# Firefox

Firefox is a popular web browser that many users have installed on their machines. The possibility of an attack from within the browser is a possibility. As such, this tool should enumerate various Firefox loading points to find suspicious information (if Firefox is installed).

This specification is targeted for Firefox 10.0 x86 builds.

## Log Output

Firefox should have its own section in the log labeled “Firefox.”

Each component of the Firefox section is listed in the subsequent sections. The components should appear in the log in the order that they appear in this document. Each line in the Firefox section should be prefixed with “FF -“ (this convention is followed in subsequent sections).

## Profile

The user profile folder is where most settings are located. On Windows 2000, XP, Vista, and Windows 7, the profile can be found in “%APPDATA%\Roaming\Mozilla\Firefox\Profiles\<profile folder>”. There can be several profiles installed for each user. All profiles in this directory with randomly generated folder names of 8 characters. The profile that starts up when Firefox loads will be in the randomly named folder appended by “.default”. For the first release, this tool should only enumerate the default profile for the currently logged in user.

### Log Output

The path to the default profile should be logged using the following format:

FF – ProfilePath - <profile path>

### Scripting

There are no applicable script commands for this component.

## Preferences

User preferences are located in the “<profile path>/prefs.js” file. There are several different “loading points” from this file that could be important for diagnosing an infection:

1. browser.startup.homepage
2. browser.search.selectedEngine
3. network.proxy.type

### Log Output

Each of the loading points should be logged using the following format:

FF – Preference - <loading point> - <loading point value>

### Scripting

Scripts should be able to reset user preferences. To do this, the keyword “:firefoxpreferences” should be specified. This keyword expects one argument which is the full name of the path to the profile to be operated on. The syntax is:

:firefoxpreferences <profile path>

After this, the script can list full preference names to be reset. These preferences do not necessarily need to be one of the loading points listed above. The action that will be taken is to simply delete the key from “prefs.js”. Note that to do this, Firefox must not be running. If it is running when the script comes across this action, the tool should open a warning dialog that Firefox will be terminated. Once the user acknowledges this fact, Firefox should be killed and the preferences file should be modified.

This is considered a system-altering action.

## Extensions

Firefox is very extensible, and allows third parties to provide “extensions” (sometimes called “addons”) that users can install into their profile. Malicious extensions exist, and therefore should be enumerated. Extensions are stored in a sqlite 3 database in “<profile path>/extensions.sqlite”.

### Log Output

Each extension should be logged using the following format (line break is unintentional and a side effect of the paper margins):

FF – Extension - <visible>|<active> - <name> - <version> - <type> - <id> - <path>

These values can be obtained from the following locations in the database:

* **visible** – addon.visible attribute. This should be a binary 0 or 1 value.
* **active** – addon.active attribute. This should be a binary 0 or 1 value.
* **name** – locale.name attribute where locale.id = addon.defaultLocale
* **version** – addon.version attribute
* **type** – addon.type attribute
* **id** – addon.id attribute
* **path** – addon.descriptor attribute

## Plugins

Firefox, like most major browsers, supports plugins. Plugins maintains a list of plugins in “<profile path>/pluginreg.dat”.

### Log Output

Each plugin should be logged in the following format:

FF – Plugin - <name> - <version> - <path>

The architecture of the plugin registry must be explained to be able to extract the require values. Everything below the line labeled “[PLUGINS]” is important for this tool. Each plugin is listed below this line. Plugins each have their own “blocks” that are structured similarly:

<file name>|$  
<full file path>|$  
<version>|$  
<installed timestamp>|<unknown?>|<unknown?>|$  
<description>|$  
<name>|$  
<mime type count>  
0|<mime type>|$  
1|<mime type>|$  
…  
<mime type count minus 1>|<mime type>|$

Each plugin “block” is put immediately next to other “blocks” with no other separation.

It should be enough to read through to the “[PLUGINS]” line and keep parsing plugin “blocks” until the “[INVALID]” line is reached.

## Components