



No Script XSS Injection Checker

False Positive in Handling Nested URL's – Bug/Optimization Stringency

Aditya K Sood, Security Researcher

<http://www.secniche.org>

Date: 1 January 2010

Note: The affected version 1.9.9.30 or less

About NoScript's

The NoScript's Firefox extension provides extra protection for Firefox, Flock, Sea Monkey and other Mozilla-based browsers: this free, open source add-on allows JavaScript, Java and Flash and other plugins to be executed only by trusted web sites of your choice (e.g. your online bank), and provides the most powerful Anti-XSS protection available in a browser. NoScript's unique whitelist based pre-emptive script blocking approach prevents exploitation of security vulnerabilities (known and even not known yet!) with no loss of functionality

Version – 1.9.9.31/ 1.9.9.32 will correct this problem in XSS Injection Checker.

The development version is under testing.

<http://noscript.net/getit#devel>

Disclaimer: The document has no commercial nature it is written for educational purposes only.

Thank to Giorge for discussing in detail.

Discussion

NoScript is showing some interesting false positive when ingress testing is performed on the anti xss component. The NoScript fails to optimize the complex xss payloads present as nested URL's thereby resulting in positive XSS attack detection and raising the XSS message bar. The problem persists severely in the following cases:

1. Whitelist approach for anti XSS protection.
2. Direct handling of XSS payloads.

Let's look into the detail scenario. It has been stated in the Anti XSS filter documentation:

*"NoScript features unique **Anti-XSS counter-measures** against XSS Type 0 (DOM based) and XSS Type 1 (Reflective, absolutely the most common) attacks targeted to whitelisted sites. Whenever a certain site tries to inject JavaScript code inside a different trusted (whitelisted and JavaScript enabled) site, NoScript filters the malicious request neutralizing its dangerous load."*

The whitelisted technique is opted to set a base for scrutinizing the integrity and trust of the content from the trusted domain marked as whitelisted. This simply projects the positive working as script included from one whitelisted domain to another whitelisted domain will be treated as trusted against all sorts of content. That's quite true for the robustness and working functionality of the software. Otherwise the technique itself fails. There will be no fun of whitelisted domains and vice versa. So it has to be followed as such. There is no doubt about the working quality of NoScript which is very good but certain conditions and considering the nature of XSS randomization vector it is sometimes hard to optimize the content in the requests that come from different domains.

Primarily looking at the functionality of NoScript , the whitelisted domains are considered as integral from security point of view. Mostly the domains that generate user content based on the information provided by the users it is considered as unsafe. So direct applicability of those websites as default white listed domains is not a benchmark to be defined. It is possible for the users to set it of their own by exploring the advance options.

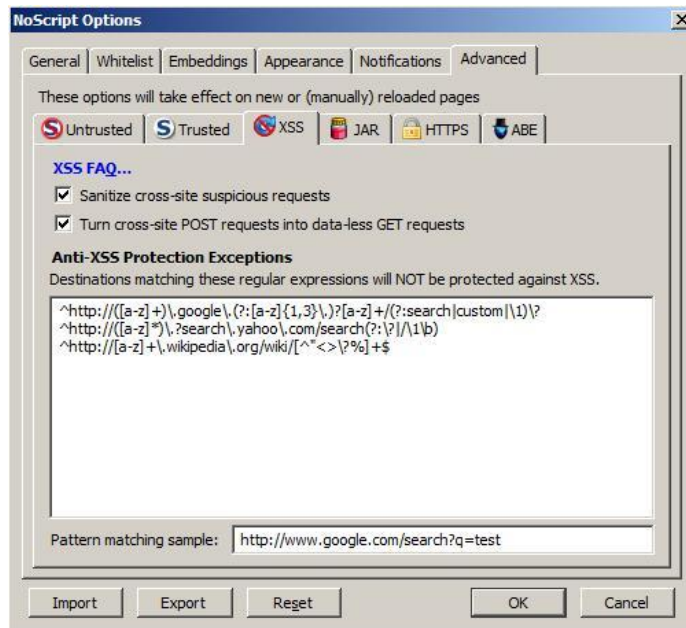
The NoScript domain documentation describes certain domains as whitelisted as mentioned below

1. **chrome:**
2. **about:xyz**
3. **addons.mozilla.org**
4. **noscript.net, flashgot.net, information.com, maone.net**
5. **gmail.com and google.com** (GMail)
6. **hotmail.com, live.com, microsoft.com, msn.com, passport.com, passport.net,**
7. **passportimages.net** (Microsoft webmail services)
8. **yahoo.com, yimg.com** (Yahoo! Mail)
9. **googlesyndication.com**

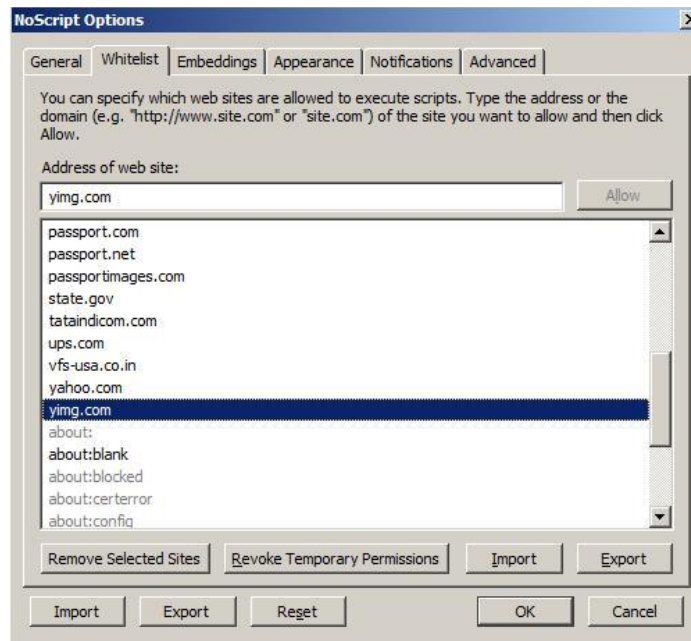
So it means the XSS Injection Checker follows the benchmark of whitelisted technique. But let's analyze the false positive bug.

While performing testing the below mentioned case and several like them are analyzed prior to report the details to the NoScript designer. The first false positive is detected between the Yahoo.com and Yimg.com which are both included in whitelisted domain by default.

The XSS filter is set as presented below for sanitizing URL's



The whitelisted domains are structured as

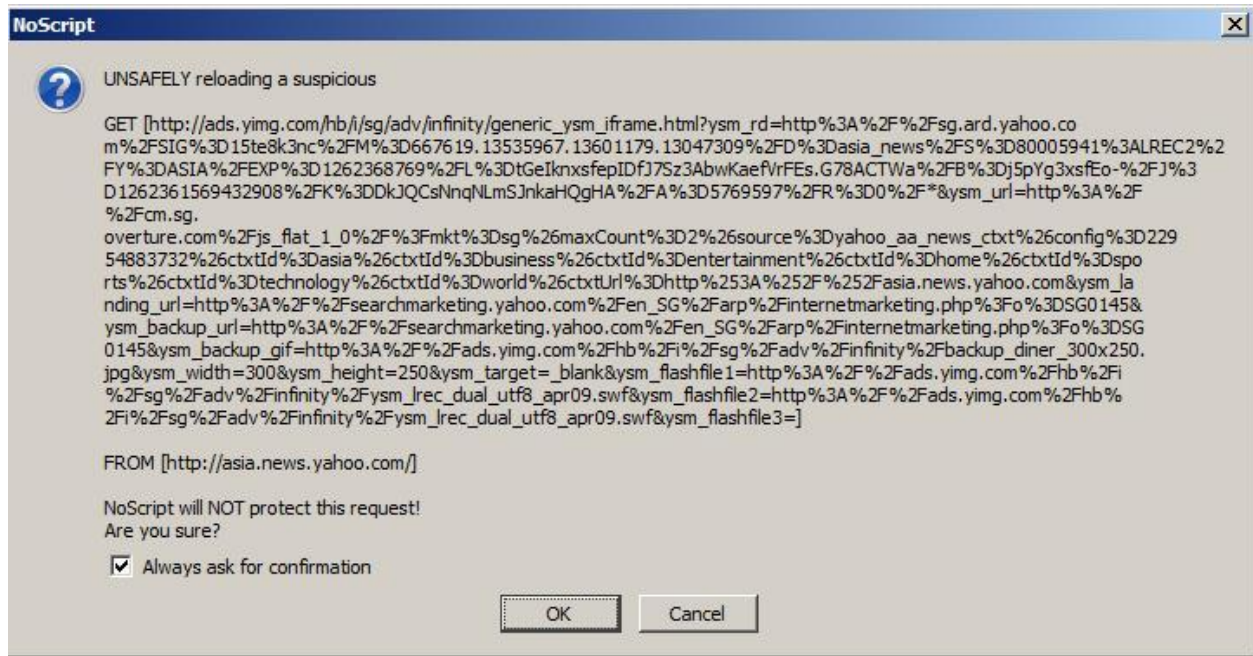


The yahoo and **yimg** domain is set as trusted ones.

But there is a problem in the specific set of version as


```
%233214267133740236789&ysm_url=http%3A%2F%2Fcm.sg.overture.com%2Fjs_flat_1_0%2F%3Fmkt%3Dsg%
26maxCount%3D2%26source%3Dyahoo_aa_news_ctxt%26config%3D22954883732%26ctxtId%3Dasia%26ctxtId
%3Dbusiness%26ctxtId%3Dentertainment%26ctxtId%3Dhome%26ctxtId%3Dsports%26ctxtId%3Dtechnology%2
6ctxtId%3Dworld%26ctxtId%3Dhttp%253A%252F%252Fasia.news.yahoo.com&ysm_landing_url=http%3A%2F%
2Fsearchmarketing.yahoo.com%2Fen_SG%2Farp%2Finternetmarketing.php%3Fo%3DSG0145&ysm_backup_url=h
tp%3A%2F%2Fsearchmarketing.yahoo.com%2Fen_SG%2Farp%2Finternetmarketing.php%3Fo%3DSG0145&ysm
_backup_gif=http%3A%2F%2Fads.yimg.com%2Fhb%2Fi%2Fsg%2Fadv%2Finfinity%2Fbackup_diner_300x250.jpg
&ysm_width=300&ysm_height=250&ysm_target=_blank&ysm_flashfile1=http%3A%2F%2Fads.yimg.com%2Fhb%
2Fi%2Fsg%2Fadv%2Finfinity%2Fysm_lrec_dual_utf8_apr09.swf&ysm_flashfile2=http%3A%2F%2Fads.yimg.com%
2Fhb%2Fi%2Fsg%2Fadv%2Finfinity%2Fysm_lrec_dual_utf8_apr09.swf&ysm_flashfile3=#3266275859084963583]
.h
```

Unsafe warning layout



On discussion with the NoScript Author the problem persists in the unescaped part of the URL as mentioned below

```
/hb/i/sg/adv/infinity/generic_ysm_iframe.html?ysm_rd=http://sg.ard.yahoo.com/SIG`/M%20667619.13535967.1
3601179.13047309/D`/S%2080005941:LREC2/Y`/EXP%201262210734/L`.S1taSztQ20L1c3W.rks7spkAAM8j/B`-
/J%201262203534421418/K`.6bZGv6oA/A%205769597/R%200/*#7273153851086377751&ysm_url=http:
```

The author further explains that due to presence of complex JavaScript fragments the XSS alert is triggered by the Injection Checker. On further point as a solution, appropriate vector can be implemented as

```
alert(function() {
```

```
/hb/i/sg/adv/infinity/generic_ysm_iframe.html?ysm_rd=http://sg.ard.yahoo.com/SIG`/M%20667619.13535967.1
3601179.13047309/D`/S%2080005941:LREC2/Y`/EXP%201262210734/L`.S1taSztQ20L1c3W.rks7spkAAM8j/B`-
/J%201262203534421418/K`.6bZGv6oA/A%205769597/R%200/*#7273153851086377751&ysm_url=http:
```

```
DUMMY_EXPR
```

```
});
```

The above mentioned code can be taken as prototype which correctly compiles and executes.

The development version is already out.

Development version

If you're brave enough and you need a specific feature or fix not released yet, or you simply want to provide feedback before official release, you may try this [unofficial development build](#).

Recent development history:

v 1.9.9.32

```
=====
+ Further InjectionChecker optimizations, providing a dramatic speed boost
  on nested URLs (e.g. on iGoogle and many ad networks)
```

v 1.9.9.31

```
=====
+ InjectionChecker accuracy optimization, preventing false positives in
  some edge cases with nested URLs (thanks Aditya K Sood for reporting)
```

Thanks.

All for security community purposes