1 Introduction

CVE-2013-7315 [1] is a vulnerability in the Spring MVC that was submitted into the CVE security vulnerabilities dictionary on 23.01.2014. This vulnerability allows users with ill intentions to read arbitrary files, cause denial of service attacks, and conduct cross-site request forgery attacks by using crafted XML with JAXB [2]. CVE-2013-4152 [3] is another vulnerability which is tightly linked with CVE-2013-7315, main difference being in the affected version and the scope of affection. Therefore, this document covers both of these vulnerabilities.

Arbitrary files are sensitive files outside of the scope of the vulnerable application. These files can be any files on the system which makes it a big vulnerability. The vulnerability is abused by doing a XML Eternal Entity attack (XXE attack) [4]. For example, by doing an XXE attack, the attacker can retrieve a database of passwords. If the passwords are encrypted, a common method to decrypt them is to download a database of common passwords, encrypt them with the same method and compare the encrypted passwords to the passwords of the stolen database.

Denial of service (DoS) [5] attacks or distributed denial of service (DDoS) attacks are probably the most common form of virtual attacks today. They involve flooding the bandwith of the target by massively generating fake requests. Because the server cannot identify fake requests from real requests, it has to answer to all of them causing a massive delay or a crash of the server. A good way to describe it is by taking McDonald's drive-through as an example. Normally, there are, for example, 5 clients passing through and each of them orders something and then pays for its order causing McDonald's to profit from each of the client. In case of a DoS or DDoS attack, there are, for example, 1000 clients and 995 of them pass through, say "hello", and leave only to come back again and repeat its action. This causes the real clients to wait for a really long time or McDonald's to shut down its doors simply because it is wasting resources for no reason.

Cross-site request forgery (CSRF) [6] attacks are basically unauthorized injections from an abusing user to a website in order to steal other users information. To be more precise, the attacker changes something in a website that other users might use. For example, the attacker can inject an image into a website that causes the victim to send sensitive information to the attacker. This form of attack is not well known because it has many limitations, the biggest
one being a poor development of the website by an unexperienced programmer. In my opinion, this attack mainly depends on social engineering.

2 Vulnerability

The cause of a Spring application being vulnerable to XXE attacks was caused by the lack of a property in the Spring OXM wrapper to disable entity resolution when using the JAXB unmarshaller [7]. Entity resolution means identifying and linking or grouping different manifestations of the same real world object and the unmarshaller is used for deserializing XML data into Java content trees. There are two source implementations passed to the unmarshaller that created this vulnerability: SAXSource and StreamSource. For these instances, Spring processed external entities by default.

By abusing this vulnerability with XXE injection, the attackers were able to read local files causing a big security hole. In addition to this, XXE injection enabled the attackers to conduct DoS and CSRF attacks on other sites by using the vulnerable Spring application as a proxy. Attackers could also automate the attacks by creating various tools [8].

The affected applications were ones using the following versions of Spring MVC and OXM:

1. 3.0.0 to 3.2.3 (Spring OXM & Spring MVC)
2. 4.0.0.M1 (Spring OXM)
3. 4.0.0.M1 to 4.0.0.M2 (Spring MVC)

There is also a possibility that earlier versions are vulnerable.

3 Solution

The solution was fortunately straightforward - disable the external entity processing by default and provide an option to re-enable it. In my opinion, this vulnerability along with the fix is a nice example of how something small can be very crucial in the life of an web application. The following code describes the changes that fix the vulnerability.
private boolean processExternalEntities = false;

... 

private Source processSource(Source source) {

...

XMLReader xmlReader = null;
InputSource inputSource = null;

...

try {

    if (xmlReader == null) {

        xmlReader = XMLReaderFactory.createXMLReader();

    }

    xmlReader.setFeature("http://xml.org/sax/features/external-general-entities",
                        this.processExternalEntities);

    return new SAXSource(xmlReader, inputSource);

}

...

}

_appendix_1 - Code snippet from the patch fixing the vulnerability. [9]
References

[1] CVE-2013-7315,
   http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-7315

[2] JAXB,
   https://jaxb.java.net/

[3] CVE-2013-4152,
   http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-4152

[4] XXE attack,
   https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Processing

   http://www.us-cert.gov/ncas/tips/ST04-015

[6] CSRF,
   https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

[7] JAXB unmarshaller,
   https://jaxb.java.net/nonav/2.2.4/docs/api/javax/xml/bind/Unmarshaller.html

[8] DAVISET - DDoS attacks via other sites
   https://www.youtube.com/watch?v=RKi35-f346I

[9] GitHub spring-framework fix
   https://github.com/spring-projects/spring-framework/pull/317/files