Bypassing NoScript Security Suite
Using Cross-Site Scripting and MITM Attacks

March 2016

Mazin Ahmed | mazin@mazinahmed.net | @mazen160

Table of Contents

Abstract 3
Introduction 3
Research 4
Solution 7
Recommendations 7
Notes 7
Disclosure Timeline 8
Conclusion 8
References 9
Acknowledgement 9

1. Abstract

NoScript Security Suite is a powerful security add-on for Firefox, Seamonkey and other Mozilla-based browsers. Its main task is to block Javascript, Flash, Java, as well as many other plugins from executing untrusted code on the user's browser through blocking it and only allowing certain trusted whitelisted sites.

This paper discusses different techniques that an attacker can use to bypass NoScript Security Suite Protection. These techniques can be used by malicious vectors in bypassing the default installation of NoScript. The paper also provides solutions and recommendations for end-users that can enhances the current protection of NoScript Security Suite.

2. Introduction

NoScript (also known as NoScript Security Suite) is a free and open-source extension that provides additional security protection from potential exploits by disabling Javascript, Java, Flash, and other plugins for untrusted sites, and also provides a number of additional features. It's a vital addition to ensure the maximum possible security for the user.

I will be demonstrating possible techniques that an attacker can use in order to bypass NoScript. I will be also explaining a mechanism that I have developed in bypassing the default installation of NoScript Security Suite.

3. Research

This section discusses the results of the conducted research.

3.1 First Bypass: Using NoScript Detection Bugs

The bypass can occur using a detection bug that affects NoScript. Similar bypasses have been demonstrated previously by Julien Voisin[1], Gareth Heyes[2], and others. All known issues has been patched.

3.2 Second Bypass: By Exploiting a Cross-Site Scripting Vulnerability Against Whitelisted Websites

This bypass can be demonstrated by finding and exploiting a cross-site scripting vulnerability against any of the whitelisted domains. Since the whitelisted domains are allowed to execute Javascript on the client's browser, a single XSS vulnerability is all what it takes to bypass the default installation of NoScript. Although NoScript provides a cross-site scripting filter protection, it's not sufficient enough to block XSS attacks.

3.2.1 Proof of Concept

live.com is whitelisted in the default installation of NoScript. I have identified a XSS vulnerability on the domain, and reported it to Microsoft. I'm not planning to disclose information regarding this XSS vulnerability, but I will be using it to demonstrate how it can be used to bypass NoScript.

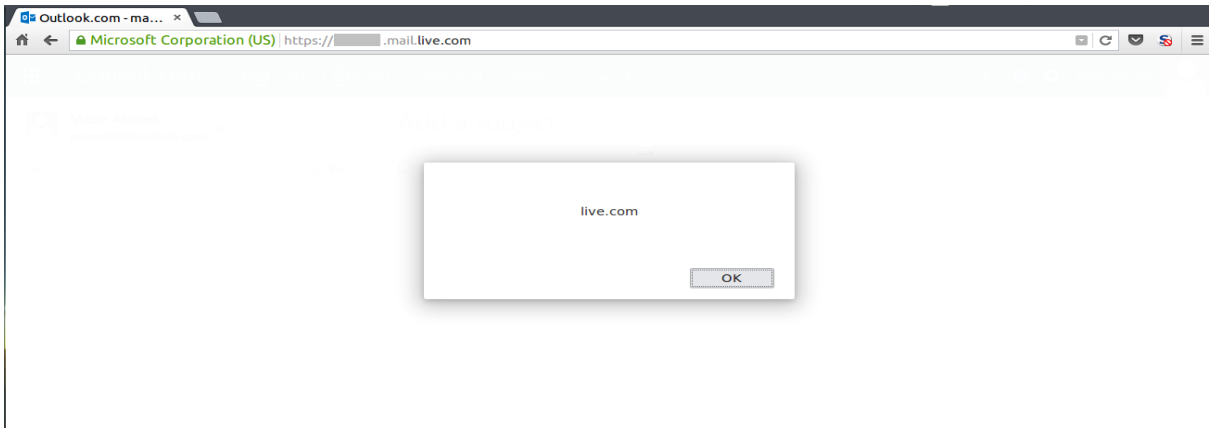

Figure 1: A proof of concept for an XSS vulnerability on live.com

The browser is using the default installation of NoScript. The XSS payload has been executed successfully without any interruption by NoScript Security Suite.

3.3 Third Bypass: Using MITM Attacks

This attack is quite different from other attacks. It uses typical MITM attacks to automate the process of bypassing NoScript Security Suite.

We already know that the default installation of NoScript Security Suite is using a number of whitelisted sites that are allowed to execute Javascript freely. The security of the executed Javascript is mainly relied on the security of the whitelisted website, and the security of the current network that is being used to handle the requests.

By exploiting the client's network in a way that can trick the client's browser into executing Javascript within the browser, NoScript Security Suite can be bypassed. Note that this is not a theoretical attack, I will be demonstrating it with a valid proof of concept.

3.3.1 Steps to Reproduce

1. The victim is using Firefox with NoScript Security Suite enabled.
2. The attacker has access into the victim's network.
3. The victim makes a request to a website that is using plain HTTP.
4. The attacker intercepts the response of the request, and injects a hidden iframe that points to a whitelisted site within the response.
5. The victim's browser makes a request to the whitelisted website.
6. The attacker intercepts the response of the request, and injects the desired Javascript payload.
7. The response will be received by the victim's browser.
8. The browser will render the response, and the Javascript payload will be executed.

3.3.2 Proof of Concept

I have written multiple proof of concepts that can be used to automate and perform this attack on the fly. I have used BetterCap, a MITM framework that is capable of automating a number of MITM attacks. The proof of concepts that I wrote are BetterCap modules that perform the task of bypassing NoScript as described above.

Download Link: [https://mazinahmed.net/uploads/HackNoScript_PoCs.zip]

Usage

```
$ bettercap -G [GATEWAY] -T [TARGET] --proxy --proxy-module
hack_noscript_poc.rb
```

Now, whenever a client that is protected by NoScript makes an HTTP request, the BetterCap module would handle all the work, and ensures the executing of the Javascript payload within the victim's browser.

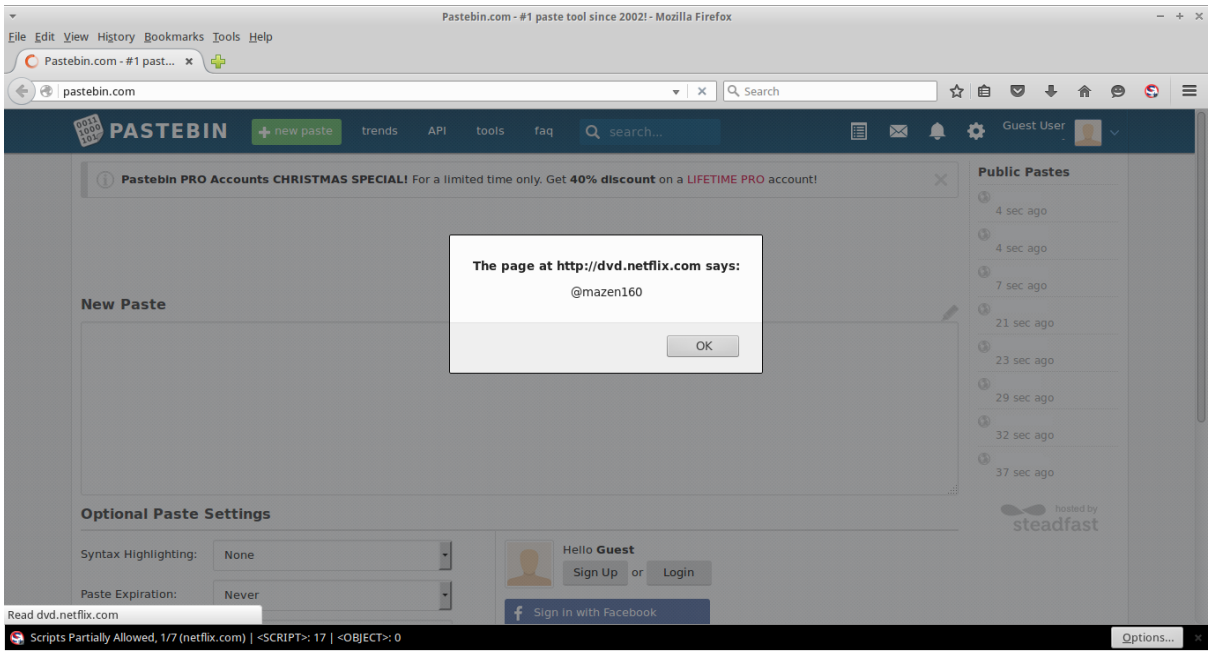

Figure 2: A screenshot that shows the execution of Javascript against a victim's browser while using the BetterCap Module.

Note that I have not exploited a cross-site scripting vulnerability on *pastebin.com* or *dvd.netflix.com*, this is the results of launching the BetterCap module within the victim's network. It will result on executing Javascript on the victim's browser that is protected by NoScript as demonstrated in *Figure 2*.

4. Solution

The following are suggested solutions for the discussed issues with the research

- Update NoScript to the latest version.
- If you would to ensure the maximum possible protection, you need to customize the configurations using the recommendations in the next section.

5. Recommendations

- Ensure that “Forbid active web content unless it comes from a secure (HTTPS) connection” option is set to “Always”.
- Validate each entry in the whitelisted domains, and delete unnecessary whitelisted domains.

If you would like a customized NoScript configuration, I have provided a NoScript configuration that can hopefully ensure a higher level of security when using NoScript Security Suite.

NoScript Configuration Download Link: [https://mazinahmed.net/uploads/noscript.conf]

6. Notes

Users of TOR browsers are not affected by the second and third bypass by default, since TOR is prebuilt with custom NoScript that does not include any whitelisted domain. However, you may need to double-check if your TOR browser has “*Forbid active web content unless it comes from a secure (HTTPS) connection*” option set to “*Always*”.

7. Disclosure Timeline

October 25th, 2015 – Initial Disclosure.
October 26th,2015/November 04th, 2015 – Discussion regarding a possible patch.
November 19th, 2015 – I Sent a proof of concept to the developer.
November 20th, 2015/November 21th, 2015 – Discussion regarding the exploitation of the issue.
December 17th, 2015 – Initial patch has been publicly released on NoScript v2.7 by removing a number of sites to reduce the attack surface.
January 2st, 2016 – I Demonstrated the missing points in the initial patch, and providing a second proof of concept.
January 08th, 2016 – The developer asked for further information.
January 11th, 2016 – I Responded with required information.
January 23th, 2016 – The developer agreed to implement the proposed solution.
March 16th, 2016 – A patch has been released on NoScript v2.9.0.5 to automatically upgrade to HTTPS sites found in the default whitelisting.

8. Conclusion

NoScript is one of the most essential projects in the field for protecting the end-user from known (and unknown exploits in few cases). I'm a big supporter of the project, and I'm glad to help in increasing the security of NoScript Security Suite.

When testing NoScript protection, I have come to conclusion that NoScript, is as same as any security product, can be bypassed in a certain way. Although the protection and the way of evading could differ from a product to another, in the end, a full evasion can always be possible.

This was a short research that discusses how can NoScript Security Suite be bypassed using Cross-Site Scripting attacks against the default whitelisted sites on the default installation of NoScript. It also showed how we can use network attacks in bypassing the default installation of NoScript Security Suite.

9. References

[1]: https://dustri.org/b/noscript-script-disabled-bypass-poc-for-tails-13.html
[2]: http://blog.portswigger.net/2015/07/noscript-xss-filter-bypass.html

10. Acknowledgement

I would like to thank the following individuals for their contribution during the research.

- Giorgio Maone
- Simone Margaritelli