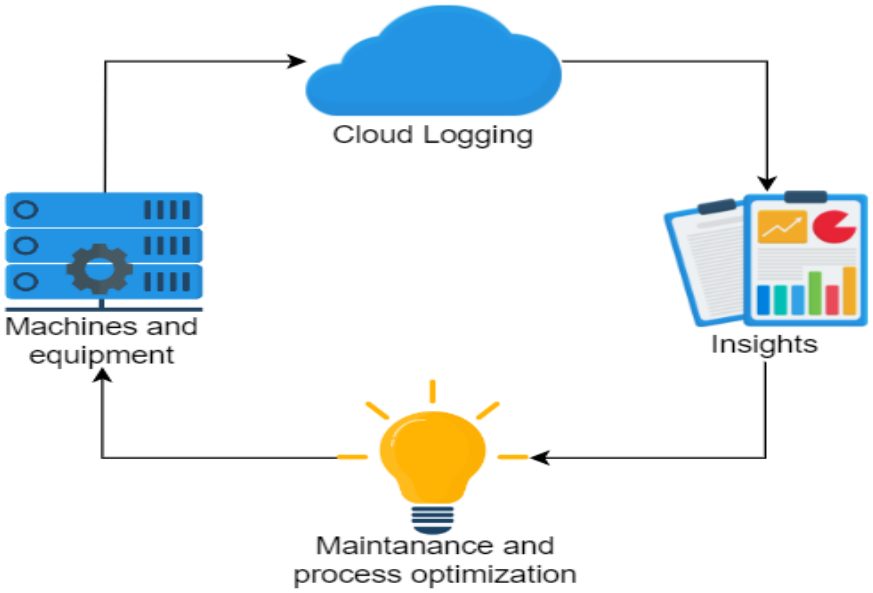
اهمیت سیستم‌های ثبت و نظارت

.

.

.

تصویر زیر نشان می‌دهد که چگونه ثبت‌ها به شناسایی الگوها کمک می‌کنند. این تصویر همچنین اطلاعاتی را برای بهبود و نگهداری سیستم ارائه می‌دهد.



شکل ۲–۳۱ – چگونگی تاثیر ثبت و نظارت به شناسایی الگوهای یک سیستم

[1] Logging and Monitoring

آسیب‌پذیری‌ها و تهدیدات ویژگی ثبت و نظارت

برخی از آسیب‌پذیری‌های ویژگی ثبت و نظارت عبارتند از:

- .
- .
- .
- .
- .

در ادامه برخی از تهدیدات ناشی از ثبت و نظارت ضعیف معرفی می‌شوند:

۱) .
۲) .
۳) .
۴) .
۵) .
۶) .

روش‌های جلوگیری از «logging and monitoring failures»

اقدامات زیر را می‌توان برای جلوگیری از خرابی‌های ثبت و نظارت انجام داد:

- .
- .
- .
- .
- .
- .
- .
- .

فصل سوم

فصل ۳- توضیحات مرتبط با ابزارهای پیاده‌سازی شده

در فصل سوم سه ابزار نوشته‌شده برای پیدا کردن آسیب پذیری «*XSS*» توضیح داده شده‌است.

۳-۱- ابزار «*dozd*»

ابزار اول که «*dozd*» نام‌گذاری شده‌است، وظیفه دارد که بعد از گرفتن یک «*URL*» تمام «*URL*»های مربوط به آن را تا عمق مشخصی پیدا کند. یعنی اگر بطور مثال «*URL*» وب‌سایت دیوار[1] به عنوان ورودی در این ابزار وارد شود ابتدا در همین ورودی تمام «*URL*»های موجود را پیدا کرده، سپس هر یک از آنها را بررسی می‌کند و «*URL*»های موجود در آنها را نیز، پیدا می‌کند. در نتیجه می‌توان تمام صفحات موجود و ارجاع داده شده در یک وب‌سایت را تا عمق مشخصی بدست‌آورد.

.

.

.

.

۳-۲- ابزار «*fallparams-master*»

ابزار دوم که «*Find All Params*» یا «*fallparams*» نام‌گذاری شده‌است، لیستی از «*URL*»های گرفته‌شده از ابزار اول را به عنوان ورودی به آن داده‌می‌شود و بررسی می‌کند که در هریک از «*URL*»ها کدام پارامترها ممکن است به آسیب پذیری «*XSS*» ختم بشوند.

.

[1] https://divar.ir

۳-۳- ابزار «*ayine*»

ابزار سوم که «*ayine*» نام‌گذاری شده است، وظیفه دارد که به ازای هرکدام از «*URL*»های کشف شده از ابزار اول و پارامترهای کشف شده از ابزار دوم، بررسی کند که کدام پارامتر در کدام «*URL*» دارای آسیب‌پذیری «*XSS*» است.

.

۳-٤- نمونه‌های ورودی و خروجی

۱) به ابزار اول یعنی «*dozd*» ورودی زیر که یک «*URL*» است داده می‌شود:

.

خروجی این ابزار شامل «*URL*» ورودی و تمامی «*URL*»های موجود در آن و دیگر دامنه‌های آن، تا عمق ۵ و حداکثر تعداد نتایج ۱۰۰۰۰۰۰ می‌باشد، که به شرح زیر است:

.

۲) به ابزار دوم یعنی «*fallparams*»، «*URL*» زیر به عنوان ورودی داده می‌شود:

.

```
→  fallparams git:(master) fallparams -u "https://google.com"

Params are located at params folder!
```

شکل ۳–1 - ورودی ابزار «fallparams»

خروجی شامل تمامی پارامترهایی است که ممکن است شامل آسیب‌پذیری «*XSS*» باشد:

.
.
.

به دلیل تعداد بالای خروجی‌های این ابزار (۱۳۸۵۴پارامتر) از آوردن تمامی آن‌ها در این گزارش خودداری شده‌است.

۳) ورودی ابزار سوم یعنی «*ayine*» شامل «*URL*»زیر و پارامترهای پیدا شده در آن که امکان آسیب-پذیری «*XSS*» در آنها وجود دارد می‌باشد:

.

```
→  ayine git:(main) node dist/index.js -u "https://mail.greatcall.com" -p ./params/mail.greatcall.com
: 0/1 - Working on
https://mail.greatcall.com
```

شکل ۳–2 - ورودی ابزار «ayine»

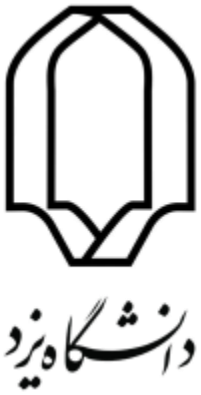
خروجی شامل پارامترهای زیر می‌باشد:

.

همانطور که در شکل ۳-۳ نمایش داده‌است، هیچ‌کدام از پارامترها «*Reflect*» نشده‌اند.

```
→  ayine git:(main) node dist/index.js -u "https://mail.greatcall.com" -p ./params/mail.greatcall.com
Info! No reflected param found!
```

شکل ۳–3 - خروجی ابزار «ayine»



Yazd University
Faculty of Engineering

Security Check of web applications

By

Fatemeh FallahPour Tafti

Supervised by
Dr. FazlAllah AdibNia

September 2022

فصل ۴- فهرست مراجع

[1] OWASP Team. (August 20, 2022) APA Citation. *OWASP Top Ten*. Retrieved from https://owasp.org/www-project-top-ten/

[2] Port Swigger Team. (August 22, 2022) APA Citation. *Server-side template injection*. Retrieved from https://portswigger.net/web-security/server-side-template-injection

[3] Port Swigger Team. (August 23, 2022) APA Citation. *Same-origin policy (SOP)*. Retrieved from https://portswigger.net/web-security/cors/same-origin-policy

[4] Port Swigger Team. (August 17, 2022) APA Citation. *Cross-origin resource sharing (CORS)*. Retrieved from https://portswigger.net/web-security/cors

[5] Port Swigger Team. (August 10, 2022) APA Citation. *Cross-site request forgery (CSRF)*. Retrieved from https://portswigger.net/web-security/csrf

[6] Port Swigger Team. (August 15, 2022) APA Citation. *Cross-site scripting*. Retrieved from https://portswigger.net/web-security/cross-site-scripting

[7] Chris on Code. (August 12, 2022) APA Citation. *The Ins and Outs of Token-Based Authentication*. Retrieved from https://www.digitalocean.com/community/tutorials/the-ins-and-outs-of-token-based-authentication

[8] Port Swigger Team. (August 12, 2022) APA Citation. *Authentication vulnerabilities*. Retrieved from https://portswigger.net/web-security/authentication

[9] Chris on Code. (August 13, 2022) APA Citation. *The Anatomy of a JSON Web Token*. Retrieved from https://www.digitalocean.com/community/tutorials/the-anatomy-of-a-json-web-token

[10] OWASP Team. (August 29, 2022) APA Citation. *Session Management Testing*. Retrieved from https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/06-Session_Management_Testing

[11] M. Jones, J. Bradley & N. Sakimura. (August 31, 2022) APA Citation. *Registered Claim Names*. Retrieved from https://www.rfc-editor.org/rfc/rfc7519#section-4.1

[12] CWE Team. (August 30, 2022) APA Citation. *Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')*. Retrieved from https://cwe.mitre.org/data/definitions/78.html

[13] Ticarpi. (September 1, 2022) APA Citation. *The JSON Web Token Toolkit*. Retrieved from https://github.com/ticarpi/jwt_tool

[14] Fortinet Team. (September 1, 2022) APA Citation. *What is Single Sign-on (SSO)*. Retrieved from https://www.fortinet.com/resources/cyberglossary/single-sign-on#:~:text=Single%20sign%2Don%20(SSO)%20is%20an%20identification%20method%20that,the%20authentication%20process%20for%20users

[15] Port Swigger Team. (August 17, 2022) APA Citation. *OAuth 2.0 authentication vulnerabilities*. Retrieved from https://portswigger.net/web-security/oauth

[16] Port Swigger Team. (August 18, 2022) APA Citation. *How to prevent OAuth authentication vulnerabilities*. Retrieved from https://portswigger.net/web-security/oauth/preventing

[17] OWASP Team. (August 19, 2022) APA Citation. *Sensitive Data Exposure*. Retrieved from https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure

[18] Code Maze Team. (August 20, 2022) APA Citation. *OWASP Top 10 – Sensitive Data Exposure*. Retrieved from https://code-maze.com/owasp-top-10-sensitive-data-exposure/#:~:text=Sensitive%20data%20exposure%20usually%20occurs,expose%20different%20types%20of%20data

[19] Port Swigger Team. (August 20, 2022) APA Citation. *Server-side request forgery (SSRF)*. Retrieved from https://portswigger.net/web-security/ssrf

[20] M. Dahan. (August 21, 2022) APA Citation. *Server-side request forgery (SSRF) attacks and how to prevent them*. Retrieved from https://www.comparitech.com/blog/information-security/server-side-request-forgery-attacks/#Defending_against_SSRF_attacks

[21] Port Swigger Team. (August 20, 2022) APA Citation. *XML external entity (XXE) injection*. Retrieved from https://portswigger.net/web-security/xxe

[22] Port Swigger Team. (August 23, 2022) APA Citation. *Access control vulnerabilities and privilege escalation*. Retrieved from https://portswigger.net/web-security/access-control

[23] OWASP Team. (August 21, 2022) APA Citation. *XML External Entity (XXE) Processing*. Retrieved from https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing

[24] OWASP Team. (August 28, 2022) APA Citation. *Security Logging and Monitoring Failures*. Retrieved from https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/

[25] OWASP Team. (August 30, 2022) APA Citation. *Security Misconfiguration*. Retrieved from https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

[26] A. Dizdar. (September 3, 2022) APA Citation. *Security Misconfiguration: Impact, Examples, and Prevention*. Retrieved from https://brightsec.com/blog/security-misconfiguration/

[27] Educative Team. (September 1, 2022) APA Citation. *What are security logging and monitoring failures*. Retrieved from https://www.educative.io/answers/what-are-security-logging-and-monitoring-failures