به نام خدا

پروژه ۱۰ آسیب پذیری اول اپلیکیشن های تحت وب در سال ۲۰۱۳از OWASP،

شماره ۱ تزریق

OWASP TOP 10 Project 2013 - A1 Injection

مقدمه:

رخنه های تزریق مثل تزریق SQL یا OS یا LDAP زمانی اتفاقی می افتد که یک داده ی بدون اعتبارسنجی، به عنوان بخشی از کوئری یا دستور به برنامه ی مفسر (interpreter) ارسال شود. داده های مخرب، برنامه ی مفسر را به گونه ای فریب می دهد که نفوذگر بتواند بدون داشتن مجوز دسترسی و احراز صلاحیت، روی سیستم مقابل command اجرا و یا به داده های آن دسترسی پیدا کند.

ویژگی های این مورد Application Specific

به بررسی کسانی می پردازد که بتوانند داده های بدون اعتبارسنجی به سیستم ارسال کنند. این افراد شامل کاربران خارجی، کاربران داخلی و مدیران سیستم می شود.

قابلیت اکسپلوییت شدن: آسان (راحت ترین حالت؛ یعنی ۳ از ۳)

این حمله با ارسال یک متن ساده صورت می پذیرد و به اصطلاح به آن text-based attack می گوییم. این حمله با استفاده از syntaxهای برنامه های مفسر، اکسپلوییت می شود و در اکثر مواقع به وسیله ی این حمله به تمام منابع هدف مثل دیتابیس ها دسترسی خواهیم داشت.

میزان شیوع: رایج (حالت وسط؛ یعنی ۲ از ۳)

قابلیت شناسایی: متوسط (حالت وسط؛ یعنی ۲ از ۳)

رخنه مربوط به تزریق زمانی رخ می دهد که یک داده ی بدون اعتبارسنجی به برنامه مفسر ارسال شود. این رخنه بسیار شایع است و اغلب در کوئری های مربوط به SQL, LDAP, XPATH, NoSQL ; در فرمان های سیستم عامل; پارسرهای XML; هدرهای SMTP; آرگومان های یک برنامه;... یافت می شود. پیدا کردن رخنه تزریق یا همان Injection flaw بسیار آسان بوده و با بررسی سورس کد به راحتی آشکار می شود اما پیدا کردن آن به وسیله آزمون و خطا به مراتب سخت تر خواهد بود. اگرچه که اسکنرها به صورت خودکار جهت آشکارسازی این رخنه به نفوذگران کمک می کنند.

میزان تاثیر آن: شدید (خطرناک ترین حالت؛ یعنی ۳ از ۳)

اینجکت می تواند باعث از بین رفتن یا خراب شدن داده ها، مشکل در حساب های کاربری یا از کار افتادن سرویس دهی شود. (حمله DOS) در بعضی از موارد، اینجکت باعث از کار افتادن یک میزبان به صورت کامل می شود.

تاثیر آن در تجارت:

بررسی ارزش تجاری مربوط به داده هایی که تحت تاثیر این آسیب پذیری قرار می گیرند و پلتفرمی که برنامه ی مفسر روی آن در حال اجراست. ممکن است تمام داده ها دزدیده، دستکاری و یا پاک شوند. آیا ممکن است به واسطه ی این آسیب پذیری به شهرت شما آسیب برسد؟

آیا من یک آسیب پذیری قابل اینجکت شدن هستم؟

برای پیدا کردن جواب این سوال، بهترین کار این است که تمام داده های بدون اعتبارسنجی را قبل از اینکه به برنامه ی مفسر ارسال کنیم از کوئری ها و دستورات جدا کنیم. برای فراخوانی های SQL این کار به این معنی است که متغیرها را در عبارات و روال های ذخیره کننده یا همان stored proceduresها استفاده کنیم و از استفاده ی مستقیم

آن ها در کوئری های داینامیک خودداری کنیم. [مترجم: وقتی این کار را انجام دهیم، ورودی ها به متغیرها الصاق یا همان bind می شوند و تاثیر مخرب آن ها از بین می رود]

چگونه از اینجکت شدن جلوگیری کنم؟

راه جلوگیری از تزریق این است که تمام داده هایی که اعتبارسنجی نشده اند را از کوئری ها و دستورات جدا کنیم.

۱- بهترین گزینه استفاده از یک API امن است. به گونه ای که از استفاده ی مستقیم از برنامه ی مفسر جلوگیری شده و به جای آن، برای اعمال پارامترهای ورودی یک واسط کاربری ایجاد کند.

۲- اگر امکان استفاده از API جهت پارامتردهی فراهم نیست، می بایست به دقت کارکترهایی که برای برنامه ی مفسر یک معنای خاص می دهد را escape کنیم؛ که این کار با استفاده از escape syntax صورت می گیرد. [مترجم: برای مثال در SQL کارکتر " به معنای پایان یافتن رشته است. بنابراین رشته های ورودی را باید چک کنیم و هر جایی که با " مواجه شدیم، آن را به وسیله ی کارکتر \ به اصطلاح escape کنیم] OWASP's ESAPI کارهای مختلفی که جهت escape صورت می پذیرد را در این لینک آورده است.

۳- استفاده از لیستی شامل ورودی های مجاز یا همان white list برای اعتبارسنجی ورودی ها نیز پیشنهاد می شود. اما به عنوان یک راه دفاعی قطعی و به صورت کامل نیست. چرا که بسیاری از برنامه ها در قسمت ورودی لازم دارند تا از کاراکترهای خاصی استفاده کنند. اگر این چنین بود رویکردهای مجاز عبارتند از ۱- کلیدواژه ی and و ۲- با توجه به توضیحات قبل، مطمئن شوید که استفاده از آن ها به صورت امن انجام می شود. OWASP's ESAPI کتابخانه ای قابل توسعه برای روش های اعتبارسنجی ورودی ها بر اساس لیست های مجاز را در این لینک قرار داده است.

سناریوهای حمله در قالب مثال:

سناریو #1: از داده های ورودی قبل از اینکه اعتبارسنجی شوند، در فراخوان های SQLای که آسیب پذیر هستند، استفاده می شود.

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

سناریو #2: مشابه مورد قبل، اعتماد کورکورانه ی برنامه به فریم ورک در یک کوئری ممکن است باعث آسیب پذیر شدن آن شود. (HQL در زیر، مخفف Hibernate Query Language است)

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

در هر دو حالت بالا، نفوذگر مقدار پارامتر id را با مقدار ' or '1'='1 عوض می کند؛ برای مثال:

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

این تغییر به این معناست که هر دو کوئری در سناریوی ۱و ۲ تمام رکوردهای موجود در جدول account را بر می گردانند. نوع خطرناک تری از این حمله این است که هکر، داده ها را دستکاری کند یا از stored procedureها استفاده کند.

منابع:

در مورد سایر منابع نیز، یا ترجمه ی آن ها در گروه موجود است و یا در حال ترجمه ی آن هستیم.

منابع موجود در OWASP:

- OWASP SQL Injection Prevention Cheat Sheet
- OWASP Query Parameterization Cheat Sheet
- OWASP Command Injection Article
- OWASP XML eXternal Entity (XXE) Reference Article
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP Testing Guide: Chapter on SQL Injection Testing

منابع خارجی:

- CWE Entry 77 on Command Injection
- CWE Entry 89 on SQL Injection
- CWE Entry 564 on Hibernate Injection

تاریخ ساخت: June 12, 2013 یا ۲۲ خرداد ۱۳۹۲

تاریخ تحقیق: July 29, 2014 یا ۷ مرداد ۱۳۹۳

/* تصحیح این مقاله، چه در ترجمه و چه در مباحث علمی ، توسط شما دوستان باعث خوشحالی خواهد بود. لطفا آن را با tamadonEH@gmail.com مطرح نمایید.*/

برای مشاهده لیست مقالات کار شده توسط گروه ما به لینک زیر مراجعه فرمایید

https://github.com/tamadonEH/list/blob/master/list.md