

# [NEWS] Firefox Popup Blocker Allows Reading Arbitrary Local Files

---

*Source:* <http://www.derkeiler.com/Mailing-Lists/Securiteam/2007-02/msg00012.html>

---

- *From:* SecuriTeam <[support@xxxxxxxxxxxxxx](mailto:support@xxxxxxxxxxxxxx)>
  - *Date:* 5 Feb 2007 17:52:53 +0200
- 

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>  
-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.  
<http://www.securiteam.com/maillinglist.html>

-----

## Firefox Popup Blocker Allows Reading Arbitrary Local Files

---

### SUMMARY

There is an interesting vulnerability in the default behavior of Firefox built-in popup blocker. This vulnerability, coupled with an additional trick, allows the attacker to read arbitrary user-accessible files on the system, and thus steal some fairly sensitive information.

### DETAILS

Vulnerable Systems:

- \* Firefox version 1.5.0.9

For security reasons, Firefox does not allow Internet-originating websites to access the file:// namespace. When the user chooses to manually allow a blocked popup however, normal URL permission checks are bypassed. The attacker may fool the browser to parse a chosen HTML document stored on the local filesystem, and because Firefox security manager treats all file:/// URLs as having "same origin", such a document could read other local files at its discretion with the use of XMLHttpRequest, and relay that information to a remote server.

## [NEWS] Firefox Popup Blocker Allows Reading Arbitrary Local Files

Now, to make the attack effective, the attacker would need to plant a predictably named file with exploit code on the target system. This sounds hard, but isn't: Firefox sometimes creates outright deterministic temporary filenames in system-wide temporary directory when opening files with external applications; even if we ignore this possibility (since it requires the user to take an additional step that may be difficult to justify), "random" temporary files are created using a flawed algorithm in `nsExternalAppHandler::SetUpTempFile` and other locations.

The problem here is that `stdlib` linear congruential PRNG (`srand/rand`) is seeded immediately prior to file creation with current time in seconds (actually, microseconds, but divided by  $1e6$ ); `rand()` is then used in direct succession to produce an "unpredictable" file name. Normally, were the PRNG seeded once on program start and then subsequently invoked, results would be deterministic, but difficult to blindly predict in the real world; but here, the job is much easier: we know when the download start, we know what the seed would be, and how many subsequent calls to it are made – we know the output.

In a different setting, there would be a level of uncertainty caused by the fact that system clocks tend to drift or have imprecise settings (although today, most Windows systems either synchronize with Windows Time, or domain time services, so this is less of a factor). Still, there's a yet another nail to the coffin: on first call, Javascript `Math.random()` is seeded using the same call in the same manner, `PR_Now() * 1e-6`. The seed, and hence a time very close to the moment of file creation, can be trivially computed by analyzing `Math.random()` output. But wait, why bother at all – Javascript can be used to directly read system clock with a 1-second resolution.

One of several attack scenarios Michal could think of might look as follows:

1) Have user click on a link on a malicious page. The link would point to "evil.cgi", and have `onClick` handler set to function `foo()`. This function would acquire current system time, and use `setTimeout` to invoke `window.open("p2.html?" + curtime, "new", "");` in 100 ms. The aforementioned `cgi` script would return:

```
Content-type: text/html
Content-disposition: attachment; filename="foo.html"
```

```
<html><body><script>
x = new XMLHttpRequest;
x.open("GET", "file:///c:/BOOT.ini", false);
x.send(null);
alert("The script attempted to read your C:/BOOT.ini:\n\n"
+ x.responseText);
</script>
```

2) After user clicks the link, a download prompt will appear, and a copy of `evil.cgi` output would be saved in – for example –

## [NEWS] Firefox Popup Blocker Allows Reading Arbitrary Local Files

C:\WINDOWS\TEMP\c3o89nr7.htm. The download prompt will be immediately hidden under the newly created p2.html window (this, by default, bypasses popup blocker. because the window is created in response to user action).

3) The page currently displayed on top, p2.html, instructs the user to accept the popup to open a movie player or whatnot; since unsolicited popups are an annoyance, not a security risk, even an educated user is likely to comply.

To create a popup warning, a script embedded on the page calls:  
window.open('file:///c:/windows/temp/xxxxxxx.htm','new2',"),

with a name calculated by repeating a procedure implemented in SetUpTempFile() with a seed calculated by the server based on reported system time (p2.html?time).

4) When the user opens that particular popup, attacker-supplied HTML file is loaded and executed with local file read privileges (in the aforementioned example, the contents of BOOT.ini file would be reported back to the victim).

### ADDITIONAL INFORMATION

The information has been provided by <<mailto:lcamtuf@xxxxxxxxxxxxx>> Michal Zalewski.

=====

This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

list-unsubscribe@xxxxxxxxxxxxx

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@xxxxxxxxxxxxx

=====

=====

### DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.