virus BULLETIN Covering the global threat landscape

ALL YOUR MEETINGS ARE BELONG TO US: REMOTE CODE EXECUTION IN APACHE OPENMEETINGS

Andreas Lindh
Recurity Labs, Sweden

As part of my personal 'Hacking Open Source Software for Fun and Non-Profit' project [1], I took a look at the latest (at the time of writing) version of *Apache OpenMeetings*, 3.0.7. The findings discussed in this article have also been confirmed to work on version 3.1.0, released after this article was researched, and have been patched in version 3.1.1, which is available at [2].

According to *Wikipedia* (there is no comprehensive description on the official project website [3]), *OpenMeetings* is:

> '…software used for presenting, online training, web conferencing, collaborative whiteboard drawing and document editing, and user desktop sharing. The product is based on OpenLaszlo RIA framework and Red5 media server, which in turn are based on a number of open source components. Communication takes place in virtual "meeting rooms" which may be set to different communication, security and video quality modes. The recommended database engine for backend support is MySQL. The product can be set up as an installed server product, or used as a hosted service.
>
> 'Work on OpenMeetings started in 2006, and it has been downloaded over 250,000 times. OpenMeetings is available in 31 languages.' [4]

One particularly interesting aspect of *OpenMeetings* from a security standpoint is that it is exposed to the Internet by design, as one of the main features it offers is the ability to have virtual meetings with external parties.

During my audit of the program code, I came across multiple issues of varying severity, among which were two vulnerabilities that, with some additional trickery, would allow for an unauthenticated attacker to gain remote code execution on the system, with knowledge of an administrator's username as the only prerequisite. The following is a brief write-up of those issues and the path to code execution.

ANOTHER ZIP ARCHIVE PATH TRAVERSAL (CVE-2016-0784)

One of the first things I found was a bug similar to one that I found in *Apache Jetspeed 2* [5], namely a ZIP archive path traversal. Just like in the *Jetspeed* case, the bug exists in a function that requires administrative privileges to access, and the issue lies in the fact that the names of files in ZIP archives are not checked for dangerous character sequences before being written to disk.

The bug exists in the Import/Export System Backup function in the *OpenMeetings* administrative menu. The vulnerable code is shown in Figure 1. Notice the missing name check on line 217 and onward.

However, as *OpenMeetings* does not support JSP files in its default configuration, achieving code execution will not be as straightforward as it was in the *Jetspeed* case.

```
209         ZipInputStream zipinputstream = new ZipInputStream(is);
210         ZipEntry zipentry = zipinputstream.getNextEntry();
211         while (zipentry != null) {
212             String fName = zipentry.getName();
213             if (File.pathSeparatorChar != '\\' && fName.indexOf('\\') > -1) {
214                 fName = fName.replace('\\', '/');
215             }
216             // for each entry to be extracted
217             File fentry = new File(f, fName);
218             File dir = fentry.isDirectory() ? fentry : fentry.getParentFile();
219             dir.mkdirs();
220             if (fentry.isDirectory()) {
221                 zipentry = zipinputstream.getNextEntry();
222                 continue;
223             }
224
225             FileHelper.copy(zipinputstream, fentry);
226             zipinputstream.closeEntry();
227             zipentry = zipinputstream.getNextEntry();
228
229         }
230         zipinputstream.close();
```

Figure 1: Vulnerable code (note the missing name check on line 217 and onward).

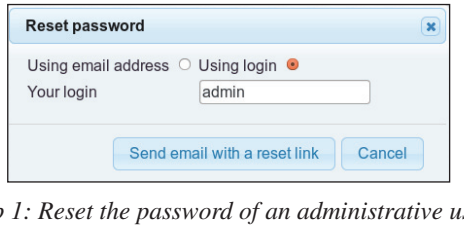
```
240     private void sendHashByUser(User us, String appLink, UserDao userDao) throws Exception {
241         String loginData = us.getLogin() + new Date();
242         log.debug("User: " + us.getLogin());
243         us.setResethash(ManageCryptStyle.getInstanceOfCrypt().createPassPhrase(loginData));
244         userDao.update(us, -1L);
245         String reset_link = appLink + "?hash=" + us.getResethash();
246
247         String email = us.getAdresses().getEmail();
248
249         String template = ResetPasswordTemplate.getEmail(reset_link);
250
251         getBean(MailHandler.class).send(email, Application.getString(517), template);
252     }
253
254 }
```

Figure 2: Extract from the password reset function.

PREDICTABLE PASSWORD RESET TOKEN (CVE-2016-0783)

When auditing code, it is always good to review security-sensitive areas such as authentication and related functions. One such function is the ability to reset a forgotten password, and in the case of *OpenMeetings*, this turned out to be a jackpot. The code shown in Figure 2 is an extract from the password reset function.

You may notice that the password reset token that is emailed to the user who has forgotten their password is made up of highly predictable components. On line 241, the username and the current system date and time are concatenated and assigned to the loginData variable, and on line 243, an MD5 hash of that variable is generated and written to the database. The hash is then read from the database and appended to the password reset link on line 245, which is then emailed to the user. As the output format of Java's Date() class does not even include milliseconds (the format is: Tue Aug 16 15:30:00 UTC 1977), an attacker with knowledge of an existing user's username can calculate the password reset token in less than a second, as shown in the PoC below:

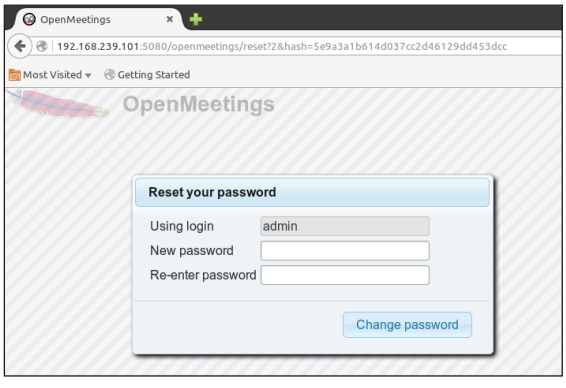


Step 1: Reset the password of an administrative user.

```
andreas@deathstar: ~/research/OpenMeetings/exploits
File  Edit  View  Search  Terminal  Help
andreas@deathstar:~/research/OpenMeetings/exploits$ ./reset_token.py admin 192.168.239.101:5080

[+] Apache Open Meetings password reset token exploit
[+] Starting attack at Tue Mar 08 23:05:40 CET 2016
[+] Attempting to extract token...
[+] Success! The password for user admin can be reset at:
[+] http://192.168.239.101:5080/openmeetings/reset?hash=5e9a3a1b614d037cc2d46129dd453dcc
[+] Attack finished at Tue Mar 08 23:05:40 CET 2016
```

Step 2: Calculate the password reset token.



Step 3: Follow the link to change the password.

GETTING CODE EXECUTION (IT'S A KIND OF MAGICK)

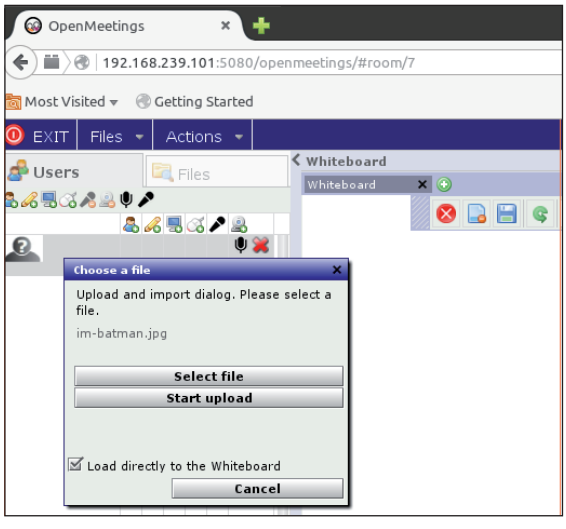
With the ability to gain access to the system by compromising an administrator's account, and being able to upload files to arbitrary locations on the file systems, the only thing missing is a way to turn these issues into code execution. As mentioned previously, *OpenMeetings* does not support JSP files in its default configuration so a different approach is needed. Enter, third-party integrations.

OpenMeetings has several third-party integrations for functions such as image conversion, IP telephony, instant messaging, and more. These integrations are usually triggered by a certain system event and are invoked using Java's getRuntime.exec(). One such third-party solution is *ImageMagick*, which is used to create a thumbnail when a user uploads a file. The default path to *ImageMagick*'s 'convert' binary, used for creating the thumbnails, is /usr/bin/.

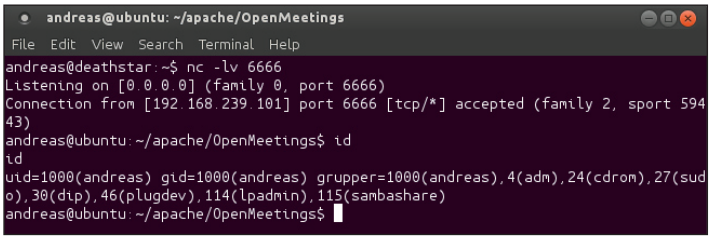
18	swftools_path	
19	imagemagick_path	/usr/bin/
20	sox_path	

Figure 3: OpenMeetings administrative menu.

As can be seen in Figure 3, the path to the 'convert' binary can be configured from the web interface. This means that an attacker with administrative access (which is assumed considering the previously described password reset vulnerability) can change the *ImageMagick* path to /tmp/ then upload a ZIP archive containing an executable file named '../../../../../../../../../../tmp/convert'. This file will be executed the next time an image is uploaded, effectively resulting in remote code execution.



Step 1: Upload a file.



Step 2: Get a shell.

(It should be mentioned that, if the *Tomcat* application server is running as the root user, which cannot be assumed, it would be possible for the attacker simply to overwrite the 'convert' file in /usr/bin/ without having to change the configuration settings. However, this would make the attack less reliable, and would also break system functionality, making the attack less stealthy.)

FINAL WORDS

Just as in [5], the chaining of two not-so-advanced vulnerabilities (plus some additional not-so-advanced trickery) resulted in remote code execution in an application that appears to be quite popular (250,000 downloads). The last reported vulnerability in *OpenMeetings* that I could find was an XSS from 2013 [6], which hints at how infrequently a lot of open-source projects are audited. As I have previously stated [1], more effort is needed in this area.

I would like to thank the *OpenMeetings* maintainers, especially Maxim, for being so cool about my report and fixing the issues in a very timely manner. The fixes are included in the new and improved *OpenMeetings* version 3.1.1, which is available for download from the official project website [2].

REFERENCES

[1] http://haxx.ml/post/137946990286/hacking-open-source-software-for-fun-and.

[2] http://openmeetings.apache.org/downloads.html.

[3] http://openmeetings.apache.org/.

[4] https://en.wikipedia.org/wiki/OpenMeetings.

[5] http://haxx.ml/post/140552592371/remote-code-execution-in-apache-jetspeed-230-and.

[6] https://issues.apache.org/jira/browse/OPENMEETINGS-793.