به نام خدا

DOM–Based XSS حمله

OWASP Attack Category: Dom Based XSS

تعریف

Dom Based XSS (که ممکن است در برخی از نوشته ها با Type-0 XSS هم شناخته شود.) نوعی از حمله ی XSS است که داده ها و دستورهای مخرب با هدف تغییر و دستکاری «محیط» یا همان «environment» مربوط به DOM در مرورگر قربانی اجرا می شود. این کار به وسیله ی اسکریپت سمت کلاینت و اجرای کدها و داده های غیرمنتظره صورت می پذیرد. در این روش، صفحه ی درخواست شده و پاسخ (response) در HTTP هیچ تغییری نمی کند؛ اما کد سمت کلاینت در صفحه ی درخواست شده، به گونه ای مختلف اجرا می شود و این تفاوت به خاطر دستکاری نامطلوبی است که در محیط DOM رخ داده است.

انواع دیگری از حملات XSS نیز وجود دارد. (Stored, Reflected) که قسمت payload حمله در قسمتی از پاسخ (response) صفحه جاسازی می شود و این کار با استفاده از رخنه ی موجود در سمت سرور انجام می شود.

مثال

فرض کنید قطعه کد زیر فرمی را می سازد که کاربر به وسیله ی آن می تواند زبان سایت را اانتخاب کند. زبان پیش فرض نیز در رشته ی کوئری و در پارامتر "defual" مشخص شده است:

```
…
Select your language:
<select><script>
document.write("<OPTION
value=1>"+document.location.href.substring(document.location.href.ind
exOf("default=")+8)+"</OPTION>");
document.write("<OPTION value=2>English</OPTION>");
</script></select>
…
```

URL صفحه مشابه زیر خواهد بود:

```
http://www.some.site/page.html?default=French
```

حمله ی DOM Based XSS با ارسال URLای مشابه زیر به سمت قربانی شکل می گیرد:

```
http://www.some.site/page.html?
default=<script>alert(document.cookie)</script>
```

زمانی که قربانی روی لینک کلیک می کند، مرورگر درخواست زیر را ارسال می کند:

```
/page.html?default=<script>alert(document.cookie)</script>
```

صفحه ای که سرور در پاسخ ارسال می نماید شامل کد جاوا اسکریپت بالا خواهد بود. مرورگر برای صفحه یک شی ای (object) از DOM می سازد که در آن شی document.location شامل زشته ی زیر می شود:

```
http://www.some.site/page.html?
default=<script>alert(document.cookie)</script>
```

کد جاوااسکریپت اصلی و اولیه در صفحه برای جلوگیری از اینکه پارامتر default شامل تگ های HTML شود، کاری نکرده است. بنابراین به سادگی تمام آن را در زمان اجرا، در صفحه DOM چاپ و ذخیره می کند. بعد از آن مرورگر صفحه ی خروجی را رندر کرده و اسکریپت هکر را اجرا می کند:

```
alert(document.cookie)
```

توجه داشته باشید که پاسخ HTTP که از سرور ارسال می شود به هیچ وجه شامل داده های هکر نخواهد بود. این داده ها دقیقا در زمان اجرا (runtime) و در سمت کلاینت تولید می شوند و این حمله زمانی امکان پذیر خواهد بود که اسکریپتِ مخرب به متغیرهای DOM مثل document.location دسترسی داشته باشد و این دسترسی صرفا از نظر مرورگر، غیرمجاز و مخرب نباشد.

تکنیک های پیشرفته و مشتقات آن

در مثال بالا، تا زمانی که payload شامل داده ها و دستورات و... در پاسخ HTTPای که توسط سرور داده می شود، الصاق و embedded نشود، به عنوان بخشی از درخواست HTTP به دست سرور می رسد و بنابراین حمله در سمت سرور قابل تشخیص خواهد بود. این لینک راه های مختلفی که برای جلوگیری از تشخیص توسط سرور استفاده می شود را بیان کرده است. علاوه بر آن راه های مختلفی که می توان از payload استفاده کرد را هم ذکر کرده است.

یکی از راه های جلوگیری از ارسال payload به سمت سرور استفاده از کاراکتر "#" می باشد. عباراتی که بعد از # می آیند، توسط مرورگر به سمت سرور ارسال نمی شود. بنابران تمام کدهای سمت کلاینت که URL در آن به صورت fragment شده در آمده [مترجم: یعنی آن را با # تکه تکه کرده ایم] ممکن است در برابر document.location آسیب پذیر باشد و در این حالت، قسمت payload هیچ وقت به سرور ارسال نخواهد شد.. بنابراین در مثال بالا DOM Based XSS به شکل زیر تغییر خواهد کرد:

```
http://www.some.site/page.html#default=<script>alert(document.cookie)
</script>
```

که با این کار، حمله بدون ارسال هیچ داده ای به سمت سرور شکل خواهد گرفت. (درخواستی که به سرور ارسال می شود شامل page.html و بدون هیچ پارامتری در URL خواهد بود)

در دسامبر ۲۰۰۶، مقاله ای توسط Stefano Di Paola و Giorgio Fedon منتشر شد که در آن یک حمله ی XSS در پلاگین Acrobat PDF نشان داده شده بود. (لینک) این حمله یک حمله ی DOM Based XSS از نوع fragment به داکیومنت های PDF بود. محققان نشان دادند که در داکیومنت های PDFای که در مرورگرها و به وسیله ی پلاگین Acrobat نمایش داده و رندر می شوند، می توان در قسمت پایانی آن ها جاوااسکریپت اجرا کرد. با این وصف برای اینکه در یک سایت بتوانیم جاوا اسکریپت خود را در یک محتوای (DOM) اجرا کنیم، تنها کافی است که یک لینک PDF در آن سایت پیدا کرده و آن را اکسپلوییت کنیم. اگر هکر بتواند کاربر را به نحوی فریب دهد که روی لینکی مشابه زیر کلیک کند:

```
http://www.some.site/somefile.pdf#somename=javascript:attackers_scrip
t_here
```

قربانی به وسیله Acrobat readerای که وصله نشده در برابر حمله، شکست می خورد. شرکت Adobe بعد از اینکه این آسیب پذیری شناخته شد، آن را وصله کرد اما برای اینکه تمام کاربران آسیب پذیر آن را وصله کنند به مدت زمان زیادی احتیاج است.

Ivan Ristic در ارائه ی خود به بیان راه کارهایی برای جلوگیری از این حمله در سمت سرور پرداخته است که در این لینک قابل مشاهده است.

توسعه و گسترش

Ory Segal در این لینک نشان داده است که چگونه صفحه ی هدف می تواند طوری فریم بندی شود که فریم والد (که تحت کنترل هکر است) از صفحه ی هدف، مطابق میل هکر بارگزاری شود. این تکنیک نشان می دهد که چگونه دستکاری DOM می تواند به منظور دستکاری جریان اجرای دستورات و ... باعث تغییر صفحه ی هدف شود.

ابزار تست و تکنیک ها

گروه امنیتی Minded Security روی DOM Based XSS یک سری تحقیقات اساسی انجام داده است و در حال حاضر روی دو پروژه برای بهبود و کمک به DOM Based XSS کار می کنند:

۱- ابزار DOMinator- که یک ابزار تجاری بوده و روی مرورگر فایرفاکس نصب می شود. این ابزار موتور جاوااسکریپت Spidermonkey را به نحوی تغییر داده است که می توان از آن برای شناسایی و تایید رخنه ی DOM Based XSS بهره برد.

لینک: https://dominator.mindedsecurity.com

۲- ویکی DOM XSS- که مشغول به ساخت مرجعی کامل در این خصوص اند ولی هنوز جای کار دارد.

لینک: http://code.google.com/p/domxsswiki

۳- DOM Snitch- یک افزونه ی کروم است که برای توسعه دهندگان و نفوذگران این امکان را فراهم می سازد که شیوه های رایجی که در کدهای سمت کلاینت آسیب پذیراند را شناسایی کند. این ابزار توسط گوگل پشتیبانی می شود:

لینک: http://code.google.com/p/domsnitch

تکنیک های مقابله

لینک: https://www.owasp.org/index.php/DOM_based_XSS_Prevention_Cheat_Sheet

منابع

[1] "DOM Based Cross Site Scripting or XSS of the Third Kind" (WASC writeup), Amit Klein, July 2005

http://www.webappsec.org/projects/articles/071105.shtml

[2] "JavaScript Code Flow Manipulation, and a real world example advisory - Adobe Flex 3 Dom-Based XSS" (Watchfire blog), Ory Segal, June 17th, 2008

http://blog.watchfire.com/wfblog/2008/06/javascript-code.html

[3] "Attacking Rich Internet Applications" (RUXCON 2008 presentation), Kuza55 and Stefano Di Paola, November 2008

http://www.ruxcon.org.au/files/2008/Attacking_Rich_Internet_Applications.pdf

[4] "Subverting Ajax" (23C3 presentation), Stefano Di Paola and Giorgio Fedon, December 2006

http://events.ccc.de/congress/2006/Fahrplan/attachments/1158-Subverting_Ajax.pdf

[5] "Protecting Web Applications from Universal PDF XSS" (2007 OWASP Europe AppSec presentation) Ivan Ristic, May 2007

http://www.owasp.org/images/c/c2/OWASPAppSec2007Milan_ProtectingWebAppsfromUniversalPDFXSS.ppt

[6] OWASP Testing Guide

Testing_for_DOM-based_Cross_site_scripting_(OWASP-DV-003)

تاریخ ساخت: Feb 13, 2014 یا ۲۴ بهمن ۱۳۹۲

تاریخ تحقیق: August 19, 2014 یا ۲۸ مرداد ۱۳۹۳

لینک مقاله: https://www.owasp.org/index.php/DOM_Based_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