۱۰ آسیب پذیری رایج OWASP - ۲۰۱۷

رایج‌ترین ریسک‌های امنیتی برنامه های تحت وب

مطالب بیشتر در:
www.bigdata-ir.com

ارائه : ابوالفضل جعفری

راهنما : استاد حبیبی

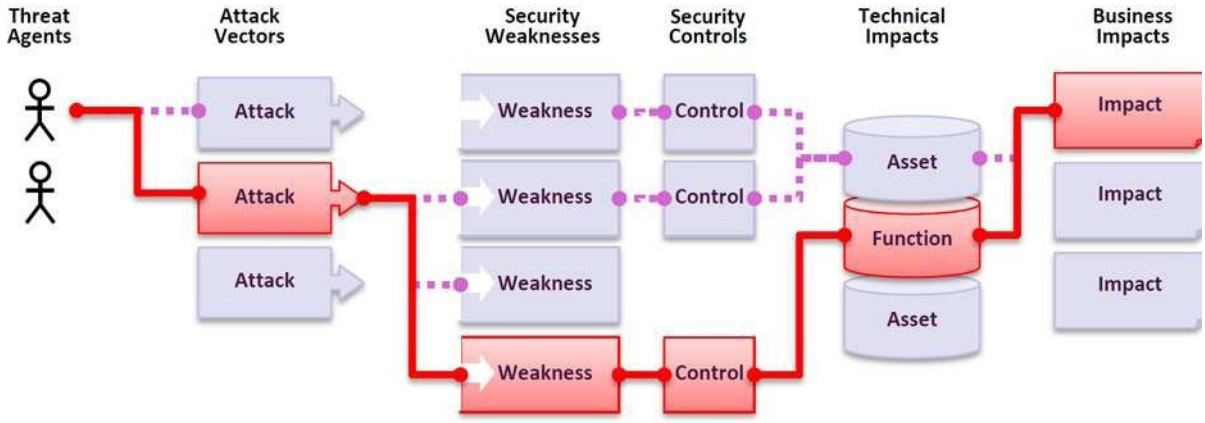
مطالب بیشتر در:
www.bigdata-ir.com

رایانش امن
نیمسال دوم ۹۷-۹۶

ریسک‌های امنیتی برنامه های کاربردی

تهدیدات امنیتی برنامه های کاربردی چیست؟

مهاجمان به طور بالقوه میتوانند از نرم افزار شما به روشهای مختلف استفاده کنند و به کسب وکار و یا سازمان شما آسیب برسانند. هر یک از این روشها یک تهدید محسوب میشود که ممکن است به اندازه ی لازم جدی باشد.



گاهی اوقات این روشها جهت تشخیص و بهره برداری، بی اهمیت هستند و گاهی نیز بسیار دشوار هستند .به طور مشابه، آسیب ناشی از این امر ممکن است هیچ نتیجه ای نداشته باشد یا ممکن است سازمان را از کسب وکار حذف کند. برای تعیین میزان تهدید برای سازمانها، می توان احتمال را با هر عامل تهدید، بردار حمله و ضعف امنیتی مرتبط کرد و آن را با برآورد تاثیرات فنی و تجاری سازمان خود ترکیب کرد. در مجموع این عوامل، ریسک کلی سازمان را تعیین میکند.

تهدید برای من چیست؟

OWASP Top10 بر شناسایی جدی ترین ریسک های مجموعه وسیعی از سازمانها تمرکز میکند. برای هر یک از این ریسک ها، اطلاعات کلی درباره احتمال و تأثیر فنی آن با استفاده از طرح رتبه بندی ساده زیر، ارائه شده است که این نیز بر اساس روش رتبه بندی ریسک OWASP تنظیم شده است.

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

در این نسخه، سیستم رتبه بندی ریسک برای کمک در محاسبه احتمال و تاثیر فنی هر یک از ریسک ها به روز رسانی شده است. برای اطلاعات بیشتر این صفحه را مشاهده کنید : Note About Risks

سازمان ها منحصر به فردند. بنابر این عوامل تهدید برای سازمان ها، اهداف آنها، و اثر هر نفوذ بر سازمان های مختلف نیز متفاوت است. اگر یک سازمان عام المنفعه، از یک سیستم مدیریت محتوا برای اطلاعات عمومی استفاده کند و یک سیستم سلامت از همان CMS برای رکوردهای حساس بخش سلامت استفاده کند، عامل تهدید و تاثیرات تجاری برای یک برنامه یکسان (CMS) بسیار متفاوت خواهند بود، شناسایی ریسک های سازمان خود بر اساس عوامل تهدید قابل اعمال و تاثیرات تجاری بسیار حیاتی است.

هرجا که امکان داشته، نام های ریسک ها در TOP10 با CWE برای ایجاد یکدستی در نام ها و جلوگیری از پراکندگی در اسامی تطبیق داده شده اند.

منابع

OWASP

OWASP Risk Rating Methodology

Article on Threat/Risk Modeling

External

ISO 31000: Risk Management Std

ISO 27001: ISMS

NIST Cyber Framework

ASD Strategic Mitigations

NIST CVSS 3.0

Microsoft Threat Modelling Tool

ریسک‌های امنیتی برنامه های کاربردی

درباره OWASP

OWASP (پروژه باز امنیت برنامه های تحت وب) انجمنی است که به سازمانها اجازه توسعه، خریداری و نگهداری نرم افزارها و API ها به شکلی مطمئن و امن می‌دهد. در OWSAP موارد زیر بصورت آزاد و باز وجود دارند:

- ابزارهای امنیت نرم افزار و استانداردها
- کتابهای کامل در مورد تست برنامه ، توسعه کد امن و بررسی کد امن
- کنترل های امنیتی استاندارد و کتابخانه ها
- مقالات جهانی
- تحقیقات لبه فناوری
- کنفرانسهای گسترده در سراسر جهان
- فهرست ایمیل‌های پستی

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

همه ابزارها، اسناد ،انجمن ها و مقالات OWASP برای همه رایگان و باز است .OWASP به هیچ شرکت فناوری وابسته نیست ،ولی استفاده آگاهانه از فناوری های امنیتی تجاری را حمایت میکند .OWASP مانند بسیاری از پروژه های نرم افزاری منبع باز است و انواع مختلفی از منابع اصلی را به صورت مشارکتی و آزاد تولید می کند.

OWASP یک نهاد انتفاعی نیست که موفقیت درازمدت پروژه اش را تضمین کند .تقریبا تمامی افراد مرتبط با OWASP ازجمله اعضای هیئت مدیره، مشاوران مقاله ، مشاوران و اعضای پروژه، به صورت داوطلبانه کار می کنند.

یک نرم افزار ناامن می تواند امور مربوط به مالی، سلامت، امنیت و دیگر زیرساختهای حیاتی شما را تضعیف کند. به همان نسبت که نرم افزارها به شکل وسیعی، بحرانی، پیچیده و بهم پیوسته می شوند، دستیابی امنیت نرم افزار نیز مشکل تر می شود. پیشروی سریع فرآیندهای مدرن توسعه نرم افزار، ایجاب می کند ریسک های رایج تر هرچه سریعتر کشف و برطرف شوند.

اگرچه هدف اصلی پروژه TOP 10 افزایش آگاهی بین توسعه دهندگان و مدیران بود، اما به استاندارد امنیتی برنامه ها تبدیل شد.

OWASP Top 10 برای اولین بار در سال ۲۰۰۳ منتشر شد و در سال ۲۰۰۴ و ۲۰۰۷ بهروزرسانی های جزئی بر روی آن انجام شد. نسخه ۲۰۱۰ جهت اولویت بندی با ریسک، بهبود داده شد و این الگو در سال ۲۰۱۳ و آخرین نسخه آن در سال ۲۰۱۷ ادامه یافت.

برای شروع کار سازمانتان، به شما پیشنهاد میشود که از ToP 10 استفاده کنید .توسعه دهندگان میتوانند از اشتباهات دیگر سازمانها یاد بگیرند. مدیران بایستی در مورد نحوه مدیریت ریسک هایی که برنامه های کاربردی و API ها در شرکتشان ایجاد می کنند، فکر کنند.

در درازمدت، شما بایستی یک نرم افزار امن سازگار با فرهنگ و فناوری خود ایجاد کنید. این نرم افزارها در تمامی فرمها و اندازه ها وجود دارند.

منابع

OWASP

OWASP Risk Rating Methodology

Article on Threat/Risk Modeling

External

ISO 31000: Risk Management Std

ISO 27001: ISMS

NIST Cyber Framework

ASD Strategic Mitigations

NIST CVSS 3.0

Microsoft Threat Modelling Tool

توضیحات

با شتاب تغییرات در طول چهار سال گذشته OWASP TOP10 نیاز به تغییر داشت . ما به طور کامل متودولوژی OWASP TOP10 را بازنویسی کردیم , روش جدیدی برای فرآیند فراخوانی داده ها استفاده کردیم , با انجمن ها کار کردیم , ریسک ها را بازنویسی کرده , و منابع را به چارچوب‌ها و زبان‌هایی که امروزه به شکل وسیع مورد استفاده قرار می‌گیرند , اضافه کردیم .

در چند سال گذشته , تکنولوژی بنیادی و معماری برنامه های کاربردی به طور قابل‌توجهی تغییر کرده‌است :

* Microservice های نوشته شده در node.js و Spring Boot , جایگزین کاربردهای یکپارچه سنتی شده‌اند . Microservice ها چالش‌های امنیتی از جمله ایجاد اعتماد بین microservices , Containers , Secret Management , و غیره را دارند . کدهای قدیمی که هرگز انتظار نمی‌رود که از اینترنت در دسترس باشد در حال حاضر پشت یک API یا سرویس وب RESTful قرار دارد تا توسط برنامه های کاربردی و برنامه های کاربردی تلفن همراه مورد استفاده قرار گیرند. فرضیات معماری از طریق کد , مانند Trusted Callers , دیگر معتبر نیستند .

* جاوا اسکریپت در حال حاضر زبان اولیه وب است با node.js قابل اجرا در سمت سرور و چارچوب های وب مدرن مانند Angular ،Electron ،Bootstrap و React قابل اجرا بر روی سرویس گیرنده.

موارد جدید، ارائه شده توسط داده ها

A4: 2017-XML Extended Entities XXE یک دسته جدید است که در ابتدا توسط مجموعه داده های ابزارهای تست و تجزیه و تحلیل امنیت کد منبع (SAST) ارائه شده است .

موارد جدید، ارائه شده توسط انجمن

ما از انجمن خواستیم نگاه دقیق تری را به دو دسته ی ضعف داشته باشد. پس از حدود ۵۰۰ تقاضای همکاری و حذف مسائلی که قبلا توسط داده ها ارائه شده بود (مانند افشای داده های حساس و XXE) دو مورد جدید عبارتند از:

A8: 2017- Insecure Deserialization : که اجازه اجرای کد دلخواه یا دستکاری اشیای حساس را در سیستم عامل های تحت تاثیر می دهد.

A10: 2017- Insufficient Logging and Monitoring : نقص موردی که می تواند از فعالیت های مخرب جلوگیری و یا به طور قابل ملاحظه ای آن را به تاخیر بیندازد ، و نبود آن باعث ایجاد مشکل در تشخیص، پاسخگویی به رخداد، و جرم شناسی های دیجیتال می شود.

ادغام شده یا بازنشسته، اما فراموش نشده:

A4 - Insecure Direct Object References , A7 : Missing Function Level Access Control در A5 : 2017 - Broken Access Control ادغام شدند.

A8 – Cross site request Forgery CSRF ، با اینکه که بسیاری از چارچوب ها دفاع از CSRF را هم شامل می شوند، تنها در 5% از برنامه ها یافت شد.

A10 - Unvalidated Redirects and Forwards : در حالی که در حدود ۸ درصد از برنامه های کاربردی یافت می شود، توسط XXE به طور کامل پوشش داده شده است.

OWASP Top 10 - 2013		OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	∪	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	∪	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

ریسک‌های امنیتی برنامه های کاربردی - ۲۰۱۷

A1:2017 Injection

ضعف تزریق، مانند OS، NoSQL، SQL، و تزریق LDAP زمانی رخ می دهد که داده های نامعتبر به عنوان بخشی از فرمان یا پرس و جو به یک مفسر ارسال شوند. داده های مخرب مهاجم می تواند مفسر را به اجرای دستورات غیرمنتظره یا دسترسی به داده ها بدون مجوز مناسب منجر کند.

A2:2017-Broken Authentication

توابع درخواست مربوط به احراز هویت و مدیریت نشست اغلب به اشتباه اجرا می شوند، و به مهاجمان اجازه می دهند رمزهای عبور، کلید ها یا نشانه های نشست به خطر بیافتد یا از ضعف های دیگر پیاده سازی استفاده کنند تا از هویت های دیگر کاربران به طور موقت یا دائمی بهره برداری کنند.

A3:2017Sensitive Data Exposure

بسیاری از برنامه های کاربردی وب و API ها از اطلاعات حساس مانند مالی، سلامت و PII به درستی محافظت نمی کنند. مهاجمان ممکن است این اطلاعات به درستی محافظت نشده را، سرقت یا برای سایر مقاصد مورد استفاده قرار داده، و یا تغییر دهند. داده های حساس ممکن است بدون حفاظت اضافی، مانند رمزگذاری در حالت استراحت یا در حین حمل، به خطر بیافتد.

A4:2017 XML External Entities (XXE)

بسیاری از پردازنده های قدیمی تر یا ضعیف پیکربندی شده XML، ارجاعات موجودیت بیرونی را در اسناد XML ارزیابی می کنند. موجودیت بیرونی می تواند به افشای فایل های داخلی با استفاده از فایل URI فایل، اشتراک فایل های داخلی، اسکن پورت داخلی ، اجرای کد از راه دور و حملات انکار سرویس منجر شود.

A5:2017 Broken Access Control

محدودیت هایی که کاربران مجاز به انجام آن مجاز هستند اغلب به درستی اعمال نمی شوند. مهاجمان می توانند از این ضعف ها برای دسترسی به قابلیت های غیر مجاز و یا داده ها مانند دسترسی به حساب های دیگر کاربران، مشاهده فایل های حساس، تغییر دادن داده های کاربران دیگر، تغییر حقوق دسترسی و غیره استفاده کنند.

A6:2017 Security Misconfiguration

تنظیمات امنیتی اشتباه یکی از رایج ترین مسائل امنیتی موجود است . که معمولا نتیجه ای از تنظیمات پیش فرض ناامن، پیکربندی های ناقص و غیر مجاز، ذخیره سازی ابر باز، هدر های HTTP غلط تنظیم شده و پیام های خطای حاوی اطلاعات حساس است. نه تنها تمام سیستم عامل ها، چارچوب ها، کتابخانه ها و برنامه های کاربردی باید به صورت ایمن پیکربندی شوند، بلکه باید آنها را به موقع اعمال و به روز رسانی کرد.

A7:2017 Cross-Site Scripting (XSS)

نقص های XSS زمانی رخ می دهد که برنامه شامل اطلاعات غیر قابل اعتماد در یک صفحه وب جدید بدون اعتبارسنجی مناسب باشد یا یک صفحه وب موجود را با داده های ارائه شده توسط کاربر با استفاده از یک API مرورگر که می تواند HTML یا جاوااسکریپت ایجاد کند، به روز می کند. XSS به مهاجمان امکان اجرای اسکریپت ها در مرورگر قربانی را می دهد که می تواند جلوی کاربر را بگیرد، وب سایت ها را خراب کند یا کاربر را به سایت های مخرب هدایت کند.

A8:2017 Insecure Deserialization

deserialization نا امن منجر به اجرای کد از راه دور می شود. حتی اگر معایب deserialization باعث اجرای کد از راه دور نباشند، از آنها می توان برای انجام حملات استفاده کرد، از جمله حملات تزریق و حملات تشدید امتیاز.

A9:2017 Using Components with Known Vulnerabilities

اجزاء مانند کتابخانه ها، چارچوب ها و دیگر ماژول های نرم افزاری، دارای دسترسی هایی مشابه با برنامه می باشند. اگر یک جزء آسیب پذیر مورد سوء استفاده قرار گیرد، حمله می تواند باعث از دست رفتن اطلاعات جدی باشد.

A10:2017 Insufficient Logging & Monitoring

نظارت و ثبت وقایع ناقص، همراه با عدم واکنش صحیح به حادثه، به مهاجمان اجازه می دهد تا به سیستم های بیشتری حمله کنند، یا اقدام به استخراج و یا نابودی اطلاعات کنند. بیشتر مطالعات نشان می دهد زمان برای تشخیص آسیب بیش از ۲۰۰ روز است، که معمولا توسط شرکت های بیرونی، نه از طریق نظارت داخلی کشف میشوند.

A1 :2017

تزریق

بردارهای حمله		ضعف امنیتی		تاثیرات	
App. Specific	قابلیت بهره برداری :۳	شیوع : ۲	قابلیت تشخیص : ۳	فنی : ۳	تجاری : ؟
تقریباً هر منبع داده ای می تواند یک عامل تزریق باشد. پارامترها، متغیرها، سرویس های وب داخلی و خارجی، و همه انواع کاربران نیز میتوانند عامل تزریق باشند. نقض تزریق زمانی رخ میدهد که یک نرم افزار داده های غیرقابل اعتماد را به مفسر ارسال کند.		نقض تزریق بسیار شایع است به ویژه در کدهایی که در آن ارثبری وجود دارد. این نقص اغلب در query های XPath، SQL، LDAP، یا NoSQL؛ دستورات OS، پارسرهای XML، هدرهای SMTP، زبانهای expression و غیره وجود دارد. تشخیص نقص تزریق هنگام بازبینی کد راحت است، اما هنگام تست کد بسیار دشوار است .اسکنرها و فازرها، مهاجمان را برای پیدا کردن نقصهای تزریق، یاری میکنند.		تزریق میتواند باعث تخریب یا از دستدادن اطلاعات، عدم پاسخگویی یا رد دسترسی شود. تزریق میتواند گاهی موجب تصاحب کامل میزبان نیز شود . تاثیر تجاری بستگی به نیازهای برنامه و داده دارد.	

آیا برنامه کاربردی آسیب‌پذیر است؟

برنامه در شرایط زیر به حمله آسیب پذیر است:

* ورودی های کاربر اعتبارسنجی یا فیلتر نشوند.
* داده های مخرب به طور مستقیم با استفاده از پرس و جو های پویا مورد استفاده قرار می گیرد.
* داده های مخرب در پارامترهای جستجوی ORM برای دسترسی به اطلاعات حساس یا تمام رکوردها به کار می رود.
* داده های مخرب یا به طور مستقیم یا ترکیب با دستورات SQL، استفاده شوند.

بعضی از تزریقات رایج عبارتند از SQL، دستور LDAP، ORM، OS و Expression Language EL و یا تزریق OGNL . سازمانها می توانند ابزار SAST و DAST را برای این منظور در اختیار بگیرند.

بررسی دستی و اتوماتیک کد منبع بهترین روش تشخیص این آسیب پذیری است .

نحوه پیشگیری از حمله

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

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

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

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

مثال‌هایی از سناریوهای حمله

سناریو شماره ۱: یک نرم افزار از داده های نامطمئن در ساختار دستور فراخوانی SQL آسیب‌پذیر زیر استفاده میکند:

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

سناریو شماره ۲: به طور مشابه، اعتماد کورکورانه نرم افزار به چارچوب ها، ممکن است منجر به نمایش داده هایی که هنوز آسیب پذیرند، شود به عنوان مثال، Hibernate Query Language HQL:

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

در هر دو مورد، مهاجم مقدار پارامتر id را در مرورگر خود به ‹ یا ۱›=‹۱ تغییر میدهد و ارسال میکند. برای مثال:

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

این مفهوم هر دو پرسش را تغییر میدهد تا تمام پرونده ها را از جدول accounts، بازیابی کند .حمله های خطرناک، داده ها را تغییر داده یا حتی روال ذخیره شده را فراخوانی میکند.

منابع

OWASP

OWASP Risk Rating Methodology

Article on Threat/Risk Modeling

External

ISO 31000: Risk Management Std

ISO 27001: ISMS

NIST Cyber Framework

A2 :2017 احراز هویت نقض‌شده

بردارهای حمله		ضعف امنیتی		تاثیرات	
App. Specific	قابلیت بهره برداری :۳	شیوع : ۲	قابلیت تشخیص : ۲	فنی : ۳	تجاری : ؟
مهاجمان به صدها میلیون نام کاربری و رمز عبور برای استفاده به عنوان لیست کاربران ادمین پیش فرض، حملات سورد خودکار و ابزارهای حمله دسترسی دارند.		شیوع این حمله به دلیل نحوه طراحی و پیاده سازی کنترل دسترسی و احراز هویت ها بسیار گسترده است.مدیریت نشست، پایه و اساس احراز هویت و کنترل دسترسی است و در تمامی برنامه های stateful وجود دارد. مهاجم با استفاده از ابزارهای دستی و بهره برداری از آنها با ابزارهای خودکار و استفاده از لیست رمزهای عبور و حملات کتابخانه ای قادر به تشخیص احراز هویت شکسته! خواهد بود.		مهاجمان برای نفوذ به یک سیستم باید به تعدادی حساب کاربری و یا یک حساب کاربری ادمین دسترسی پیدا کنند. بر اساس ناحیه برنامه، این می تواند منجر به پولشویی، کلاهبرداری امنیتی اجتماعی و سرقت هویت و یا انتشار اطلاعات بسیار حساس محافظت شده شود.	

آیا برنامه کاربردی آسیب‌پذیر است؟

تایید هویت کاربر، احراز هویت و مدیریت نشست برای حفاظت در برار حملات مبتنی بر احراز هویت ، بسیار حائز اهمیت است.

ضعف های آسیب پذیری وجود خواهند داشت اگر برنامه کاربردی:

* اجازه حملات خودکار را به مهاجم بدهد. مانند حملات جاسازی احراز هویت ، که در آن مهاجم لیستی از نامهای کاربری و رمزهای عبور مجاز را در اختیار دارد.
* اجازه حملات خودکار رمز عبور یا حملات خودکار دیگر را بدهد.
* اجازه ثبت رمزهای عبور پیش‌فرض، ضعیف یا شناخته شده مانند "Password1" یا "admin/admin" را بدهد.
* از فرآیندهای ضعیف یا ناکارآمد بازیابی یا فراموشی احراز هویت مانند –پرسش و پاسخ های دانشی– استفاده کند که امن نیستند.
* از داده های کشف، رمز یا هش شده با الگوریتم های ضعیف استفاده کند. (A3: ۲۰۱۷ را ببینید)
* از روش های احراز هویت چند مرحله ای ناکارآمد استفاده کند.
* شناسه نشست در URL قابل مشاهده باشد. (مثل URL Rewriting)
* شناسه نشست پس از ورود موفق، تغییر نکرده باشد.
* شناسه های نشست تداوم نداشته باشند. نشست های کاربر یا توکن های احراز هویت (به خصوص توکن های Single sign-on SSO) در هنگام خروج از سیستم یا

نحوه پیشگیری از حمله

* برای جلوگیری از حملات GPU Cracking کلمه عبور را با استفاده از یک تابع هش مدرن یکطرفه، مانند Argon2 یا PBKDF2 ذخیره کنید.
* رمزهای عبور ضعیف را بررسی کنید، مانند تست گذرواژه های جدید یا تغییر یافته در برابر لیست ۱۰۰۰۰ رمز عبور بد.
* تنظیم طول رمز عبور، پیچیدگی و سیاست های تغییر با دستورالعمل های NIST B 63-800
* اطمینان از امنیت فرآیند ثبت نام، بازیابی مشخصات کاربری، و مسیرهای API در برابر حمله ی account Enumeration با استفاده از پیام های مشابه برای تمام نتایج
* در صورت امکان، احراز هویت چند مرحله ای را برای جلوگیری از انواع حملات خودکار و سرقت اطلاعات کاربری اعمال کنید
* تأیید هویت های ناموفق را ثبت کنید و هنگام تشخیص حملات به مدیران هشدار د هید

مثال‌هایی از سناریوهای حمله

سناریو شماره ۱: یک نرم افزار از داده های نامطمئن در ساختار دستور فراخوانی SQL آسیب‌پذیر زیر استفاده میکند:

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

سناریو شماره ۲: به‌طور مشابه، اعتماد کورکورانه نرم افزار در چارچوب ها، ممکن است منجر به نمایش داده هایی که هنوز آسیب پذیرند، شود) به عنوان مثال، Hibernate Query Language (HQL):

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

در هر دو مورد، مهاجم مقدار پارامتر id را در مرورگر خود به ‹ یا ۱=›۱ تغییر میدهد و ارسال میکند. برای مثال:

http://example.com/app/accountView?id=› or‹=۱1 ‹ این مفهوم هر دو پرسش را تغییر میدهد تا

منابع

OWASP

OWASP Risk Rating Methodology

Article on Threat/Risk Modeling

External

ISO 31000: Risk Management Std

ISO 27001: ISMS

NIST Cyber Framework

A3 :2017 افشای اطلاعات حساس

بردارهای حمله		ضعف امنیتی		تاثیرات	
App. Specific	قابلیت بهره برداری :۲	شیوع : ۳	قابلیت تشخیص : ۲	فنی : ۳	تجاری : ؟
مهاجمان بجای حملات مخفیانه به صورت مستقیم، اقدام به سرقت کلیدها، اجرای حملات مرد میانی، یا سرقت اطلاعات رمز نشده سمت سرور یا کاربر، هنگام انتقال اطلاعات می کنند. برای مثال مرورگر. معمولاً یک حمله دستی لازم است. قبلاً بانک های داده رمز عبور بازیابی شده باید توسط GPU ها مورد حمله brute force قرار می گرفت.		در طول چند سال گذشته این به یکی از حملات رایج پرآسیب تبدیل شده است. بزرگترین اشتباه رایج هم رمز نکردن اطلاعات حساس است. برای اطلاعات در حال انتقال، ضعف های سمت سرور به راحتی قابل تشخیص هستند.		یک شکست امنیتی عموماً تمامی داده هایی که باید محافظت شوند را به خطر می اندازد. به طور معمول، این اطلاعات شامل اطلاعات شخصی حساس (PII) مانند پرونده های بهداشتی، اعتبارنامه، اطلاعات شخصی و کارت های اعتباری است که اغلب نیاز به حفاظت بر اساس قوانین یا مقرراتی مانند GDPR اتحادیه اروپا یا قوانین حفظ حریم خصوصی محلی دارند.	

آیا برنامه کاربردی آسیب‌پذیر است؟

اولین نکته این است که نیازهای حفاظت از داده ها در هنگام انتقال و ثابت مشخص شود. به عنوان مثال، رمزهای عبور، شماره کارت اعتباری، پرونده های بهداشتی، اطلاعات شخصی و اسرار تجاری، حفاظت بیشتری نیاز دارند، به ویژه اگر داده ها تحت قانون حریم خصوصی قرار بگیرند، برای مثال مقررات حفاظت کلی اطلاعات اتحادیه اروپا (GDPR) یا مقررات، مانند حفاظت از اطلاعات مالی مانند PCI Data Security Standard PCI DSS. **برای چنین اطلاعاتی:**

* آیا داده ها به شکل رمز نشده ارسال شده اند؟ این مربوط به پروتکل هایی مانند HTTP، SMTP و FTP است به ویژه ترافیک اینترنتی خارجی خطرناک است. تمام ترافیک داخلی را بررسی کنید، به عنوان مثال بین load balancer ها، وب سرور ها، و یا سیستم های back-end .
* آیا داده های حساس به صورت کشف ذخیره می شوند، از جمله پشتیبان ها؟
* آیا الگوریتم رمزنگاری قدیمی یا ضعیف استفاده می شود؟
* آیا رمزگذاری اعمال نشده است، به عنوان مثال آیا هر کدام دستورالعمل های امنیتی یا هدر های امنیتی کاربر (مرورگر) از دست رفته است؟
* آیا عامل کاربر (مثلا برنامه، مشتری پست الکترونیکی) تأییدیه گواهی سرور دریافت شده را تأیید نمی کند؟
* شناسه نشست در URL قابل مشاهده باشد. (مثل URL Rewriting)
* شناسه نشست پس از ورود موفق، تغییر نکرده باشد.
* شناسه های نشست تداوم نداشته باشند. نشست های کاربر یا توکن های احراز هویت (به خصوص توکن های Single sign-on SSO) در هنگام خروج از سیستم یا عدم فعالیت در بازه ای از زمان، نامعتبر نمی شود.

نحوه پیشگیری از حمله

حداقل موارد زیر را دنبال کنید و مراجع را دنبال کنید:

* داده های پردازش شده، ذخیره شده، و یا ارسال شده توسط یک برنامه را طبقه بندی کنید. بر اساس قوانین حریم خصوصی، الزامات قانونی یا نیازهای تجاری، داده های حساس را شناسایی کنید. طبق طبقه بندی، کنترل ها را اعمال کنید.
* در صورت عدم نیاز، داده های حساس را ذخیره نکنید. در اسرع وقت آن را حذف کنید یا از توافق PCI DSS برای علامت گذاری یا حتی ناقص سازی استفاده کنید. داده هایی که ذخیره نمی شوند نمی توانند سرقت شوند.
* اطمینان حاصل کنید که همه اطلاعات حساس در حالت ذخیره را رمزگذاری کرده اید.
* اطمینان حاصل کنید از الگوریتم ها، پروتکل ها و کلیدهای استاندارد به روز و قوی در جای خود استفاده می شود. از مدیریت کلید مناسب استفاده کنید.
* رمزگذاری تمام داده های در حال انتقال با پروتکل های امن مانند TLS با رمزهای محرمانه بدون نقص (PFS)، اولویت بندی رمز توسط سرور و پارامترهای امن. اعمال رمزگذاری با استفاده از دستورالعمل هایی مانند HTTP Security Transport Strict Security HSTS.
* برای پاسخ هایی که حاوی اطلاعات حساس هستند، ذخیره سازی (Caching)غیرفعال شود.
* رمزهای عبور را با استفاده از توابع هش قوی منطبق و هش salting با یک عامل کار (عامل تاخیر) مانند Argon2 و Scrypt و bcrypt یا BFKDF2 ذخیره کنید.
* به طور مستقل اثربخشی پیکربندی و تنظیمات را بررسی کنید.

مثال‌هایی از سناریوهای حمله

سناریو شماره ۱: یک برنامه شماره های کارت اعتباری را در پایگاه داده با استفاده از رمزگذاری خودکار پایگاه داده رمزنگاری می کند. با این حال، هنگام فراخوانی، این داده ها به صورت خودکار رمزگشایی می شود، و به یک نقص تزریق SQL اجازه می دهد شماره کارت اعتباری را در حالت کشف بازیابی کند.

سناریو شماره ۲: یک سایت از TLS برای تمام صفحات استفاده نمی کند و یا از رمزنگاری ضعیف پشتیبانی می کند. یک مهاجم ترافیک شبکه را شنود می کند (به عنوان مثال در یک شبکه بی سیم ناامن)، ارتباطات را از HTTPS به HTTP تنزل می دهد، درخواست های بین کاربر اصلی و سرور را قطع می کند، و کوکی نشست کاربر را سرقت می کند. مهاجم پس از آن از این کوکی استفاده می کند و نشست کاربر (احراز هویت شده) را دزدیده، به داده های شخصی کاربر دسترسی پیدا می کند یا آنها را تغییر می دهد. به جای موارد بالا می تواند تمام داده های منتقل شده مثل دریافت کننده یک انتقال مالی را تغییر دهد.

سناریو شماره ۳: پایگاه داده رمز عبور از هش های unsalted یا ساده برای ذخیره کلمه عبور استفاده می کند. یک نقص آپلود فایل به مهاجم اجازه می دهد رمز عبور پایگاه داده را بازیابی کند. تمام هش های unsalted را می توان با یک جدول از هش های پیش محاسبه شده شکست. هش های تولید شده توسط توابع هش ساده یا سریع ممکن است توسط GPU ها شکسته شوند، حتی اگر از نوع salted باشند.

منابع

OWASP

OWASP Risk Rating Methodology

Article on Threat/Risk Modeling

External

ISO 31000: Risk Management Std

ISO 27001: ISMS

NIST Cyber Framework

XML External Entities (XXE)

A4
:2017

بردارهای حمله		ضعف امنیتی		تاثیرات	
App. Specific	قابلیت بهره برداری :۲	شیوع : ۲	قابلیت تشخیص : ۳	فنی : ۳	تجاری : ؟
اگر مهاجمان بتوانند XML بارگذاری کنند یا محتوای آلوده در یک سند XML وارد کنند، از کد ها ، وابستگی ها یا ادغام های آسیب پذیر استفاده کنند، می توانند از پردازنده های XML آسیب پذیر برای مقاصد خود بهره برداری کنند.		به طور پیش فرض، بسیاری از پردازنده های قدیمی تر XML اجازه تعیین یک موجود خارجی را می دهند،(یک URI که در هنگام پردازش XML ارزیابی می شود). ابزارهای SAST می توانند این مسئله را با بررسی وابستگی ها و پیکربندی کشف کنند. ابزارهای DAST نیاز به مراحل دستی بیشتر برای شناسایی و بهره برداری از این مسئله دارند. آزمایش کنندگان دستی باید برای چگونگی آزمایش XXE آموزش ببینند، زیرا معمولا از سال ۲۰۱۷ آزمایش نشده اند.		این نقص ها می تواند برای استخراج داده ها، اجرای یک درخواست ریموت از سمت سرور، اسکن سیستم های داخلی، انجام حمله اختلال در سرویس و همچنین اجرای سایر حملات استفاده شود. تاثیر کسب و کار بستگی به الزامات حفاظتی همه برنامه های کاربردی متاثر و داده ها دارد.	

آیا برنامه کاربردی آسیب‌پذیر است؟

برنامه های کاربردی و به ویژه سرویس های وب مبتنی بر XML یا ادغام های پایین دست (downstream integrations) ممکن است در شرایط زیر برای حمله آسیب پذیر باشند:

* برنامه XML را به طور مستقیم یا آپلودهای XML را قبول می کند، به ویژه از منابع نامشخص، یا داده های غیر قابل اعتماد را به اسناد XML وارد می کند که سپس توسط یک پردازنده XML پردازش می شود.
* هر یک از پردازنده های XML در نرم افزار یا وب سرویس های مبتنی بر SOAP (document type definition) ها را فعال کرده اند. به عنوان مکانیزم دقیق برای غیرفعال کردن پردازش DTD، بهترین کار استفاده از مرجعی مانند 'OWASP Cheat Sheet 'XXE Prevention است.
* اگر برنامه شما از SAML برای پردازش هویت در خلال امنیت یکپارچه یا اهداف (SSO) استفاده می کند.SAML از XML برای اثبات هویت استفاده **می** کند و ممکن است آسیب پذیر باشد.
* اگر برنامه از SOAP قبل از نسخه ۱,۲ استفاده کند، احتمالا حساس به حملات XXE است اگر موجودیت های XML به چارچوب SOAP منتقل شوند.
* آسیب پذیر بودن به حملات XXE به این معنی است که برنامه به حملات انکار سرویس، از جمله حمله Billion Laughs، آسیب پذیر است.

نحوه پیشگیری از حمله

حداقل موارد زیر را دنبال کنید و مراجع را دنبال کنید:

آموزش برنامه نویس برای شناسایی و مقابله با XXE ضروری است. علاوه بر این، جلوگیری از XXE نیاز دارد:

* هر زمان که امکان دارد، از فرمت های داده ای کمتری مانند JSON استفاده شود و از سریال سازی (تسلسل و ترتیب) اطلاعات حساس جلوگیری شود.
* Patch یا ارتقاء تمامی پردازنده های XML و کتابخانه ها که توسط برنامه کاربردی یا در سیستم عامل اصلی استفاده می شود. از بررسی کننده های وابستگی استفاده کنید. SOAP را به ۱,۲ SOAP یا بالاتر به روز رسانی کنید.
* موجودیت خارجی XML و پردازش DTD در تمام پارسر های XML در برنامه را غیر فعال کنید ، همانطور که در 'OWASP Cheat Sheet 'XXE Prevention غیر فعال است.
* اعتبار سنجی ، فیلتر کردن و sanitization ورودی را در سمت سرور برای جلوگیری از انتقال اطلاعات مخرب در اسناد XML ، هدرها یا گره ها پیاده سازی کنید.
* بررسی کنید که عملکرد آپلود فایل XML یا XSL، ورودی XML را با استفاده از XSD یا مشابه آن، اعتبار سنجی می کند.
* ابزارهای SAST می توانند به شناسایی XXE در کد برنامه کمک کند، اگر چه بررسی دستی کد بهترین گزینه در برنامه های بزرگ و پیچیده است.
* اگر این کنترل ها امکان پذیر نیست، از وصله های مجازی، دروازه های امنیتی API یا فایروال های وب (WAF) برای شناسایی، نظارت و توقف حملات XXE استفاده کنید .

مثال‌هایی از سناریوهای حمله

موارد متعدد XXE به صورت عمومی کشف شده است، از جمله حمله به دستگاه های Embedded. XXE در بسیاری از مکان های غیر منتظره، از جمله وابستگی های (nested dependencies) عمیق رخ می دهد. ساده ترین راه این است که فایل XML مخرب را آپلود کنید، اگر قبول شد:

سناریو شماره ۱: مهاجم تلاش می کند تا داده ها را از سرور استخراج کند:

```
<?"1-8859-encoding="ISO "1.0"=xml version?>
DOCTYPE foo!> [
< ELEMENT foo ANY!>
<[< "ENTITY xxe SYSTEM "file:///etc/passwd!>
<foo>&xxe;</foo>
```

سناریو شماره ۲: مهاجم شبکه خصوصی سرور را با تغییر خط ENTITY بالا به خط زیر، کاوش می کند:

```
<[<"ENTITY xxe SYSTEM "https://192.168.1.1/private !>
```

سناریو شماره ۳: یک مهاجم با استفاده از یک فایل بی پایان، حمله انکار سرویس را انجام می دهد:

```
<[<"ENTITY xxe SYSTEM "file: /// dev / random !>
```

منابع

OWASP

OWASP Risk Rating Methodology

Article on Threat/Risk Modeling

External

ISO 31000: Risk Management Std

ISO 27001: ISMS

NIST Cyber Framework

A5 :2017

نقض کنترل دسترسی

بردارهای حمله		ضعف امنیتی		تاثیرات	
App. Specific	قابلیت بهره برداری :۲	شیوع : ۲	قابلیت تشخیص : ۲	فنی : ۳	تجاری : ؟
بهره برداری از کنترل دسترسی یک مهارت اصلی مهاجمان است. ابزارهای SAST و DAST می توانند فقدان کنترل دسترسی را تشخیص دهند اما اگر کنترل دسترسی وجود داشته باشد، عملکرد آن را نمی توانند تأیید کنند. کنترل دسترسی با استفاده از ابزار دستی و یا احتمالا از طریق خودکار سازی برای فقدان کنترل دسترسی ها در چارچوب های خاص قابل شناسایی است.		ضعف های کنترل دسترسی عموماً به دلیل نقص در تشخیص خودکار و نقص در تست عملکردی مؤثر توسط توسعه دهندگان نرم افزار وجود دارند. تشخیص کنترل دسترسی به طور معمول نمی تواند با توسل به آزمایش ایستا یا پویای خودکار انجام شود. تست دستی بهترین راه برای شناسایی کنترل دسترسی ناکارا یا نبود کنترل دسترسی است، از جمله روش (HTTP GET , POST)، کنترل کننده ها، direct object references و غیره.		تأثیر فنی این است که مهاجمان به عنوان کاربران یا ادمین ها عمل می کنند، یا استفاده کاربران از توابع با دسترسی خاص ، و یا ایجاد، دسترسی، به روز رسانی و یا حذف هر رکورد. تاثیر کسب و کار بستگی به الزامات حفاظتی برنامه و داده ها دارد.	

آیا برنامه کاربردی آسیب‌پذیر است؟

کنترل دسترسی سیاست را به گونه ای اعمال می کند که کاربران نمی توانند خارج از مجوز های مرتبط با خود عمل کنند. شکست ها معمولا به افشای اطلاعات غیر مجاز، اصلاح یا خراب کردن تمام داده ها یا انجام یک کار تجاری در خارج از محدوده کاربر منجر می شود.

آسیب پذیری های رایج کنترل دسترسی عبارتند از:

* دور زدن بررسی های کنترل دسترسی از طریق تغییر URL ، وضعیت برنامه کاربردی ، یا صفحه HTML ، یا با استفاده از یک ابزار سفارشی حمله API .
* اجازه دادن به کلید اصلی برای تعویض شدن با رکورد کاربران دیگر ، اجازه مشاهده یا ویرایش حساب کاربری دیگر.
* بالا بردن امتیاز. کار کردن به عنوان یک کاربر بدون ورود به سیستم و یا کار کردن به عنوان یک مدیر زمانی که به عنوان یک کاربر وارد سیستم شده است.
* دستکاری متا دیتا، مانند بازنویسی یا مداخله با یک JSON Web Token (JWT) ، توکن کنترل دسترسی یا یک کوکی یا فیلد پنهان دستکاری شده برای افزایش امتیاز یا سوء استفاده از باطل سازیJWT
* اشتباه تنظیم CORS اجازه دسترسی غیر مجاز API را می دهد.
* اجبار به مرور صفحات مجاز به عنوان یک کاربر نامعتبر یا صفحات مجاز به عنوان یک کاربر معمولی. دسترسی به API با کنترل دسترسی های منقضی شده برای PUT ، POST و DELETE.

نحوه پیشگیری از حمله

کنترل دسترسی تنها در صورتی که در کد سمت سرور مورد اعتماد یا API بدون سرور اعمال شود، مؤثر است، جایی که مهاجم نمی تواند بررسی کنترل دسترسی یا متا دیتا را تغییر دهد.

* به استثنای منابع عمومی، انکار به طور پیشفرض . (Deny by default)
* یک بار اعمال مکانیسم های کنترل دسترسی و استفاده مجدد از آنها در طول برنامه، از جمله به حداقل رساندن استفاده از CORS.
* کنترل دسترسی باید مالکیت رکوردها را به جای پذیرفتن اینکه کاربر می تواند هر رکوردی را ایجاد، خواندن، به روز رسانی و یا حذف کند، اعمال کند.
* الزامات محدودیت کسب و کار یکتا باید توسط مدل های دامنه اعمال شود.
* غیر فعال کردن فهرست دایرکتوری وب سرور و اطمینان از اینکه متا دیتای فایل (به عنوان مثال git) و فایل های پشتیبان در ریشه های وب موجود نیست.
* شکست های کنترل دسترسی را ثبت کنید، در صورت لزوم به مدیران هشدار دهد (برای مثال، شکست های مکرر).
* بعد از خروج ، JWT Token ها باید روی سرور نا معتبر شود. توسعه دهندگان و کارکنان QA باید کنترل دسترسی عملکردی و تست های یکپارچه سازی را در نظر بگیرند.

مثال‌هایی از سناریوهای حمله

سناریو شماره ۱: برنامه از داده های تأیید نشده در یک تماس SQL که دسترسی به اطلاعات حساب را دارد استفاده می کند:

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

مهاجم به سادگی پارامتر 'acct' را در مرورگر برای ارسال به هر تعداد حساب کاربری مورد نظر تغییر می دهد. اگر به درستی تأیید نشده باشد، مهاجم میتواند به هر حساب کاربر دسترسی پیدا کند.

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

سناریو شماره ۲: مهاجم به سادگی URL های هدف را مرور می کند. حقوق مدیر برای دسترسی به صفحه مدیریت لازم است.

http://example.com/app/getappInfo

http://example.com/app/admin_getappInfo

اگر یک کاربر احراز هویت نشده بتواند به هر کدام از صفحه ها دسترسی داشته باشد، این یک نقص است. اگر غیر مدیر بتواند به صفحه مدیریت دسترسی پیدا کند، این یک نقص است.

منابع

OWASP

OWASP Risk Rating Methodology

Article on Threat/Risk Modeling

External

ISO 31000: Risk Management Std
ISO 27001: ISMS
NIST Cyber Framework

A6 :2017

تنظیمات اشتباه امنیتی

بردارهای حمله		ضعف امنیتی		تاثیرات	
App. Specific	قابلیت بهره برداری :۳	شیوع : ۳	قابلیت تشخیص : ۳	فنی : ۲	تجاری : ؟
مهاجمین خارجی ناشناس و همچنین کاربران مجاز را که ممکن است تلاش کنند تا سیستم را به خطر اندازند در نظر بگیرید. همچنین افراد داخلی را که مایل به انجام اقداماتشان هستند را در نظر بگیرید. مهاجمین به حساب های پیشفرض، صفحات استفاده نشده، نقص های ناخواسته ،فایلها و دایرکتوریهای محافظت نشده دسترسی پیدا میکنند تا دسترسی غیرمجاز به سیستم یا دانش آن را کسب کنند.		تنظیمات اشتباه امنیتی در هر سطحی از یک پشته نرم افزار، ازجمله پلتفرم، سرور وب، سرور برنامه، پایگاه داده، چارچوب و کد سفارشی، ممکن است رخ دهد .توسعه دهندگان و مدیران سیستم باید باهم کار کنند تا اطمینان حاصل شود که کل پشته به درستی پیکربندی شده است. اسکنرهای خودکار برای تشخیص پچ های ازدست رفته، غلط بودن تنظیمات، استفاده از حسابهای پیشفرض، سرویسهای غیرضروری و غیره مفید هستند.		این سیستم، بدون اطلاع شما میتواند به طور کامل به خطر بیفتد. تمام اطلاعات شما میتواند در طول زمان به سرقت رفته و یا تغییر کند. هزینه های بازیابی میتواند گران باشد.	

آیا برنامه کاربردی آسیب‌پذیر است؟

آیا نرم افزار شما از تنظیمات امنیتی مناسب در هر بخشی از برنامه رنج میبرد؟ از جمله:

۱. آیا هیچکدام از برنامه های شما از بین رفته اند؟ این برنامه شامل OS، وب / برنامه سرور ،DBMS، برنامه های کاربردی، API ها، و تمام اجزا و کتابخانه ها میشود.

۲. آیا ویژگیهای غیرضروری فعال یا نصب شده اند (به عنوان مثال، پورتها، سرویسها، صفحات ،حسابها، امتیازات)

۳. آیا حسابهای پیشفرض و گذر واژه های آنها هنوز هم فعال و بدون تغییر هستند؟

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

۵. آیا تنظیمات امنیتی در سرورهای برنامه شما، چارچوب برنامه (مانند ،Struts ،Spring ASP.NET)، کتابخانه ها، پایگاه های داده و غیره برای مقداردهی، امن نشده است؟ بدون انجام پیکربندی امنیتی هماهنگ و قابل تکرار ،سیستمها در معرض خطر بیشتر هستند.

نحوه پیشگیری از حمله

توصیه های اولیه مقرر کردن تمام موارد زیر است:

۱. یک فرایند تکرارپذیر که باعث گسترش سریع و آسان یک محیط دیگر است که کاملا محافظت شده میشود. توسعه ،QA، و محیطهای تولید باید یکسان پیکربندی شده باشند (با کلمه عبور مختلف استفاده شده در هر محیط). این فرآیند باید به صورت خودکار انجام شود تا تلاش کمتری برای راه اندازی یک محیط امنیتی جدید ایجاد شود.

۲. یک فرایند که تمام به روزرسانی ها و پچ های برنامه ی جدید که به طور همزمان به هر محیط مستقرشده، نگهداری و گسترش میدهد. این فرایند باید تمام کامپننت ها و کتابخانه ها را نیز شامل شود.

۳. معماری برنامه ی قوی که جداسازی مؤثر و امن بین کامپنتها را فراهم میکند.

۴. یک فرآیند خودکار برای تأیید اینکه پیکربندی ها و تنظیمات در کلیه محیط ها به درستی پیکربندی شده اند.

مثال‌هایی از سناریوهای حمله

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

سناریو شماره ۲: فهرست دایرکتوری در سرور وب شما غیرفعال نیست. مهاجمین میتوانند به‌سادگی از دایرکتوریها برای یافتن هر فایلی استفاده کنند. مهاجم همه کلاسهای جاوا کامپایل شده شما را پیدا میکند و آنها را دانلود میکند و سپس تجزیه و تحلیل میکند و با استفاده از مهندسی معکوس تمامی کدهای سفارشی خود را از آن دریافت میکند. سپس مهاجم یک نقص کنترل دسترسی خطرناک در برنامه شما پیدا میکند.

سناریو شماره ۳: پیکربندی سرور برنامه اجازه میدهد تا ردیابی پشته ها به کاربران بازگردانده شود ،به‌طور بالقوه معایب اساسی مانند نسخه های چارچوب که شناخته شده اند، آسیب پذیرند.

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

منابع

OWASP

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

External

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

Cross-Site Scripting (XSS)

A7
:2017

بردارهای حمله		ضعف امنیتی		تاثیرات	
App. Specific	قابلیت بهره برداری :۳	شیوع : ۳	قابلیت تشخیص : ۳	فنی : ۲	تجاری : ؟
نقصهای XSS زمانی رخ میدهد که یک برنامه حاوی اطلاعات غیرقابل اعتماد در یک صفحه وب جدید بدون مجوز باشد یا یک صفحه وب موجود را با داده های ارائه شده توسط کاربر با استفاده از یک مرورگر API که میتواند جاوا اسکریپت تولید کند ،به‌روزرسانی کند .XSS به مهاجمین امکان اجرای اسکریپتها در مرورگر قربانی را میدهد که میتواند جلوی کاربر را بگیرد، وب‌سایتها را خراب کند یا کاربر را به سایت‌های مخرب هدایت کند.		XSS زمانی رخ میدهد که برنامه یک صفحه وب را با شیوع اطلاعات کنترل شده مهاجم، بدون بازگشت از آن محتوا یا بدون بسیار شایع استفاده از یک API جاوا اسکریپت امن ،به‌روز میکند. معایب XSS ضعف امنیتی شامل دودسته اصلی: ذخیرهشده و بازتاب شده است و هرکدام از آنها میتوانند بر روی سرور یا کلاینت ایجاد شوند. تشخیص اکثر نقص‌های XSS از طریق تست یا تجزیه و تحلیل کد بسیار راحت است . شناسایی XSS سمت کلاینت بسیار دشوار است.		مهاجمان میتوانند اسکریپتها را در مرورگر قربانی اجرا کنند تا کاربر را مسدود کنند، وب سایتها را از بین ببرند، محتوای خصمانه خود را متعادل وارد کنند، کاربران را هدایت کنند و مرورگر کاربر را با استفاده از برنامه های مخرب بشکنند.

آیا برنامه کاربردی آسیب‌پذیر است؟

شما به XSS سمت سرور آسیبپذیر هستید اگر کد سمت سرور شما از ورودی های ارسال شده توسط کاربر به‌عنوان بخشی از خروجی HTML استفاده کند و شما باید از escape حساس به متن، استفاده کنید تا مطمئن شوید که نمیتواند اجرا شود. اگر یک صفحه وب برای انتقال پویای داده های قابل کنترل مهاجم به یک صفحه، از جاوا اسکریپت استفاده کند آنگاه شما دارای نقص XSS کلاینت هستید. به‌طور ایده‌آل باید از ارسال داده های قابل کنترل مهاجم به API های جاوا اسکریپت ناامن اجتناب کنید، اما استفاده از escape داده های ورودی میتواند شما را امن سازد.

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

نحوه پیشگیری از حمله

پیشگیری از نقص XSS نیاز به جداسازی داده های غیرقابل اعتماد از محتوای مرورگر فعال دارد.

۱. برای پیشگیری از XSS سمت سرور، گزینه اول این است که از داده های نامعلوم در چارچوب HTML (بدنه، ویژگی، جاوا اسکریپت ،CSS یا URL) که داده ها در آن قرار داده میشوند، استفاده نکنید .

برای اطلاعات بیشتر در این زمینه OWASP XSS Prevention Cheat Sheet را ببینید.

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

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

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

مثال‌هایی از سناریوهای حمله

برنامه بدون تائید یا escape، از داده های غیرقابل اعتماد در ساخت قطعه HTML زیر استفاده میکند:

```
(String) page =+ "input name='creditcard' type='TEXT'>"
+ "'value="
```

مهاجم پارامتر CC" را در مرورگر خود تغییر میدهد به "request.getParameter("CC") + "'<"; :

```
'<foo='+document.cookie</script
```

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

توجه داشته باشید که مهاجمان میتوانند از XSS برای جلوگیری از هرگونه دفاع خودکار CSRF استفاده کنند. جزئیات بیشتر در مورد CSRF در A8-۲۰۱۷ آمده است.

منابع

OWASP

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

External

OWASP XSS Filter Evasion Cheat Sheet
CWE Entry 79 on Cross-Site Scripting

A8 :2017

Insecure Deserialization

بردارهای حمله		ضعف امنیتی		تاثیرات	
App. Specific	قابلیت بهره برداری : ۱	شیوع : ۲	قابلیت تشخیص : ۲	فنی : ۳	تجاری : ؟
بهره برداری از diserialization تا حدودی دشوار است.		این مسئله در Top 10 بر اساس تحقیقات صنعت و نه بر روی داده های قابل اندازه گیری وجود دارد. برخی از ابزارها می توانند نقاط ضعف را بیابند، اما تست های انسانی اغلب برای اعتبارسنجی مشکل ضروری است. انتظار می رود که داده های شایع برای نقص های خنثی سازی افزایش یابد، زیرا ابزار توسعه برای کمک به شناسایی و رفع آن کمک می کند.		تاثیر ضعفهای Deserialization را نمی توان نادیده گرفت. این نقص ها می تواند به حملات کد های راه دور منجر شود، که یکی از جدی ترین حملات ممکن است. تاثیر تجاری به نیازهای حفاظت از برنامه و داده ها بستگی دارد.	

آیا برنامه کاربردی آسیب‌پذیر است؟

برنامه های کاربردی و API ها آسیب پذیر خواهند بود اگر آنها از اشیاء متخاصم یا دزدیده شده توسط مهاجم استفاده کنند. این می تواند به دو نوع اصلی حملات منجر شود:

• حملات مرتبط با ساختار داده ها و جایی که مهاجم منطق برنامه را تغییر می دهد یا اگر اشیاء موجود در برنامه وجود داشته باشد که می توانند رفتار را در طی یا پس از پایان دادن به کار تغییر دهند، اجرای کد دلخواه را اجرا می کند.

• داده ها را دستکاری می کند، مانند حملات مرتبط با کنترل دسترسی، که در آن ساختار داده های موجود استفاده می شود اما محتوا تغییر می کند.

Serialization ممکن است در برنامه های کاربردی برای:

• ارتباط از راه دور و اینترفیس (RPC / IPC)

• پروتکل های سیم، خدمات وب، کارگزاران پیام

• ذخیره سازی / پایداری

• پایگاه های داده، سرورهای ذخیره سازی، سیستم های فایل

• کوکی HTTP، پارامترهای فرم HTML، نشانه های تأیید API

استفاده شود.

نحوه پیشگیری از حمله

تنها الگوی معماری امن، تشخیص اشیاء سریالی از منابع نامشخص است یا استفاده از رسانه های سریالی که تنها نوع داده های اولیه را اجازه می دهد. اگر این امکان پذیر نیست، یکی از موارد زیر را اعمال کنید:

• پیاده سازی چک های یکپارچه مانند امضاهای دیجیتالی بر روی هر شیء سریالی برای جلوگیری از ایجاد شی خصمانه یا جاسوسی داده ها.

• اجرای محدودیت های سخت نوع در deserialization قبل از ایجاد شی به عنوان کد.

• عدم نمایش موارد استثنا و خرابی ها، از جمله اینکه کدام نوع ورودی نوع مورد انتظار نیست، یا عدم تمدید.

• محدود کردن یا نظارت بر اتصال به شبکه های ورودی و خروجی.

• نظارت بر عدم وجود هشدار.

مثال‌هایی از سناریوهای حمله

سناریو شماره ۱: برنامه React یک مجموعه از سرویس های مایکروسافت Spring Boot را فراخوانی می کند. برنامه نویسان سعی کردند اطمینان حاصل کنند که کد آنها تغییرناپذیر است. راه حل هایی که با آن روبرو شدیم، سریالی است و هر درخواست را به عقب و جلو منتقل می کند. یک مهاجم به امضای شی "R00" اشاره می کند و از ابزار Java Serial Killer برای به دست آوردن اجرای کدهای راه دور در سرور برنامه استفاده می کند.

سناریو شماره ۲: یک انجمن PHP از پیاده سازی شیء برای ذخیره یک کوکی، حاوی شناسه کاربری کاربر، نقش، هش رمز عبور و حالت های دیگر استفاده می کند:

```
;"user":4:s;2:Mallory";i":7:s;1:i;132:i;0:i}:4:a
{;"b6a8b3bea87fe0e05022f8f3c88bc960":32:s;3:i
```

یک مهاجم شیء سریالی را برای دادن امتیازات مدیریت خود تغییر می دهد:

```
;"admin":5:s;2:Alice";i":5:s;1:i;1:i;0:i}:4:a
{;"b6a8b3bea87fe0e05022f8f3c88bc960":32:s;3:i
```

منابع

OWASP

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

External

OWASP XSS Filter Evasion Cheat Sheet
CWE Entry 79 on Cross-Site Scripting

A9 :2017

استفاده از مولفه های با آسیب پذیری های شناخته شده

بردارهای حمله		ضعف امنیتی		تاثیرات	
App. Specific	قابلیت بهره برداری :۲	شیوع : ۲	قابلیت تشخیص : ۲	فنی : ۳	تجاری : ؟
در حالی که پیدا کردن اکسپلویت های نوشته شده برای بسیاری از آسیب‌پذیری‌های شناخته‌شده بسیار آسان است اما آسیب‌پذیری‌های دیگر بدنبال ایجاد اکسپلویت سفارشی هستند.		بسیاری از نرم افزارها این مسائل را دارند، چون گروه های توسعه دهنده ی آنها به بروزبودن مولفه و کتابخانه ها توجه نکرده اند. در برخی موارد، حتی تمام مولفه های مورداستفاده خود را نمیدانند، و هرگز به نسخه های آنها توجه نمی کنند برخی ابزارهانظیر retire.js به شناسایی کمک می کنند ولی تعیین توان اکسپلویت زدن نیازمند تلاش بیشتر است.		در حالی که برخی از آسیب‌پذیری‌های شناخته‌شده تنها به اثرات جزئی منتهی می‌شوند ، برخی از بزرگ‌ترین رخنه ها تا به امروز بر بهره‌برداری ازآسیب‌پذیری‌های شناخته‌شده در قطعات تکیه کرده‌اند. بسته به دارایی‌هایی که از آن محافظت می‌شود ، شاید این ریسک باید در بالای لیست باشد.	

آیا برنامه کاربردی آسیب‌پذیر است؟

شما احتمالا آسیب پذیر هستید:

• اگر نسخه های تمام مولفه هایی که از آن‌ها استفاده می‌کنید را ندانید (هر دو طرف کلاینت و سرور) این شامل اجزایی است که مستقیما از وابستگی‌های تو در تو (توزیع شده) استفاده می‌کنند.

• اگر برنامه ، آسیب‌پذیر یا پشتیبانی نشده بوده یا بروز نباشد این شامل سیستم‌عامل، وب اپلیکیشن یا سرور، سیستم مدیریت پایگاه‌داده،برنامه ها، API ها و همه اجزا، محیطهای زمان اجرا و کتابخانه ها می‌شود.

• اگر به طور منظم آسیب پذیری ها را اسکن نکنید وامنیت بخش هایی که از آنها استفاده می کنید را کنترل نکنید

• اگر پلت فرم ، چارچوب‌ها و وابستگی‌های موجود را در یک مد مبتنی بر ریسک تنظیم و یا ارتقا ندهید این امر معمولا در محیطهای زمانی رخ می‌دهد که وصله کردن یک وظیفه ماهانه یا فصلی است که سازمان‌ها را چندین روز یا ماه ها برای مواجه ه غیر ضروری برای رفع آسیب‌پذیری‌ها مشغول می سازد.

• اگر برنامه نویسان ، سازگاری کتابخانه های بروز شده، ارتقا یافته و یا پچ را تست نکنند.

• اگر پیکربندی مولفه ها را امن نکنید.

نحوه پیشگیری از حمله

باید یک فرآیند مدیریت وصله وجود داشته باشد:

• حذف وابستگی های استفاده نشده، ویژگی های غیر ضروری، اجزاء، فایل ها و اسناد

• مرتب کردن نسخه های هر دو طرف کلاینت و سرور (به عنوان مثال چارچوب ، کتابخانه ها) و وابستگی‌های آن‌ها با استفاده از ابزارهایی مانند retire.js ، DependencyCheck ، versions و غیره. به طور مرتب بر منابعی مانند CVE و NVD برای آسیب پذیری های اجزا نظارت شود، از ابزارهای تجزیه و تحلیل برنامه برای بهینه سازی فرآیند استفاده شود، به هشدارهای ایمیل برای آسیب‌پذیری‌های امنیتی مربوط به اجزای مورد استفاده توجه شود.

• مولفه ها فقط از منابع رسمی بر روی لینک های ایمن دریافت شود. بسته های امضا شده برای کاهش یک جزء مخرب اصلاح شده ترجیح داده شود.

• نظارت بر کتابخانه ها و اجزایی که پشتیبانی نشده یا وصله های امنیتی برای نسخه های قدیمی‌تر ندارند. اگر وصله کردن امکان پذیر نیست ، استفاده از یک قطعه مجازی برای نظارت ، تشخیص یا حفاظت در برابر مساله کشف‌شده

هر سازمان باید اطمینان حاصل کند که یک برنامه مداوم برای نظارت و استفاده از به روزرسانی و یا تغییرات پیکربندی برای طول عمر برنامه یا نمونه کارها وجود دارد.

مثال‌هایی از سناریوهای حمله

سناریو شماره ۱: اجزایا مولفه ها تقریبا همیشه با دسترسی کامل از برنامه اجرا میشوند، بنابراین نقص در هر جزء می تواندمنجر به تأثیر جدی شود چنین نقصی می تواند تصادفی)مثلا خطای برنامه نویسی(یا عمدی باشد برخی ازموارد آسیب پذیری جزء که مورد سوءاستفاده قرار می گیرند شامل:

• CVE-۲۰۱۷-۵۶۳۸: یک آسیب‌پذیری با قابلیت کنترل از راه دور که اجرای کد دلخواه بر روی سرور را امکان پذیر می‌سازد ، به خاطر رخنه قابل‌توجه مورد نقد قرار گرفته‌است.

• در حالی که وصله کردن در اینترنت اشیا (IoT) اغلب دشوار یا غیر ممکن است، اهمیت وصله کردن آن‌ها می‌تواند بسیار قابل توجه باشد (به عنوان مثال وسایل پزشکی).

ابزارهای خودکار برای کمک به مهاجمان برای یافتن سیستم‌های وصله نشده یا با پیکربندی اشتباه وجود دارد به عنوان مثال موتور جستجوی Shodan IoT می‌تواند به شما کمک کند تا دستگاه هایی را پیدا کنید که هنوز از آسیب Heartbleed که در آوریل ۲۰۱۴ تعمیر شد، رنج می‌برند .

منابع

OWASP

OWASP Application Security Verification Standard: V1 Architecture design and threat modelling

(OWASP Dependency Check (for Java and NET libraries

(OWASP Testing Guide: Map Application Architecture (OTGINFO-010

OWASP Virtual Patching Best Practices

The Unfortunate Reality of Insecure Libraries

MITRE Common Vulnerabilities and Exposures (CVE) search

(National Vulnerability Database (NVD

Retirejs for detecting known vulnerable JavaScript libraries

Node Libraries Security Advisories

Ruby Libraries Security Advisory Database and Tools

A10 :2017

نظارت و ثبت رویداد ناکافی

بردارهای حمله		ضعف امنیتی		تاثیرات	
App. Specific	قابلیت بهره برداری :۲	شیوع : ۲	قابلیت تشخیص : ۲	فنی : ۳	تجاری : ؟
اکسپلویت از نظارت و ثبت رویداد ناکافی تقریبا بستر اصلی هر اتفاق مهم است مهاجمان به عدم نظارت و پاسخ به موقع ،برای رسیدن به اهداف خود بدون شناسایی شدن متکی هستند.		این موضوع در جدول TOP10 براساس یک بررسی صنعتی گنجانده شده‌است. یک استراتژی برای تعیین اینکه آیا شما نظارت کافی دارید، بررسی مجدد رویدادهای مربوط به تست نفوذ است اقدامات تست کنندگان باید به اندازه کافی ثبت شود تا بدانند که چه آسیبی به آنها وارد شده است.		اکثر حملات موفق با کاوش آسیب‌پذیری آغاز می‌شوند. اجازه دادن به این کاوشگرها می‌تواند احتمال اکسپلویت موفق را تقریبا تا ۱۰۰ درصد افزایش دهد. در سال ۲۰۱۶ ، شناسایی یک شکاف (رخنه) به طور متوسط ۱۹۱ روز(زمان زیاد برای آسیب زدن) طول کشید.	

آیا برنامه کاربردی آسیب‌پذیر است؟

ثبت رویداد ، تشخیص، نظارت و پاسخ فعال ناکافی در هر زمان رخ می‌دهد:

- رویدادهای قابل بررسی، مانند ورود به سیستم ، ورود ناموفق به سیستم و تراکنش های با ارزش بالا در سیستم ثبت نشده اند.
- هشدارها و اشتباهات موجب ایجاد پیام های رویداد مشخص یا کافی نمی شود.
- رویداد برنامه ها و API ها برای فعالیت مشکوک پایش نمی‌شوند.
- رویدادها فقط بصورت محلی ثبت شده اند.
- آستانه های مربوط به هشدار و فرآیندهای تشدید پاسخ، مناسب و یا موثر نیستند.
- تست نفوذ و اسکن توسط ابزارهای DAST مانند(OWASP ZAP) باعث هشدار نمی شود.
- برنامه قادر به تشخیص، تشدید شدن یا هشدار برای حملات فعال در زمان واقعی یا نزدیک به زمان واقعی نیست.

شما به علت نشت اطلاعات آسیب‌پذیر هستید، اگر لاگ وقایع و هشدار ها قابل مشاهده برای یک کاربر و یا یک فرد باشد.

نحوه پیشگیری از حمله

همچنین در مورد خطر اطلاعات ذخیره شده یا پردازش شده توسط نرم افزار داریم:

- اطمینان حاصل کنید که تمام ورودی ها به سیستم، خطاهای کنترل دسترسی و اعتبار سنجی ورودی طرف سرور را می توان با زمینه کاربری کافی برای شناسایی حساب‌های مشکوک یا نادرست ثبت کرد و زمان کافی را برای اجازه دادن به تجزیه و تحلیل قانونی به تاخیر انداخت.
- اطمینان حاصل کنید که رویدادها در یک قالبی تولید می شود که می تواند به راحتی توسط یک راه حل مدیریت رویداد متمرکز مورد استفاده قرار گیرد.
- اطمینان از اینکه تراکنش های با ارزش بالا دارای یک دنباله حسابرسی با کنترل های یکپارچه برای جلوگیری از دستکاری یا حذف، مانند جداول پایگاه داده اضافه یا مشابه باشند.
- ایجاد نظارت مؤثر و هشدار به طوری که فعالیت های مشکوک شناسایی و به موقع پاسخ داده شوند.
- ایجاد و یا اتخاذ یک پاسخ تصادفی و برنامه ریکاوری، مانند NIST ۸۰۰-۶۱ rev ۲ یا بالاتر.

چارچوب های حفاظت از برنامه های کاربردی متن باز و تجاری مانند OWASP AppSensor ، فایروال های وب مانند ModSecurity with the OWASP ModSecurity Core Rule Set و برنامه همبستگی log با داشبورد های سفارشی و هشدار دهنده وجود دارد.

مثال‌هایی از سناریوهای حمله

سناریو ۱:یک نرم افزار انجمن منبع باز که توسط یک تیم کوچک اجرا می شد با استفاده از نقص در نرم افزار آن هک شد. مهاجمان موفق به از بین بردن منبع کد داخلی حاوی نسخه بعدی و تمامی محتویات انجمن شدند اگرچه این منبع کد را می توان بازیابی کرد ، اما فقدان نظارت ، ثبت رویداد و یا هشدار دادن منجر به نقض بسیار بدتری شد. پروژه برنامه ی انجمن در نتیجه این موضوع، دیگر فعال نیست .

سناریو ۲: یک مهاجم کاربران را با استفاده از گذر واژه معمولی اسکن می‌کند آن‌ها می‌توانند تمام حساب‌ها را با استفاده از این گذرواژه در اختیار بگیرند. برای کاربران دیگر ، این اسکن فقط یک ورود(login) ناموفق به نظر می رسد. پس از چند روز ، این کار ممکن است با گذرواژه متفاوت تکرار شود.

سناریو ۳:گفته می‌شود که یک خرده‌فروش بزرگ آمریکایی تحلیل داخلی بدافزار را تجزیه و تحلیل می‌کند برنامه sandbox به طور بالقوه برنامه ناخواسته را شناسایی کرده بود ، اما هیچ‌کس به این کشف واکنش نشان نداد. قبل از این که نفوذ به دلیل تراکنش های کارت اعتباری توسط یک بانک خارجی شناسایی شود ، sandbox چندین مرتبه هشدار داده بود.

منابع

OWASP

- OWASP Proactive Controls: Implement Logging and Intrusion Detection
- OWASP Application Security Verification Standard: V8 Logging and Monitoring
- OWASP Testing Guide: Testing for Detailed Error Code
- OWASP Cheat Sheet: Logging

External

- Omission of Security-relevant Information :223-CWE

Insufficient Logging :778-CWE

+D چه چیز دیگری برای توسعه دهندگان وجود دارد

ایجاد و استفاده از فرایندهای امنیت قابل تکرار و کنترل‌های امنیتی استاندارد

موضوع امنیت برنامه وب چه اینکه موضوع جدیدی باشد وچه اینکه از قبل با این ریسک‌ها آشنا باشید، تولید یک برنامه و بی امن و یا تعمیر یک برنامه موجود می‌تواند مشکل باشد. اگر شما می‌خواهید یک برنامه بزرگ را مدیریت کنید ، این کار می‌تواند دلهره‌آور باشد.

برای کمک به سازمان‌ها و توسعه دهندگان که ریسک‌های امنیتی برنامه خود را به شیوه‌ای مقرون‌به‌صرفه کاهش دهند، OWASP منابع رایگان و آزاد فراوانی را تولید کرده‌است که می‌توان از آن برای پرداختن به امنیت برنامه در سازمان خود استفاده کرد. در ادامه OWASP منابع متعددی برای کمک به سازمان‌ها تولید کرده که بتوانند برنامه های تحت وب و API های امن تولیدکنند. در صفحه بعد منابع OWASP دیگری ارائه می‌شود که می‌توانند به سازمان‌ها در بررسی امنیت برنامه ها وAPI های خود کمک کنند.

نیازمندیهای امنیتی برنامه

برای ایجاد یک برنامه تحت وب امن ،باید مشخص کرد که چه ابزار امنی برای آن برنامه وجود دارد. OWASP توصیه می‌کند که از استاندارد تایید امنیت برنامه (OWASP ASVS) به عنوان راهنما برای تنظیم نیازمندی‌های امنیتی برنامه استفاده شود. اگر قرار است که برون سپاری شود، می بایست قرارداد امنیت نرم افزار OWASP ضمیمه گردد. . نکته : ضمیمه، قانون قراردادی آمریکا است، بنابراین قبل از استفاده از ضمیمه نمونه،می بایست با مشاوره حقوقی با صلاحیت مشورت کرد

معماری امنیتی برنامه

به جای ارتقاء امنیت برنامه ها و API ها، طراحی امنیتی از ابتدا بسیار مؤثرتر است. OWASP Prevention Cheat ، OWASP Sheets را به عنوان نقطه شروع خوب برای راهنمایی در مورد چگونگی طراحی امنیت از ابتدا توصیه می کند.

کنترل‌های امنیتی استاندارد

دستیابی به کنترل‌های امنیتی قوی و قابل‌استفاده دشوار است . با استفاده از مجموعه‌ای از کنترل‌های امنیتی استاندارد، توسعه برنامه ها و API های امن تسهیل می شود . کنترل‌های فعال OWASP Proactive Controls یک نقطه شروع خوب برای توسعه دهندگان است و بسیاری از چارچوب‌های مدرن اکنون با کنترل‌های امنیتی استاندارد و موثر برای مجوز، اعتبار سنجی ، پیش‌گیری

چرخه توسعه امن

OWASP برای بهبود فرآیند سازمان شما برای ساخت برنامه ها و API ها، مدل بلوغ تضمین برنامه (SAMM) را پیشنهاد می‌کند . این مدل به سازمان‌ها کمک می‌کند تا یک استراتژی برای امنیت برنامه تدوین و پیاده‌سازی کنند که متناسب با ریسک‌های خاص آن سازمان هاست .

آموزش امنیت برنامه

پروژه آموزش OWASP برای کمک به توسعه دهندگان در زمینه امنیت برنامه های وب مطالب آموزشی فراهم می‌کند. برای یادگیری در مورد آسیب‌پذیری ، OWASP NodeJS ، WebGoat.NET ، OWASP WebGoat ، پروژه OWASP Juice Shop و یا پروژه برنامه های وب شکسته OWASP را امتحان کنید. برای درجریان بودن(بروز بودن) به Conference OWASP AppSec ، آموزش کنفرانس OWASP یا OWASP Chapter meeting مراجعه شود.

منابع OWASP متعددی وجود دارند که برای استفاده در دسترس هستند . صفحه پروژه OWASP قابل بازدید می باشد که همه Flagship ، آزمایشگاه و پروژه های Incubator را در فهرست پروژه OWASP لیست می‌کند. اکثر منابع OWASP در ویکی در دسترس هستند و بسیاری از اسناد OWASP را می توان در hardcopy یا به عنوان eBooks سفارش داد .

+T دیگر چه چیزی برای تست کننده های امنیتی وجود دارد؟

ایجاد تست امنیتی مداوم برنامه

ساختن کد به صورت امن مهم است، اما مهم تراینکه اطمینان حاصل شود که امنیتی که برای ساخت درنظر گرفته شده ، واقعا وجود دارد ، به درستی اجرا شده و در هر جایی که قرار است باشد استفاده می‌شود. هدف از آزمون امنیت برنامه ، ارائه این گواهی است. این کار دشوار و پیچیده است و فرآیندهای پیشرفته توسعه با سرعت بالا مانند Agile و DevOps فشار بسیار زیادی را بر روش‌ها و ابزارهای سنتی گذاشته اند. بنابراین ما به شدت شما را تشویق می‌کنیم تا به این فکر کنید که چطور بر روی آنچه در کل برنامه تان مهم است تمرکز کنید و این کار را به طور موثر انجام دهید.

ریسک‌های مدرن به سرعت حرکت می‌کنند، بنابراین روزها اسکن یا تست نفوذ برای آسیب‌پذیری یک برنامه سالی یکبار و یا به همین ترتیب زمانی طولانی سپری می‌شود. توسعه برنامه مدرن نیازمند تست پیوسته امنیت برنامه در سراسر چرخه عمر توسعه برنامه است. راه تسهیل خطوط توسعه موجود ،استفاده ازخودکارسازی امنیتی است که توسعه را کند نمی‌کند. هر روشی که انتخاب می کنید، هزینه سالیانه را برای تست، ترمیم، بازآموزی و بازنشانی یک برنامه تکمیلی، ضرب در اندازه برنامه در نظر بگیرید.

درک مدل تهدید

قبل از اینکه شروع به آزمایش کنید ، سعی کنید بفهمید چه چیزی مهم است که زمان روی آن صرف کنید . اولویت‌ها از مدل تهدید ناشی می‌شوند ، بنابراین اگر یکی را ندارید، نیاز است که قبل از آزمایش آن را ایجاد کنید . استفاده از OWASP ASVS و راهنمای آزمایش OWASP به عنوان یک ورودی را در نظر بگیرید و به صرف فروشندگان ابزار برای تصمیم‌گیری درباره اینکه چه چیز برای کسب‌وکار شما مهم است، اتکا نکنید.

درک SDLC شما

رویکرد شما برای تست امنیت برنامه باید به شدت با افراد ، فرآیندها ، و ابزارهایی که در چرخه حیات توسعه برنامه (SDLC) استفاده می‌کنید ، سازگار باشد. تلاش برای اعمال گام‌های بیشتر ، گیت ها و بررسی ها به احتمال زیاد باعث ایجاد اصطکاک، دورزدن و درگیری می‌شود. به دنبال فرصت‌های طبیعی برای جمع‌آوری اطلاعات امنیتی و بازخورد آن به برنامه خودباشید.

استراتژی‌های آزمایشی

ساده‌ترین و سریع‌ترین و دقیق‌ترین روش را انتخاب کنید تا هر الزام را تایید کنید . چارچوب دانش امنیتی OWASP و استاندارد تایید امنیت برنامه می‌توانند منابع خوبی از الزامات امنیتی عملیاتی و غیر عملیاتی در واحدتان و تست یکپارچگی باشند . اطمینان حاصل کنید که منابع انسانی مورد نیاز برای مقابله با مثبت های کاذب استفاده از ابزارهای خودکار و همچنین خطرات جدی منفی های کاذب در نظر گرفته شده است.

دستیابی به پوشش و دقت

نیازی نیست که همه چیز را آزمایش کنید . بر روی آنچه مهم است تمرکز کرده و برنامه بررسی خود را در طول زمان گسترش دهید . این به معنی گسترش دفاع امنیتی وریسک هاست که به طور خودکار بررسی شده و بخوبی گسترش مجموعه‌ای از برنامه ها و API ها پوشش داده شوند . هدف دستیابی به وضعیتی است که در آن امنیت لازم تمام برنامه ها و API ها به طور مداوم بررسی می‌شود .

یافته های خود رابیان کنید

مهم نیست که در حال آزمایش چقدر خوب هستید ، فرقی نمی‌کند مگر این که به طور موثر با آن ارتباط برقرار کنید . با نشان دادن نحوه کارکردن برنامه اعتماد سازی کنید . توضیح دهید که چگونه می‌تواند بدون " زبان " مورد سوء استفاده قرارگیرد و شامل یک سناریوی حمله برای واقعی ساختن آن می باشد . برآوردی واقع بینانه از اینکه کشف و بهره برداری آسیب پذیری چقدر سخت است و میزان بد بودن آن داشته باشید . در نهایت یافته های تیم توسعه ابزارهادر حال استفاده بوده وPDF نیست.

چه چیزدیگری برای سازمان ها وجود دارد

حالا فرآیند امنیت برنامه خود را شروع کنید

امنیت برنامه دیگراختیاری نیست. بین افزایش حملات وفشارهای تنظیمی یانظارتی، سازمان‌ها باید فرایندهای موثر وقابلیت‌هایی برای ایمن‌سازی برنامه هاو API ها ایجادکنند.با توجه به مقدار سرسام‌آور کد دربرنامه های متعدد وAPIهادرحال حاضردرتولید،بسیاری از سازمان‌ها در تلاش هستند تادسته عظیمی ازآسیب‌پذیری رابه دست آورند.

Owasp به سازمان‌هاتوصیه می‌کندکه یک برنامه امنیتی کاربردی برای کسب شناخت و بهبودامنیت در برنامه ها و API ها ایجادکنند. دستیابی به امنیت برنامه نیازمندبسیاری از بخش‌های مختلف یک سازمان است ،از جمله امنیت وحسابرسی،توسعه برنامه ،تجارت ومدیریت اجرایی که به طور موثرمی بایست بایکدیگرهم‌کاری کنند. امنیت بایدقابل‌مشاهده واندازه‌گیری باشد، به طوری که تمام عوامل مختلف بتوانندوضعیت امنیتی برنامه سازمان رادیده و درک کنند. تمرکز بر فعالیت‌ها ونتایجی که در واقع به بهبودامنیت شرکت باحذف یاکاهش ریسک کمک می‌کنند.

SAMM OWASP و OWASP Application Security Guide for CISOs (راهنمای امنیت برنامهOWASP برایCISOs)، منبع اکثرفعالیت‌های کلیدی دراین فهرست است.

شروع

- تمام برنامه های کاربردی و دارایی‌های مرتبط باآن را مستند کنید. سازمان‌های بزرگ‌تر باید اجرای پایگاه داده مدیریت پیکربندی را برای این منظور در نظر بگیرند.
- ایجاد برنامه امنیت برنامه و راه‌اندازی سلسله مراتب.
- قابلیت تجزیه و تحلیل شکاف توانمندی سازمان خود را با همتایان خود مقایسه تاحوزه های بهبود کلیدی و یک برنامه اجرایی را تعریف کنید.
- کسب تاییدیه مدیریت و ایجاد یک کمپین آگاهی امنیتی کاربردی برای IT کل سازمان

رویکرد دارایی مبتنی برریسک

- شناسایی نیازهای حفاظت از دارایی برنامه تان از دیدگاه کسب وکار. این باید بخشی از قوانین حریم خصوصی وسایر مقررات مربوط به دارایی ها باشد.
- یک مدل رتبه بندی ریسک را با مجموعه ای از عوامل مؤثر و تأثیرگذار بر تحمل پذیری ریسک سازمان ایجاد کنید.
- بر این اساس تمام برنامه ها وAPI هایخود را اولویت بندی و نتایج را به CMDB خود اضافه کنید.
- ایجاد دستورالعمل‌های تضمین برای تعریف درست پوشش وسطح شدت مورد نیاز.

ایجاد یک پایه واساس قوی

ایجاد مجموعه‌ای از سیاست‌ها و استانداردهایی که اساس امنیت برنامه را برای همه ی تیم‌های توسعه مشخص می‌کنند تا بر اساس آن عمل کنند.

تعریف یک مجموعه مشترک ازکنترل‌های امنیتی قابل‌استفاده برای تکمیل این سیاست‌ها و استانداردها و ارائه راهنمایی‌های طراحی و توسعه در مورد استفاده از آن‌ها

ایجاد یک برنامه آموزشی امنیتی کاربردی که مورد نیاز بوده و نقش‌ها و موضوعات مختلف توسعه را هدف قرار می دهد.

ادغام امنیت درفرآیندهای موجود

تعریف و ادغام پیاده‌سازی امن و فعالیت‌های بررسی در توسعه موجود و فرآیندهای عملیاتی. فعالیت‌ها شامل مدل‌سازی تهدید، طراحی امن و مرور طراحی،کد گذاری امن و مرورکد، تست نفوذ و اصلاح هستند.

تامین کارشناسان موضوعی و پشتیبانی از سرویس های توسعه و تیم پروژه برای موفقیت.

ارائه دید مدیریتی

مدیریت با معیارها. تصمیمات سرمایه‌گذاری براساس معیارها و داده های آنالیزشده را پیشرفت دهید. معیارها عبارتند از پایبندی به اقدامات امنیتی وفعالیت‌ها ، آسیب‌های معرفی شده ،آسیب‌پذیری، پوشش برنامه، تراکم نقص توسط نوع و تعداد نمونه و غیره.

تجزیه و تحلیل داده ها از اجرا و بررسی فعالیت‌ها برای جستجوی علل ریشه‌ای و الگوهای آسیب‌پذیری درجهت پیشبرد اهداف راهبردی و سیستمی در سراسر سازمان صورت می گیرد. ازاشتباهات یاد بگیرید و مشوق‌های مثبت برای ارتقاء پیشرفت ارائه دهید.

چه چیزدیگری برای مدیران برنامه وجود دارد

مدیریت چرخه کامل برنامه

برنامه ها به پیچیده ترین سیستم هاتعلق دارند که به طور مرتب ایجاد و حفظ می شوند. مدیریت فناوری اطلاعات برای یک برنامه باید توسط متخصصان فناوری اطلاعات انجام شود که مسئولیت کل چرخه عمر فناوری اطلاعات یک برنامه را دارند. پیشنهاد می شود نقش مدیر برنامه به عنوان متخصص فنی مالک نرم افزار تعیین گردد. مدیر برنامه مسئول کل چرخه عمر برنامه از دیدگاه فناوری اطلاعات است، از جمع آوری نیازمندی ها تا روند سیستم بازنشستگی، که اغلب نادیده گرفته می شود.

نیازمندی ها و مدیریت منابع

- جمع‌آوری ومذاکره درباره نیازمندی های کسب‌وکاربرای یک برنامه شامل:دارایی‌های نیازمندحفاظت وکسب وکارمورد انتظاردررابطه با محرمانگی ،اصالت،یکپارچگی ودر دسترس بودن همه داده ها
- نیازمندی های فنی شامل نیازمندی‌های امنیتی عملیاتی وغیرعملیاتی را گردآوری کنید.
- برنامه ریزی و مذاکره بودجه که شامل تمام جنبه های طراحی، ساخت، آزمایش و عملیات، از جمله فعالیت های امنیتی است.

درخواست پیشنهادها و قرارداد(RFP)

- شرایط لازم را با توسعه دهندگان داخلی یا خارجی، از جمله دستورالعمل ها و الزامات امنیتی مربوط به برنامه امنیتی خود، به عنوان مثال SDLC، مذاکره کنید.
- ارزیابی اجرای تمام الزامات فنی، از جمله مرحله برنامه ریزی و طراحی
- مذاکره همه شرایط فنی، از جمله طراحی، امنیت، و توافق نامه های سطح سرویس (SLA)
- اتخاذ قالب ها و چک لیست ها، مانند OWASP Secure Software Contract Annex.

توجه: این ضمیمه برای قانون قراردادی ایالات متحده است، بنابراین قبل از استفاده از ضمیمه نمونه لطفا مشاوره حقوقی ازمشاور واجد شرایط دریافت کنید.

برنامه ریزی و طراحی

- برنامه ریزی و طراحی را با توسعه دهندگان و سهامداران داخلی مذاکره کنید، برای مثال متخصصان امنیتی.
- معماری امنیتی، کنترل ها و اقدامات متقابل مناسب برای نیازهای حفاظت و سطح تهدید پیش بینی شده را تعریف کنید. این باید توسط متخصصان امنیتی پشتیبانی شود.
- اطمینان حاصل کنید که مالک نرم افزار ریسک های باقیمانده را دریافت می کند یا منابع بیشتری را فراهم می کند.
- در هر سرعت، از ایجاد داستان های امنیتی که شامل محدودیت های اضافه شده برای نیازمندی های غیر عملیاتی است، مطمئن شوید.

استقرار، تست و راه اندازی

- راه اندازی خودکار امن برنامه، رابط ها و تمام اجزای مورد نیاز از جمله مجوزهای لازم
- تست عملکرد فنی و یکپارچه سازی با معماری فناوری اطلاعات و هماهنگ سازی آزمون های کسب و کار
- ایجاد موارد "استفاده" و "سوء استفاده" از دیدگاه های فنی و تجاری
- مدیریت تست های امنیتی بر اساس فرآیندهای داخلی، نیازهای حفاظت، و سطح تهدید فرض شده توسط برنامه
- قرار دادن برنامه درحالت عملیاتی و مهاجرت از برنامه های قبلی در صورت لزوم
- نهایی کردن تمام مستندات، شامل پایگاه اطلاعات مدیریت تغییر (CMDB) و معماری امنیتی

عملیات و مدیریت تغییر

- عملیات باید شامل دستورالعمل هایی برای مدیریت امنیت برنامه (مانند مدیریت پچ) باشد.
- افزایش آگاهی امنیتی از کاربران و مدیریت اختلالات در مورد قابلیت استفاده در مقابل امنیت.
- برنامه ریزی و مدیریت تغییرات، به عنوان مثال مهاجرت به نسخه های جدید برنامه یا اجزای دیگر مانند سیستم عامل، میان افزار و کتابخانه ها.
- به روزرسانی تمام اسناد و مدارک، از جمله در CMDB و معماری امنیتی، کنترل ها و اقدامات متقابل نظیر هرگونه مستندات پروژه.

سیستم های کناره گیر

- هر اطلاعات مورد نیاز باید بایگانی شود. تمام داده های دیگر باید بصورت امن پاک شود. به صورت کاملا مطمئن برنامه شامل حساب ها ، نقش ها و مجوزهای غیرمفید حذف شده کنارگذاشته شوند.
- وضعیت برنامه خود را تنظیم کنید تا در CMDB کنار گذاشته شود.

+R به ریسک ها توجه شود

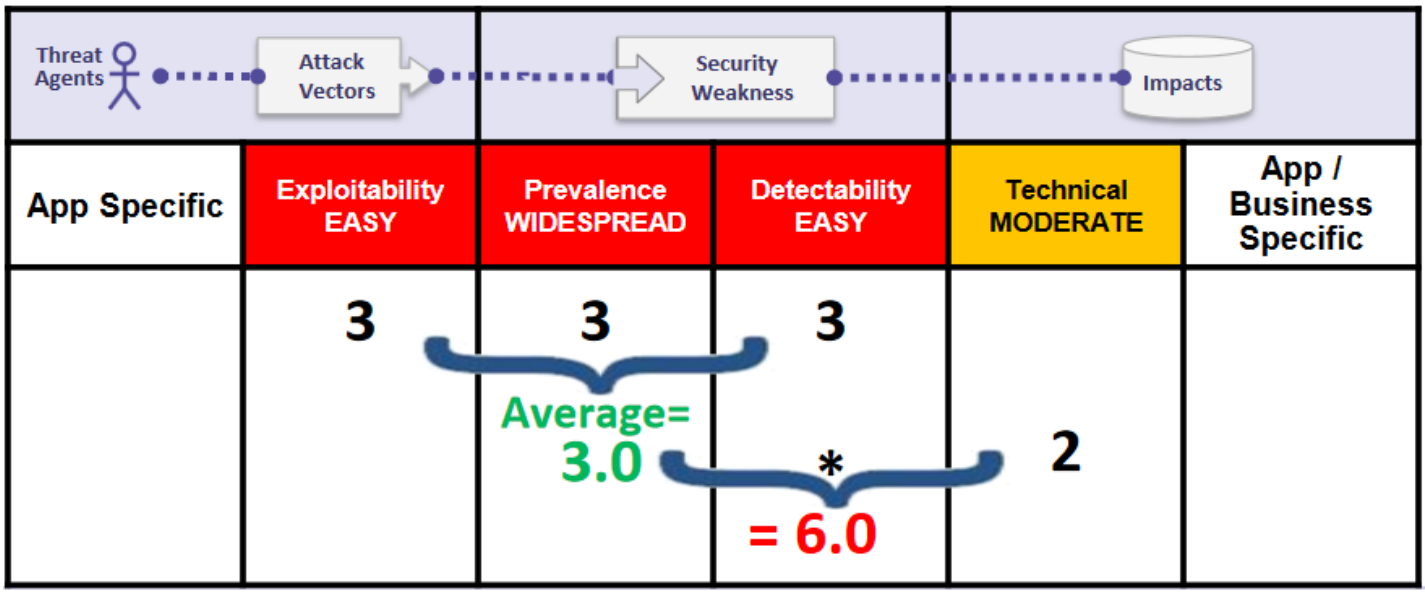
این درباره ریسک هایی است که ضعف ها رانشان می دهد

روش رتبه بندی ریسک برای Top 10 براساس روش OWASP Risk Rating است. برای هر گروه از Top 10 ما ریسک معمولی را تخمین زدیم که هر ضعف به یک برنامه وب معمولی با نگاهی به عوامل موثر و تاثیرگذار هر ضعف وارد می شود. سپس ما ۱۰ امتیاز را با توجه به ضعف هایی که به طور معمول بیشترین خطر را برای یک برنامه ارائه می دهند، اختصاص دادیم. این عامل ها با هر نسخه جدید ۱۰ Top به روز می شود. چراکه همه چیزتغییرمی‌کند وتکامل می‌یابد.

روش ارزیابی ریسک OWASP عوامل متعددی را برای کمک به محاسبه خطر آسیب پذیری مشخص شده تعیین می کند. با این حال، ۱۰ Top باید در مورد کلیات، نه آسیب پذیری های خاص برنامه های واقعی و API ها صحبت کند. در نتیجه ما هرگز نمی‌توانیم به اندازه مالکان برنامه و یا مدیران هنگام محاسبه ریسک برای برنامه آن‌ها دقیق باشیم. شما برای قضاوت درباره اهمیت برنامه و اطلاعات خود، که تهدیدات شما کدامند و اینکه چگونه سیستم شما ساخته و فعال شده‌است، بهترین هستید. روش ما شامل سه فاکتور احتمالی برای هر ضعف (شیوع، تشخیص و اکسپلویت آسان) و یک عامل موثر (تاثیر فنی) است. مقیاس‌های ریسک برای هرفاکتور از۱تا۳ به بالا با اصطلاحات خاص برای هرعامل متغیر است. شیوع یک ضعف فاکتوری است که معمولا مجبور نیستید حساب کنید. برای جمع‌آوری داده ها، آمار شیوع ازتعدادی از سازمان‌های مختلف (که درصفحه۲۵به آن اشاره شده)گرفته و ارائه شده و ما داده های آن‌هارا با هم جمع کردیم تا۱۰ احتمال بالای لیست موجودیت باتوجه به شیوع را ارائه دهیم. سپس این داده ها با دو عامل احتمال دیگر(تشخیص و اکسپلویت آسان) ترکیب شدند تا یک امتیاز احتمالی برای هر ضعف محاسبه شود. پس از آن، ضریب احتمال با ضریب تاثیر تخمین زده شده برای هر مورد، ضرب می شود تا رتبه بندی ریسک کلی برای هر مورد در ۱۰ Top را نشان دهد (هرچه بالاتر ، ریسک هم بیشتر) تشخیص، اکسپلویت آسان و تاثیر از طریق تجزیه و تحلیل CVE های گزارش شده که با هر یک از ۱۰ دسته برتر مرتبط بودند محاسبه شد .

توجه: این رویکرد احتمال عامل تهدید را درنظر نمی‌گیرد. همچنین هیچ یک ازجزییات فنی مرتبط با برنامه ویژه شمارا نیز در نظر نمی‌گیرد. هر یک از این عوامل می تواند به طور قابل توجهی احتمال حمله مهاجم و بهره برداری از آسیب پذیری خاص را تحت تاثیر قرار دهد. این رتبه تاثیر واقعی روی کسب وکارشما را در نظر نمی‌گیرد. سازمان شما باید تصمیم بگیرد که چه مقدار از خطر امنیتی از برنامه ها و API ها را با توجه به فرهنگ، صنعت و محیط قانونی شما می تواند بپذیرد. هدف ۱۰ OWASP Top این نیست که این تحلیل ریسک را برای شما انجام دهد.

موارد زیر محاسبه ما از ریسک A6: Security Misconfiguration-2017 را نشان می دهد.



جزئیات فاکتورهای ریسک

+RF

خلاصه ی ده عامل ریسک برتر

جدول زیرخلاصه‌ای از ۱۰ ریسک امنیتی کاربردی ۲۰۱۷ و عوامل که ما برای هر ریسک تعیین کرده‌ایم را نشان می‌دهد این فاکتورها براساس آمار موجود وتجربه تیم OWASP Top ۱۰ تعیین شده اند. برای درک این ریسک ها برای یک برنامه یا سازمان خاص، شما باید عوامل تهدید خاص و اثرات کسب‌وکار خود را در نظر بگیرید. درصورتی که هیچ عامل تهدید درموقعیتی برای انجام حمله لازم وجود نداشته باشد یا تاثیرکسب‌وکار ناچیزباشد, حتی وجود ضعف‌های برنامه ی شدید نیزممکن است خطرجدی را درپی نداشته باشد.

امتیاز	تاثیرات		ضعف امنیتی		مسیرهای حمله		ریسک
	کاربردی	فنی	تشخیص	شیوع	اکسپلویت شدن	عوامل تهدید	
۸	برنامه خاص	شدید	آسان	معمولی	آسان	برنامه خاص	A1:2017- Injection
۷	برنامه خاص	شدید	متوسط	معمولی	آسان	برنامه خاص	A2:2017-Authentication
۷	برنامه خاص	شدید	متوسط	گسترده	متوسط	برنامه خاص	A3:2017-Sens. Data Exposure
۷	برنامه خاص	شدید	آسان	معمولی	متوسط	برنامه خاص	A4:2017-XML External Entities (XXE)
۶	برنامه خاص	شدید	متوسط	معمولی	متوسط	برنامه خاص	A5:2017-Broken Access Control
۶	برنامه خاص	درحد متوسط	آسان	گسترده	آسان	برنامه خاص	A6:2017-Security Misconfiguration
۶	برنامه خاص	درحد متوسط	آسان	گسترده	آسان	برنامه خاص	A7:2017-Cross-Site Scripting (XSS)
۵	برنامه خاص	شدید	متوسط	معمولی	سخت	برنامه خاص	A8:2017-Insecure Deserialization
۴.۷	برنامه خاص	درحد متوسط	متوسط	گسترده	متوسط	برنامه خاص	A9:2017-Vulnerable Components
۴	برنامه خاص	درحد متوسط	سخت	گسترده	متوسط	برنامه خاص	A10:2017Logging&Monitoring-

ریسک های اضافی برای در نظر گرفتن

Top10 بسیاری از زمینه ها را پوشش می دهد، اما ریسک های زیادی وجود دارد که باید در سازمان خود در نظر بگیرید و ارزیابی کنید. بعضی از این ها در نسخه های قبلی Top ۱۰ ظاهر شده اند و برخی از جمله تکنیک های جدید حمله که در همه زمان ها شناسایی می شوند، دیده نشده اند. دیگر خطرات امنیتی مهم برنامه (به ترتیبCWE-ID) که شما نیز باید در نظر بگیرید عبارتند از:

- (CWE-352: Cross-Site Request Forgery (CSRF
- ('CWE-400: Uncontrolled Resource Consumption ('Resource Exhaustion', 'AppDoS
- CWE-434: Unrestricted Upload of File with Dangerous Type
- (CWE-451: User Interface (UI) Misrepresentation of Critical Information (Clickjacking and others
- CWE-601: Unvalidated Forward and Redirects
- (CWE-799: Improper Control of Interaction Frequency (Anti-Automation
- (CWE-829: Inclusion of Functionality from Untrusted Control Sphere (3rd Party Content
- (CWE-918: Server-Side Request Forgery (SSRF

مرور

درنشست پروژهOWASP شرکت کنندگان فعال واعضای جامعه از منظر آسیب‌پذیری تصمیم گرفتند. کلاس‌های آسیب‌پذیری را با استفاده از داده های کمی و تا حدی با بررسی‌های کیفی ، ۲ پله به جلو ببرند.

برآورد رتبه بندی شده صنعت

برای این نظرسنجی، ما دسته های آسیب پذیری را که پیش از این به عنوان on the cusp شناخته شده بودند و یا در لیست بازخورد RC1 ۲۰۱۷ در لیست ۱۰ TOP ذکر شده بودند جمع آوری کردیم. ما آنها را با یک نظرسنجی رتبه بندی کرده و از پاسخ دهندگان خواسته ایم تا چهار آسیب پذیری بالایی را رتبه بندی کنند که می بایست در OWASP Top ۱۰-۲۰۱۷ قرار گیرند. این نظرسنجی از ۲ تا ۱۸ سپتامبر ۲۰۱۷ انجام شد. ۵۱۶ پاسخ جمع آوری شد و آسیب پذیری ها رتبه بندی شدند.

ردیف	بررسی گروههای آسیب پذیری	امتیاز
۱	درمعرض افشاء قرار گرفتن اطلاعات خصوصی [CWE-۳۵۹] ('Privacy Violation')	۷۴۸
۲	شکستن رمزنگاری [CWE-۳۱۰/۳۱۱/۳۱۲/۳۲۶/۳۲۷]	۵۸۴
۳	شکستن رمزنگاری [CWE-۳۱۰/۳۱۱/۳۱۲/۳۲۶/۳۲۷]	۵۱۴
۴	مجوز عبور از کلید کنترل کاربر [CWE-۶۳۹] (IDOR* & Path Traversal)	۵۱۴
۵	رویدادنگاری و نظارت ناکافی [CWE-۲۲۳ / CWE-۷۷۸]	۴۴۰

در معرض افشاء قرار گرفتن اطلاعات خصوصی به وضوح بالاترین آسیب پذیری است، اما به راحتی به عنوان یک تاکید اضافی در مورد قرار گرفتن در معرض افشاءاطلاعات A3 ۲۰۱۷ حساس است. شکست های رمزنگاری می توانند در محدوده اطلاعات حساس قرار گیرند. رعایت نکردن امنیت در رتبه سوم قرار داشت و پس از آن رتبه ۱۰ را به عنوان A8: ۲۰۱۷-Insecure Deserialization اضافه کرد. چهارمین رده Broken Access Control -A5:۲۰۱۷ (شکسته شدن کنترل دسترسی)است. دربسیاری از نظر سنجی ها، به عنوان آسیب پذیری های مجوز، رتبه بالایی دارد. رتبه ۵ رده بندی در نظرسنجی رویدادنگاری و نظارت ناکافی است که ما معتقدیم که مناسب برای لیست ۱۰ TOP است، به همین دلیل است که آن تبدیل به A10: ۲۰۱۷ Insufficient Logging & Monitoring شده است. ما به یک نقطه ای رسیده ایم که در آن برنامه ها باید بتوانند تعریف کنند که چه چیزی ممکن است یک حمله باشد و قادر به ثبت هشدار، تشدید و واکنش مناسب باشند.

فراخوانی اطلاعات عمومی

به طور سنتی، داده‌های جمع‌آوری‌شده وآنالیز شده بیشتر در امتداد خطوط داده‌های فرکانس بودند: چند آسیب‌پذیری درکاربردهای آزمایش شده یافت شدند. همانطورکه می‌دانیم، ابزارهایی به طور سنتی همه نمونه‌های یافت‌شده ازآسیب‌پذیری راگزارش می‌کنند و انسان‌ها به طور سنتی یک یافته تنها با چند مثال را گزارش می‌کنند این باعث می شود که ترکیب دو سبک گزارش دهی به شیوه ای قابل مقایسه خیلی سخت باشد.

برای سال۲۰۱۷ نرخ شیوع توسط چندین برنامه دریک مجموعه داده که یک یا چند نمونه آسیب‌پذیری را داشتند محاسبه شد. داده های بسیاری از همکاران بزرگ در دو دیدگاه ارائه شده است. اول، فرکانس سنتی شمارش هر نمونه آسیب پذیری بود، در دومی تعداد برنامه های کاربردی بود که در آن هر یک از آسیب پذیری ها (یک یا چند بار) یافت شد. در حالی که کامل نیست ، این منطق به ما اجازه می دهد که داده ها را با ابزارهای کمکی و کمک های ابزار انسان مقایسه کنیم. داده های خام و تجزیه و تحلیل کار در GitHub در دسترس است. ما قصد داریم این کار را با ساختاری بیشتر برای نسخه‌های آینده ۱۰ Top گسترش دهیم. ما بیش از ۴۰ مطالب ارسالی را درفراخوانی برای داده دریافت کردیم وچون بسیاری ازداده‌های اصلی روی فرکانس متمرکزشده بودند، ما قادر بودیم از داده‌های۲۳شرکت‌کننده در پوشش ۱۱۴۰۰۰ برنامه استفاده کنیم.

ما از یک بلوک زمانی یک ساله استفاده کردیم که امکان شناسایی توسط شرکت کننده را دارد. اکثریت برنامه های کاربردی منحصر به فرد هستند، هرچند ما احتمال برخی برنامه های تکراری بین داده های سالانه Veracode را تایید می کنیم. ۲۳ مجموعه داده مورد استفاده به عنوان ابزار آزمایش انسانی یا به طور خاص نرخ شیوع از ابزارهای کمکی انسانی شناسایی شدند. بی قاعدگی‌ها در داده‌های انتخابی ۱۰۰%+ شیوع موارد به حداکثر۱۰۰ درصدتنظیم شدند. برای محاسبه میزان شیوع، ما درصد کل برنامه های موجود را که حاوی هر نوع آسیب پذیری بود، محاسبه کردیم. رتبه بندی شیوع برای محاسبه شیوع در ریسک کلی برای رتبه بندی TOP10 مورد استفاده قرار گرفت.