symantec™ | Security Response

Firefox and Malware:
When Browsers Attack

Candid Wüest & Elia Florio

Contents

Abstract

Mozilla Firefox is a very popular browser and its open-design framework makes it easy to extend the functionality by either changing the core code directly or creating extension plug-ins that work on multiple operating systems.

As with browser helper objects for Microsoft Internet Explorer, Firefox extensions can also be misused to carry out malicious actions on the user's computer without the user's consent. Any installed extension has the same full rights as the browser itself and therefore can do a lot more than just display fancy Web pages. This includes accessing the file system in read and write mode, opening new network sockets and even creating new processes. This leads to a variety of security problems that can introduce or hide malicious code on a system. There have already been a number of cases where malware dropped malicious extensions or harmless extensions downloaded malicious code and the numbers are increasing, even full backdoor Trojans are possible this way. Furthermore, badly written extensions can be exploited through Web pages and therefore open new attack vectors.

This paper will highlight the security concerns with Firefox extensions and will show the methods that Firefox malware uses today.

Introduction

Web browsers have long ago changed from pure HTML rendering to versatile instruments for displaying media-rich and interactive Web content. To keep up with the fast changing demands of features that a browser should support, the idea of extending it with plug-ins or add-ons was developed. This is not a new idea. Microsoft's Internet Explorer has supported so-called browser helper objects (BHO) for many years and even Netscape's Navigator supported plug-ins. A study from Market Share shows Firefox had a total market share of 22.48% in the middle of 2009, with more than 90% being Firefox version 3.0 and higher.[1] With the growing popularity of the Mozilla Firefox browser; the community developing Firefox extensions has also grown. According to the Mozilla foundation, there are more than 12,000 extensions available and they counted more than 1 billion extension downloads so far.[2] Quite an irresistible target for a malware author, don't you think?

Scope of this paper

The focus of this paper lies with Mozilla Firefox extensions and not with plug-ins or themes; although similar security concerns apply to plug-ins and themes. Third party plug-ins; like the PDF reader, have been shown to be susceptible to vulnerabilities as well.[3] All tests were conducted with Firefox 3.0.10, if not noted differently.

There are also other Mozilla software packages besides Firefox, like Thunderbird, that can make use of extensions in similar ways, but this is outside the scope of this paper.

What are extensions?

Mozilla based software allows three primary types of add-ons: extensions, plug-ins and themes. Simplified browser extensions are small programs that extend the browser's feature set, for example by adding features for displaying multimedia Web content or managing general tasks like storing passwords for Web sites. In contrast to browser plug-ins most Firefox extensions are created by private users from the community, and are rarely digitally signed.

Firefox extensions can be written in JavaScript or in programming languages like C/C++ or Python. It is also possible to include native code binaries, if needed. Extensions can use the XML User Interface Language (XUL) and the Cross Platform Component Object Model (XPCOM) API in order to access some lower level system functionality. JavaScript can use the XPConnect interface to access XPCOM objects. The code files are compressed to an XPI file (cross platform install).[4] This abstraction layer allows the extension to be platform independent running on any operating system that Firefox is available on.

Mozilla recently released a new toolkit called Jetpack that will help to make it easier to create extensions for future versions of Mozilla software.[5] Time will show if this will introduce new attack vectors as well, but it will definitely boost the number of available extensions once more.

XPI file layout

Cross platform installer files (XPI) are normal zip compressed archives, typically containing the following files and folders:

• install.rdf	[installer file]
• install.js	[installer file]
• chrome.manifest	[settings file]
• chrome	[code folder]
• components	[plug-in folder]
• defaults	[preferences]

The content inside the chrome folder can sometimes be compressed into a zip file with the JAR extension. The browser will then, on the fly, extract the needed files.

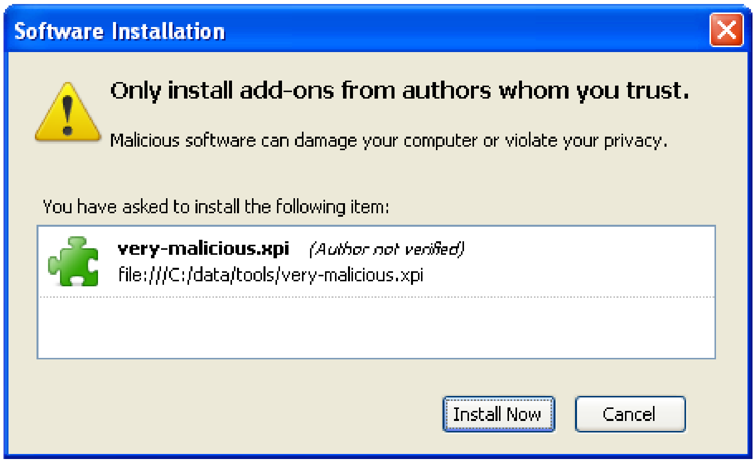
XPI extension files can also contain plug-in files with native code like .DLL files for Windows or .so files for Linux.[6] In general, they could contain any file the author wanted to include, such as malicious binary files, which can then be referenced from the extensions.

How Firefox extensions are installed

There are two main techniques to install Firefox extensions. Either by opening a linked or local XPI file; or by extracting the files directly into the corresponding folder by another local running program with access rights.[7] If the first method is used, a warning message as seen in Figure 1 is displayed, reminding the user that extensions could contain a malicious payload. After the installation, a restart of the browser is requested in order to load the extension.

Figure 1

XPI installation warning



Software Installation

Only install add-ons from authors whom you trust.

Malicious software can damage your computer or violate your privacy.

You have asked to install the following item:

very-malicious.xpi *(Author not verified)*
file:///C:/data/tools/very-malicious.xpi

If the files are extracted directly to the extension folder the changes will take place the next time the browser is restarted, but no software installation warning dialog is displayed.

There is also the following command line switch for Firefox that can be used to install new extensions globally:

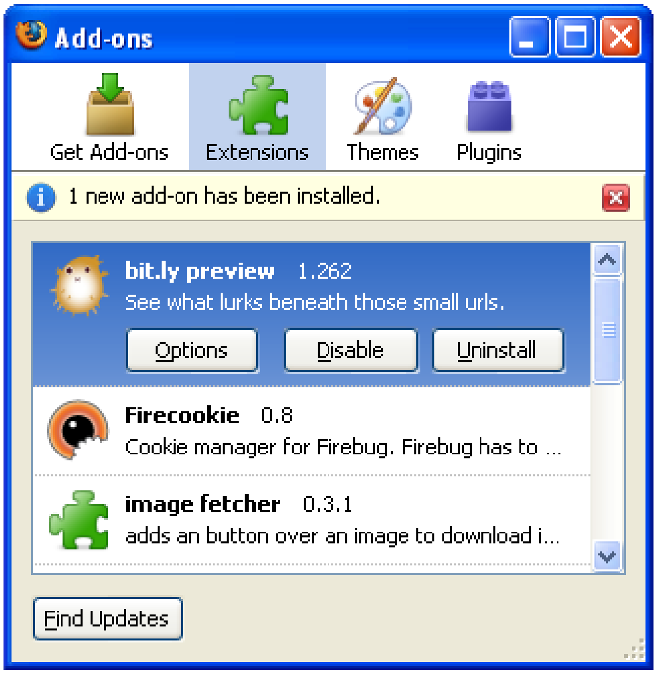
```
-install-global-extension “C:\some _ random _ extension.xpi”
```

A global installation will install an extension to the application directory rather than within a profile, so it will be available to all users.

Firefox version 3 and above will show an informational message when a new extension is installed as shown in Figure 2. This happens regardless of which of the above methods was used for installation.

Figure 2

“New add-on has been installed” information message



Add-ons

1 new add-on has been installed.

bit.ly preview 1.262
See what lurks beneath those small urls.

Firecookie 0.8
Cookie manager for Firebug. Firebug has to ...

image fetcher 0.3.1
adds an button over an image to download i...

To check which extensions are installed in Firefox, a user can look it up using the Windows menu tools under add-ons -> extensions or manually browse to one of the following folders on disk. The same locations can also be used to manually uninstall an extension if needed.

On Windows systems:

```
%UserProfile%\Application Data\Mozilla\Firefox\Profiles\[RANDOM].default\extensions
```

or

```
%ProgramFiles%\Mozilla Firefox\extensions
```

On a Mac OS X system:

```
/Library/Application Support/Mozilla/Extensions
```

The path might vary slightly depending on the version and installation options of Firefox and depending on if the extension was installed globally or only for one user.

On Windows systems, a user can also check for any additional extension folders referenced under one of the following registry keys:

- HKEY _ LOCAL _ MACHINE\SOFTWARE\Mozilla\Firefox\Extensions
- HKEY _ CURRENT _ USER\Software\Mozilla\Firefox\Extensions

Once an extension is successfully installed, it can check for updates periodically and install any newer version, if requested to do so. This behavior can be disabled under the option tag in the browser, but it is enabled by default. An extension can even cancel an impending removal by calling the cancelUninstallItem() function of the extension manager object on the notification of an unload event.

Possible issues

This chapter will highlight a couple of security concerns and issues that can occur with Firefox extensions.

Update attack

An innocent looking and clean extension can deliberately place a malicious update at its predefined update URL. This will then be downloaded and prompted for installation the next time the browser checks for updates, which by default is once a day. Even the average security-conscious person usually checks the extension only before installing it the first time, any update with malicious intent might slip through. This could also happen if any of the update sites get silently hijacked. Most Firefox extensions now share the updates over the official addons.mozilla.org domain, making it harder to hijack updates.

In April 2009, the widely used Firefox extension called NoScript, which can block Web scripts on a per domain basis, released a rather unusual update.[8] The author decided to make a small patch, after some preceding back and forth with another extension named Adblock Plus, which was designed to block advertisements, and did exactly this on the NoScript extension's home page. The modification to the NoScript extension contained, encoded in hexadecimal, some slightly obfuscated code that tampered with the other extension. The main goal was to show the advertisements on his domain regardless of the filter settings used by Adblock Plus. Without much information about the intention of modifying an external extension the user was left on his own. Furthermore, as the NoScript extension had a long and good reputation, it was released to the addons.mozilla.org repository without manual inspection, thus getting downloaded to many unsuspecting users. So even though the payload was not devastating, it should be taken as an example that even well known extensions could potentially at some point include malicious code in an update.

Browser versions prior to Firefox 3 were susceptible to man in the middle attacks during the update process itself as it was possible to have an update URL that was not an HTTPS URL. Therefore a man in the middle attack on the network could redirect the update request to a malicious update file and install a malicious extension. Many popular extensions were vulnerable to this attack. This has since been changed and the update URL must now either be HTTPS or contain a public key of the signed update information file in order to ensure integrity. The update.rdf file then contains hash values of the files to download.

Unintentional infections

In February 2008, the Vietnamese language pack add-on for Mozilla got infected by an HTML file infector.[9] The author's network or one of his computers was infected by this worm, which then modified the HTML help files to include a malicious script link. This modification was not noticed at Mozilla as at the time it got uploaded, the ClamAV scanner used did not detect this worm. There was no re-scanning of uploaded files afterwards and so it took two months before a user noticed and reported the infected language pack. Mozilla then quickly removed the offending add-on and cleaned it up. The offending version got blacklisted and all users were requested to upgrade. Mozilla has since changed their procedure of re-scanning for malicious code in uploaded code.

This shows that it is possible for malicious code to sneak into existing extensions and get potentially distributed to thousands of unknowing users.

Hide extensions

It is not always possible to view all installed extensions with the built-in functions. Extensions can use the tag word “hidden” in the install file in order to hide a global installed extension from the extension manager. Once installed, an extension can iterate through all the items listed in the Firefox extension manager object (@mozilla.org/extensions/manager) and simply remove itself from this list or use a cascading style sheet property to hide itself. Another method is to modify the extension.rdf file in the profile folder and remove itself from it or set the type to 0, in order to be invisible in the extension manager.

Of course as Firefox is an open source application, it is possible for a malicious extension to place a code hook at any suitable place in Firefox and there are many of them. For instance, an extra manifest file located in the `%ProgramFiles%\Mozilla Firefox\chrome` folder can start an XUL file which then runs like an extension, but is not listed anywhere. This will go unnoticed as there are no integrity checks on the code files. There are many root files for Firefox that could similarly be infected. They may even be used as obfuscated loading points for malware to execute normal binary files placed somewhere on the system.

These obfuscation tricks make it much harder to trace infections as these loading points are not listed under the tools->add-ons menu page and are not monitored for by many security software programs. If files of the core code base are modified, the malicious extension will run even if Firefox is started in safe mode.

Obfuscation

As we have seen from many JavaScript attack toolkits, a certain level of obfuscation can be achieved in JavaScript, allowing even some level of polymorphism. Encoding and splitting of keywords can make it difficult to analyze the code. The same techniques can obviously be applied to Firefox extensions coded in JavaScript in order to make it harder to analyze or to avoid detection by signatures.

Hijacking other extensions

XUL and XPCOM code can be directly injected into other Firefox extensions as a kind of parasitic extension infector, either by replacing the whole code file, adding a new malicious snippet or by changing the update URL to a location controlled by the attacker. New code can be added as self-contained files and be referenced in the chrome.manifest file or hidden and buried deep in the extension’s code.

This even works on signed extensions as the integrity of an extension is only checked on the first installation. Any later modification of the installed extension files is not checked against the digital signature. This technique was, for example, used by BrowserSpy which was able to infect the Google toolbar extension.[10] Such cross infections make it hard to trace and remediate infections. It would be imaginable that even double infections from multiple viruses could occur, which could result in mutations like we have seen with office macro viruses in the past. Fortunately installed Firefox extensions are not exchanged as frequently as office macro files were, so the chance of spreading through human distribution is probably marginal.

A minor case of extension hijacking is possible with extensions that rely on external scripts. The Greasemonkey extension is a good example of this. The extension allows user to create scripts that can be applied and executed to specific domains. This allows the script to locally modify the appearance of Web sites. If users do not carefully check scripts that they download, they might end up with malicious scripts running in Greasemonkey. There are several reports that malicious user scripts for popular sites have been distributed on the Internet[11] or used to carry on advanced phishing attack against social networking sites.[12]

Browser overlay

Extensions can make use of so called overlays to extend or modify the appearance of the browser. Unfortunately, this also means that security relevant dialogs or setting forms can be changed. For example, it would be easy for a malicious extension to remove any phishing warning or SSL certificate mismatch prompt that the browser generates.

Vulnerable extensions

In April 2007, multiple vulnerabilities were found in the popular Firefox extension called Firebug.[13] If an attacker placed specially crafted code on a Web page that was viewed by a user with Firebug enabled, he could execute script code under the chrome context. Chrome refers to the set of user interface elements of the browser that are outside of a window's content area. This includes toolbars, menu bars and progress bars. Running arbitrary code with chrome privileges allows many powerful operations as highlighted below in "Potential Damage". This includes silently downloading files and executing local binaries. Mozilla explains chrome privileges as follows: "The code running with chrome privileges is allowed to do everything, unlike the web content, which is restricted in several ways."[14]

Similar vulnerabilities could be deliberately planted into a popular extension by a mischievous author. Such planted bugs can be very hard to spot. This would allow an attacker to place exploit code in common Web pages. For example they could put a special tag with a URL that advises the extension to download and install the binary behind the URL. On the other hand, an extension would need to be quite popular in order for the attack to scale, unless of course it is a targeted attack for a small group.

Potential damage

Once an extension is installed, it is granted the same privileges as Firefox is running. Through XPCOM, the extension can access the local file system, write to files and create network sockets. Regardless of the type of operating system it is on.[15]

Needless to say, it has full control over the browser and can change any appearance or the rendering of Web sites. An extension can read out stored passwords, access the clipboard, check memory usage, or browser history. When running on Windows systems, it can even access and modify the Windows registry. This makes it a favored candidate for Adware and Spyware risks or Infostealer threats.[16]

Through the nsILocalFile interface and the launch() function or the nsIProcess interface and the run() function, local binaries can be executed, spawning new processes. There are other APIs available to control processes even further, like the nsIProcess2 interface. This could be used as a hidden loading point for malware that is present on the computer or to start binary files that were hidden in the XPI extension archive.

There are various events available that can be used as triggers; for example, the loading of the browser, the rendering of a new Web site or the closing of a tab.

With more than 12,000 extensions freely available, there exist examples for nearly all common computer tasks.[17] For instance, there are password managers and local Web servers which potentially could be modified to run invisibly and steal information from the user.

As a result, it is next to nothing for an attacker to create a malicious extension with full control over the system and access to Web traffic before it gets encrypted by SSL.[18]

Mitigation strategies

This chapter will illustrate some of the processes in place to minimize the discussed risks. Obviously there are many possible ways to increase security and this list is by no means meant to be exhaustive.

AMO Publishing guidelines

There exists a Mozilla reviewer's guide which highlights some basic security checks that should be performed before new extensions are added to the Mozilla repository.[19] This includes checking for any obfuscated code, loading of remote components or heavy use of binary files.

New and freshly added extensions on addons.mozilla.org (AMO) are placed in a so-called sandbox. The term sandbox might be misleading here as it is more like a beta tester program. The extensions in the sandbox are not visible for normal visitors by default, but any user can explicitly browse them and install any of them. They are simply less tested. Over time an extension can get reviewed by administrators and moved from the sandbox into the public realm. Once an extension has established a trust level, new updates will become public without an immediate review. As we have elaborated in the Update Attack section, this is not a foolproof mechanism and could potentially be misused. It should further be noted that the above-mentioned guidelines are only applied if the extension is hosted on addons.mozilla.org. If the author decides to host the extension on his own Web site, no guarantee is given for any checks.

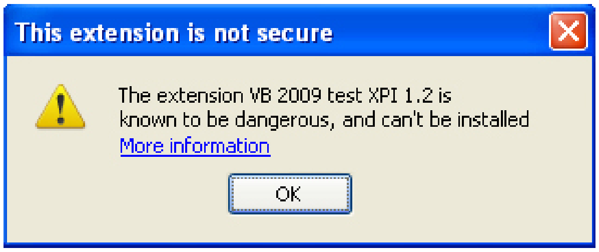
Security mechanisms

Mozilla incorporated a local revocation list or blacklist of bad add-ons. The local file blocklist.xml contains IDs and version numbers of known bad extensions which are blocked from installing. Currently the list contains 7 items including the infected language pack mentioned in the Unintentional Infections section. As this list works by matching names or GUIDs, it should be clear that a malicious extension could simply randomize the name to get past this blacklist. But the list is helpful in preventing legitimate extensions which have been found to be unsafe or buggy from being installed by an unaware user.

There exists a project from the University of Illinois at Chicago which introduces the concept of validating extension integrity by user signing and enforcing behavior policy to limit the potential damage.[20] These or similar approaches could massively improve the security around Firefox extensions as they would limit their power over the system or restrict their behaviour.

Figure 3

Blacklisted extension warning



This extension is not secure

The extension VB 2009 test XPI 1.2 is known to be dangerous, and can't be installed

Firefox risks examples

This chapter analyzes a selection of the current Firefox extensions and malware that we have seen in the wild and compares their feature set. There are many other examples like the “eBay captcha populator” or “office poltergeist”, that could be mentioned here as well.

JS.FFsniFF

One of the earliest malicious extensions for Firefox, which was widely distributed in the wild in 2006, is the FFsniFF extension.[21] This is a simple extension which must be installed manually by the user or dropped by a Trojan. Once installed, it can hide itself from the extension manager by setting its install attribute to hidden and removing itself from the visible item list in the extension manager object.

It then starts monitoring all form submissions in the browser and if any of them contains a password field, it logs all data and sends an email with the gathered data to a predefined account. This is a very simple but still effective way of stealing static logon credentials for many Web sites.

During the infection, the following files are created in the user’s Firefox profile folder under the GUID folder of FFsniFF.

```
{66cdf40a-d0f2-46d0-abf4-eccba8205aef}
 ├──chrome
 │  └──ffsniff.jar
 │     └──content
 │        └──ffsniff
 │           ├──contents.rdf
 │           ├──ffsniffOverlay.js
 │           └──ffsniffOverlay.xul
 ├──chrome.manifest
 └──install.rdf
```

Trojan.Brojack

Trojan.Brojack drops a Firefox extension into the global extension folder.
`%ProgramFiles%\Mozilla Firefox\extensions\sotfone-tracker@sotfone.ru`

The extension hides from the browser by setting <em:hidden>true</em:hidden> in the install.rdf file, so that it will not be listed. It also creates a DLL that is then used as a BHO in Internet Explorer.

Once installed, the Trojan captures all URLs visited by the user in Internet Explorer and Mozilla Firefox and submits the data back to a remote server.[22]

Infostealer.Snifula

Infostealer.Snifula masquerades as the legitimate extension "NumberedLinks v0.9". The threat comes with a dropper that installs the extension into the Firefox profile folder, but also drops Windows binaries and registers them through the usual registry run key load point.

The extension does not need to hide from the extension manager as it impersonates a legitimate extension. The payload of this threat is to log entered passwords and confidential information and send them to a remote server.[23]

Adware.MyCentria

A component of Adware.MyCentria is a Firefox extension installer called InstallerFF.exe, which drops a Firefox extension named AdCentria Infobar into the user's profile folder.[24] This extension will then register three event listeners in Firefox. One event listener registered on DOMContentLoaded is able to check any new loaded page. Further event listeners registered on the select event and on the click event are able to hijack control when needed. With these triggers set, it monitors for a predefined list of Web sites and when visited can display its own ads that it downloaded with standard XMLHttpRequest() calls. It has also been reported that MyCentria was targeting popular search engines in order to modify on-the-fly the results from search queries.

The following snippet is used by the extension to add an additional user agent to the browser which can be used to identify if the requests are coming from an installed Infobar or not.

```
modify_ua : function()
{
  adcentria.set_str("general.useragent.extra.
  adcentriaim","AdCentriaIM/"+adcentria.ver)
  adcentria.main()
}
```

MyWebSearch Toolbar

Some variants of the MyWebSearch toolbar install a component directly into the Firefox core code base by dropping the following two files:

- `[FIREFOX PATH]/chrome/m3ffxtbr.jar`
- `[FIREFOX PATH]/chrome/m3ffxtbr.manifest`

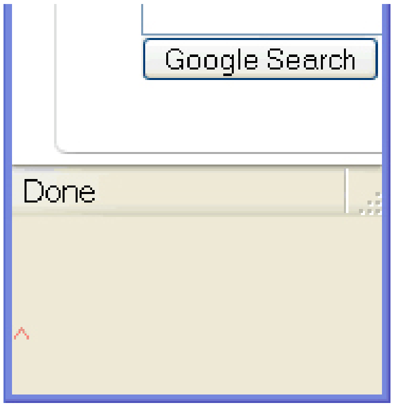
The manifest file contains the following two lines that will install and run the toolbar:

```
content m3ffxtbr jar:m3ffxtbr.jar!/
xpcnativewrappers=yes
overlay chrome://browser/content/browser.xul chrome://
m3ffxtbr/content/menu.xul
```

This makes the installment invisible for normal users as it will not show up in the extension manager and cannot be disabled by starting Firefox in safe

Figure 4

Grey status error bar



mode. There is also no warning of "new extension being installed"; as technically it is not installed as an extension, it just modifies the browser directly.

Since there was a coding error in a previous version of this toolbar application many people saw a broken Firefox toolbar and wondered where it came from. This issue was even so widespread that Mozilla had to mention it on their support forum and made a patch to remove these modification manifest files on the next Firefox code update.[25]

Trojan.Hanambot

This Trojan drops the malicious extension file amba.jar into the `[FIREFOX PATH]/chrome/` folder and then adds the following two lines to the browser.manifest which is located in the same folder:

- `content amba jar:amba.jar!/`
- `overlay chrome://browser/content/browser.xul chrome://amba/content/amba.xul`

This references the file amba.xul that is compressed in the amba.jar file. The .xul file then starts executing the JavaScript file, through the following call:

```
<?xml version="1.0"?>
<overlay id="pamba-overlay" xmlns = "http://www.mozilla.org/keymaster/gatekeeper/there.
is.only.xul">
<script type="application/x-javascript" src="amba.js"/>
</overlay>
```

Next, the malware deletes all saved cookies to provoke new logins. Then it monitors any new loaded page and checks with regular expressions if they match any of the predefined financial sites. If a site of interest is found, the Trojan logs the submitted credentials and cookies and also grabs the current balance and last login. The harvested data is then sent back to a remote server.[26]

The following snipped shows an example of an URL it monitors and the regular expressions to match the current balance and last log-in data.

```
['www.pay***.com/us/cgi-bin/webscr', /U.S.Dollar[\d\D]{1,64}(\$[\d\,\.]+)/gmi, /Last log
in (.*?)<\/span>/gmi,'pay***', true]
```

The Trojan stores configuration settings in the Windows registry and synchronizes them from the JavaScript. Besides the malicious extension, Trojan.Hanambot also installs a rootkit and registers a binary file in the current user run key. This enables the Trojan to receive bot commands outside of Firefox.

Adware.Purityscan

Adware.Purityscan installs a Firefox extension component into the following non-standard folder:

```
%ProgramFiles%\Outerinfo\FF\
```

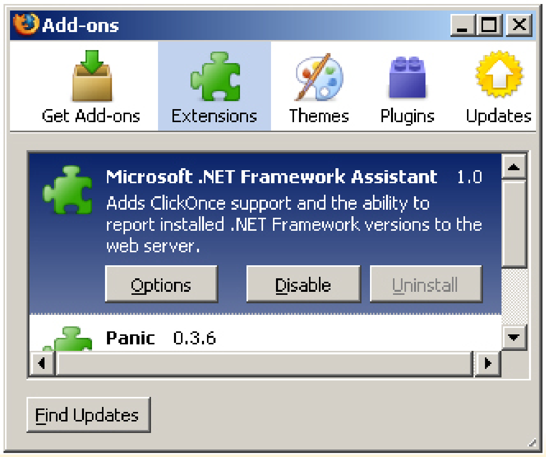
In there it creates an empty chrome.manifest file; empty because it does not make use of any XUL overlays. Next it creates an install.rdf file with the basic information about it and the folder components containing the FF.dll and OuterinfoAds.xpt. The XPT file is an interface description file generated by the XPIDL compiler. This allows the FF.dll to be loaded.

The extension component is registered through a link in the Windows registry:

```
HKEY_CURRENT_USER\Software\Mozilla\Fire-
fox\Extensions\"{59A40AC9-E67D-4155-B31D-
4B7330FCD2D6}" = "%ProgramFiles%\Outerinfo\FF\"
```

Figure 5

Greyed-out uninstall button



Microsoft .NET Framework Assistant 1.0

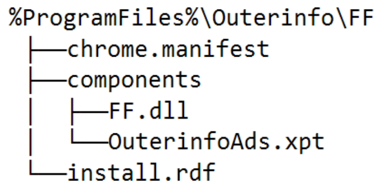
Adds ClickOnce support and the ability to report installed .NET Framework versions to the web server.

Panic 0.3.6

Registering an extension through the afore-mentioned registry key does gray-out the uninstall button in the extension manager GUI. Consequently, a user can only disable it but not remove it in the GUI. The user has to remove it manually, outside of Firefox.

The same trick was used by a Microsoft update of the .NET components in 2009 that installed a Firefox extension called "Microsoft .NET Framework Assistant 1.0". As many users got upset that they were not able to normally uninstall this extension, Microsoft decided to provide a new update which installed the extension per user so that it can be uninstalled from within the browser.[27]

As reference, here are the extension files created during the installation.



```
%ProgramFiles%\Outerinfo\FF
  ├──chrome.manifest
  ├──components
  │   ├──FF.dll
  │   └──OuterinfoAds.xpt
  └──install.rdf
```

Besides the extension, Adware.Purityscan also installs a BHO for Internet Explorer and performs other system changes not related to Firefox.[28] The adware can download and display advertisements in the browser.

Trojan.Chromeinject.A

Chromeinject is a family of Trojans discovered during the end of 2008. This family shows a big variety of samples in the wild after a short period of life and this fact probably makes this malware the most common banking Trojan targeting the Firefox browser today. Trojan.Chromeinject is installed by a binary executable below 30 KB in size which commonly drops the following files:

- %ProgramFiles%\Mozilla Firefox\chrome\chrome\content\browser.js
- %ProgramFiles%\Mozilla Firefox\chrome\chrome\content\browser.xul
- %ProgramFiles%\Mozilla Firefox\plugins\npbasic.dll
- %ProgramFiles%\Mozilla Firefox\plugins\npbasic.dll1

The Trojan is composed of a malicious code written in JavaScript which works in combination with a DLL plug-in; both components are silently installed into Firefox by writing directly into the Firefox folders. The extension registers two event listeners in Firefox. Again, one event listener registered on DOMContentLoaded is able to check any new loaded page.

```
function init()
{
window.addEventListener("load",chromeLoad,false);
}
function chromeLoad()
{
var appContent = document.getElementById("appcontent");
appContent.addEventListener("DOMContentLoaded",contentLoad,false);
}
```

When the load event gets triggered, the listener callback routine installed by the Trojan will enumerate all the available IFRAMEs in the web page. For each of them it will inject an EMBED tag which forces the browser to load the malicious DLL plug-in. The code injection is performed only for a Web page which matches a specific check based on a list of URLs and domain names. The following JavaScript code is responsible for the injection:

```
function proc _ ff(cc)
{
        var doc=cc.document;

 var loc=doc.location.href;
 var doi=0;
 if(check(loc,doc.domain))doi=1;
 if(doi==1)
{
 doc.body.innerHTML=’<div style=”width: 100%;height: 100%;position: absolute; left: 0px; 
top: 0px;”>’+
 ‘<embed type=”application/basic-plugin” ‘+
 ‘ width=”100%” height=”100%” param-location=”’+doc.location.href+’”></embed>’+
 ‘</div>’;
}
}
function contentLoad()
{
 if(content.frames.length>1)
{
  var a=content.document;
  modify(a);
  for(var i=0;i<content.frames.length;i++)proc _ ff(content.frames[i]);
}
 proc _ ff(content);
};
```

The odd feature of this Trojan is the usage of JavaScript regular expression functions to parse URLs searching for domain names of popular bank websites. Below follows a snippet of this function, where original bank names have been masked:

```
function check(loc,dom)
{
 var domains=[‘127.0.0.3’,’127.0.0.2’];
 var urls=[‘das’,’http://127.0.0.2/’];
 var urlsr=[‘yandeeex.ru’,’sss.re’];

var zurl=[‘*a[****].com*’,
‘*[****]ell.net*’,
‘*cre[****].it*’,
‘*area[****]if.es*’,
‘*ban[****]en.es*’,
‘*o[****]ank.es*’,
‘*pos[****]t*’,
‘*ban[****]s*’,

...[list continues]
```

The fact that the malicious DLL plug-in gets loaded into Firefox’s memory only when the user is surfing certain bank Web sites makes forensic analysis in memory more complicated. A first look to a fresh Firefox process and its loaded DLLs will in fact not reveal the malicious plug-in in memory.

Conclusion

The Web browser is the main tool used to communicate with the Internet and for that reason usually has all rights to pass the firewall or other local security products. Furthermore, a lot of interesting data like passwords or credit card numbers are submitted through the Web browser. Hence it is an ideal target for attackers, especially for adware, spyware, or infostealer tools. With an estimated market share of 22% Firefox is wide spread enough to become a target.

The phenomenon of malicious Firefox extensions is nothing new. Similar malware has been seen with Internet Explorer for many years. With Firefox, we have seen a few spyware using extensions back in 2005, but the numbers have been growing since. Creating new extensions is very easy as there are many how to examples and it can be coded in languages like JavaScript with easy access to all interesting browser functions. As most extensions are created by private users and not digitally signed, people are accustomed to installing unsigned extensions.

We should also keep in mind that Firefox extensions can be cross-platform functional, meaning that they can operate on Windows, Linux or Mac OS X machines; increasing the number of possible victims.

Although we have not seen any cross-platform bot agent extensions yet, it could very well be possible for an attacker to create such an extension. As extensions can control the full browser, any information window can be modified by overlays and therefore any prompts or settings cannot be trusted. Modifying information displayed by online banking Web sites is one example and an easy one to achieve.

As the activity is coming from the Firefox process itself, a behavior-based analysis system needs to be smart to identify that it is a malicious extension performing these actions and not convict the browser. This can make remediation very tricky, especially when core code files of Firefox are modified and need to be repaired.

We expect that we will see an increase in malicious extensions in the future. Most likely this will be malware that can install both browser helper objects for Internet Explorer as well as extensions for Mozilla Firefox, since we have started to see this trend already.

Acknowledgement

The authors want to thank Liam O’Murchu for helpful ideas and discussion.

Appendix

Comparison matrix

If we compare the most interesting attributes of the different extension samples, then we see that they vary quite a bit depending on their goal set.

Table 1

Attributes Comparison Matrix

Comparison of different features of extensions

Legend: √ = feature is present (√) = feature is attempted “—” = feature is not present	JS.FFsniFF	Trojan.Brojack	Infostealer.Snifula	Adware.MyCentria	MyWebSearchToolbar	Trojan.Hanambot	Adware.Purityscan	Trojan.Chromeinject.A
Hides from extension manager	√	√	(√)	—	√	√	(√)	√
Hides “new installation” warning	—	—	—	—	√	√	—	√
Steals confidential data	√	√	√	—	(√)	√	(√)	√
Installed by a dropper	—	√	√	√	√	√	√	√
Infects other extensions	—	—	(√)	—	—	—	—	—
Installs into FF core code base	—	—	—	—	√	√	—	√
Uses non-default extension folder	—	—	—	—	√	√	√	√
Contains binary files	—	√	√	—	—	√	√	√
XPI allows full remote control	—	—	—	—	—	—	—	—

References

1. "Browser market share", Market Share, June 2009.
http://marketshare.hitslink.com/browser-market-share.aspx?qprid=0

2. "1 Billion Add-Ons", Mozilla blog, Nov. 2008.
http://blog.mozilla.com/addons/2008/11/19/1-billion-add-on-downloads/

3. "Symantec Internet Security Threat Report XIV", Symantec, April 2009.
http://www.symantec.com/threatreport

4. "XML User Interface Language", Mozilla developer center, 2008.
https://developer.mozilla.org/En/XUL

5. "Jetpack", Mozilla labs, May 2009.
https://jetpack.mozillalabs.com/

6. "Shipping a plugin as an extension", Mozilla developer center, 2008.
https://developer.mozilla.org/en/Shipping_a_plugin_as_an_extension

7. "Installing Extensions", Mozilla developer center.
https://developer.mozilla.org/en/Installing_extensions

8. "Dear Adblock Plus and NoScript Users, Dear Mozilla Community",Giorgio Maone, May 2009.
http://hackademix.net/2009/05/04/dear-adblock-plus-and-noscript-users-dear-mozilla-community/

9. "Firefox infects", Ryan Singel, May 2008.
http://www.wired.com/threatlevel/2008/05/firefox-infects/

10. "Extensible Web Browser Security", M. Louw et al., July 2007.
http://www.mike.tl/view/Research/ExtensibleWebBrowserSecurity

11. "Cookie stealing scripts", userscripts.org forum, June 2007.
http://userscripts.org/topics/704

12. "The Orkut Worm – Digging Deeper", Silas Barnes, Feb. 2008.
https://forums2.symantec.com/t5/blogs/blogarticlepage/blog-id/malicious_code/article-id/196#M196

13. "Firebug goes evil", pdp GNUCITIZEN, April 2007.
http://www.gnucitizen.org/projects/firebug-goes-evil

14. "Chrome", Mozilla developer center, 2008.
https://developer.mozilla.org/en/Chrome

15. "Browser rootkits", Christophe Devaux et. Al., Oct. 2008.
http://2009.hack.lu/archive/2008/rootkits-navigateurs.pdf

16. "Malicious Firefox Extensions", Philippe Beaucamps, Daniel Reynaud, June 2008.
http://lhs.loria.fr/index.php?option=com_content&view=article&id=62:malicious-firefox-extensions&catid=36:news&Itemid=54

17. "1 Billion Add-Ons", Mozilla blog, Nov. 2008.
http://blog.mozilla.com/addons/2008/11/19/1-billion-add-on-downloads/

18. "Extending good and evil", Candid Wüest, Oct. 2006.
https://forums2.symantec.com/t5/blogs/blogarticlepage/blog-id/emerging/article-id/16#M16

19. "Editors/Reviewing Guide", Mozilla wiki, March 2009.
https://wiki.mozilla.org/Update:Editors/ReviewingGuide

20. "Extensible Web Browser Security", M. Louw et al., July 2007.
http://www.mike.tl/view/Research/ExtensibleWebBrowserSecurity

21. "FFsniFF (FireFox sniffer)", Azurit, June 2008.
http://azurit.elbiahosting.sk/ffsniff/

22. "Trojan.Brojack", Symantec, July 2008.
http://www.symantec.com/security_response/writeup.jsp?docid=2008-070310-5229-99&tabid=2

23. “Infostealer.Snifula”, Symantec , July 2006.
http://www.symantec.com/business/security_response/writeup.jsp?docid=2006-072610-2145-99&tabid=2

24. “Adware.MyCentria”, Symantec, Dec. 2008.
http://www.symantec.com/security_response/writeup.jsp?docid=2008-121112-1557-99&tabid=2

25. “Gray bar below status bar”, Mozillazine KB, Feb. 2009.
http://kb.mozillazine.org/Gray_bar_below_status_bar

26. “Trojan.Hanambot”, Symantec, July 2009.
http://www.symantec.com/security_response/writeup.jsp?docid=2009-060121-0427-99&tabid=2

27. “Update to .NET Framework 3.5 SP1 for the .NET Framework Assistant 1.0 for Firefox”, Microsoft Download Center, May 2009.
http://www.microsoft.com/DownLoads/details.aspx?displaylang=en&FamilyID=cecc62dc-96a7-4657-af91-6383ba034eab

28. “Adware.Purityscan”, Symantec, Feb. 2007.
http://www.symantec.com/de/de/security_response/writeup.jsp?docid=2003-090516-2325-99&tabid=2

Originally published by Virus Bulletin Conference, September 2009. Copyright held by Virus Bulletin, Ltd., but is made available courtesy of Virus Bulletin.
For more information on Virus Bulletin, please visit:
http://virusbtn.com/

NO WARRANTY . The technical information is being delivered to you as is and Symantec Corporation makes no warranty as to its accuracy or use. Any use of the technical documentation or the information contained herein is at the risk of the user. Documentation may include technical or other inaccuracies or typographical errors. Symantec reserves the right to make changes without prior notice.

About Symantec

Symantec is a global leader in providing security, storage and systems management solutions to help businesses and consumers secure and manage their information. Headquartered in Cupertino, Calif., Symantec has operations in more than 40 countries. More information is available at www.symantec.com.

About the authors

Candid Wüest is a senior engineer at Symantec Security Response.

Elia Florio is an engineer with the Data Protection Authority in Italy.

For specific country offices and contact numbers, please visit our Web site. For product information in the U.S., call toll-free 1 (800) 745 6054.

Symantec Corporation
World Headquarters
20330 Stevens Creek Blvd.
Cupertino, CA 95014 USA
+1 (408) 517 8000
1 (800) 721 3934
www.symantec.com