OWASP XML eXternal Entity injection Lab Exercise

The development of this document is/was funded by three grants from the DOD Grant Number H98230-19-1-0301

1 Overview

This Labtainer exercise explore XML External Entity attack. An XML External Entity attack is a type of attack against an application that parses XML input. It is important to note that in some instances the XML parses is unstable and will not process XML input. This attack occurs when untrusted XML input either locally or containing reference to an external entity is processed by a weakly configured XML parser. These types of attacks may lead to the disclosure of confidential data, denial of service, and Server-Side Request Forgery (SSRF).

2 Lab Environment

This lab runs in the Labtainer framework, available at http://my.nps.edu/web/c3o/labtainers. That site includes links to a pre-built virtual machine that has Labtainers installed, however Labtainers can be run on any Linux host that supports Docker containers.

From your labtainer-student directory start the lab using:

```
labtainer web-xxe
```

On most Linux systems, these are links that you can right click on and select "Open Link". **If you chose to edit the lab report on a different system, you are responsible for copying the completed report back to the displayed path on your Linux system before using "stoplab" to stop the lab for the last time.**

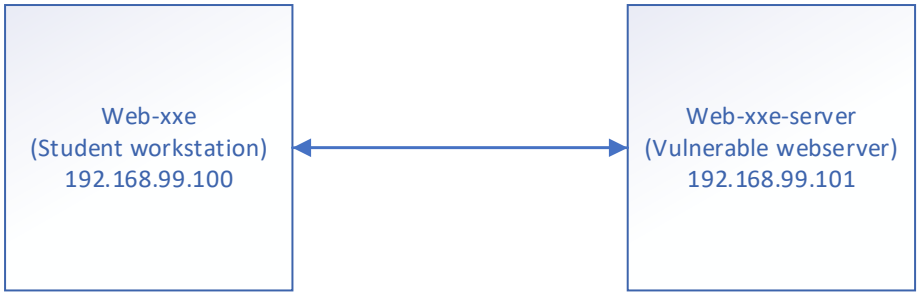
The resulting virtual terminal is connected to the student workstation; you will have OWASP ZAP and Firefox located on this workstation.

There are several accounts that are configured

User	Username	Password	Systems
Admin user	admin@juice-sh.op	admin123	web login
Nonadmin user	jim@juice-sh.op	ncc-1701	web login
User	Ubuntu	ubuntu	Student workstation

3 Network Configuration

The student workstation (web-xxe) is configured to have IP address `192.168.99.101` while the vulnerable webserver (web-xxe-server) is `192.168.99.100`.



4 Lab Tasks

It is assumed that the student has received instruction or independent study on the basic operation of web operations

4.1 Verify connectivity between student workstation and web server

A simple ping from the student workstation system will be sufficient.

```
ping 192.168.99.100
```

Note: to stop the ping use CTRL + C

4.2 Open Firefox and browse to the web server

At a terminal on the student workstation type:

```
firefox &
```

this will load Firefox, and type in the IP of the web server:

```
http://192.168.99.100
```

Record in Item #1 of your report why firefox might have been chosen to be the web browser used.

4.3 Open & Set up OWASP ZAP

At the terminal of the student workstation, type:

```
owasp-zap &
```

Note: if Firefox is running at the terminal and the "&" was not included then Firefox is not running in the background. Close Firefox and reopen using "Firefox &" at the terminal

OWASP ZAP Application should be open and it should be prompting the user for input.

- OWASP ZAP user input: select "yes, I want to persist this session with the name based on the current timestamp" then click start. This will open ZAP application.
- If you are prompted to "Manage Add-on" click close

4.4 Configure Firefox to use OWASP-ZAP as a Proxy

Within the preference section of Firefox configure the following steps:

- In the student workstations Firefox, open "Preferences"
- In the find window type "proxy"
- In Network Proxy Setting, select "Settings"
- Select "Manual proxy configuration"

- In the HTTP Proxy section: use “127.0.0.1” and Port “8080”
- Also select “Use this proxy server for all protocols”
- Click “ok” to accept the settings

The above setting ensures that Firefox will use OWASP Zap as the proxy. Perform the following steps to ensure the Firefox is connecting and using ZAP as a proxy

- Refresh the webpage “192.168.99.100”
- A security warning stating “Your connection is not secure” will be displayed.
- This warning message must be accepted. To do that click on “Advanced”
- It will display the SSL certificate and should show a “SEC_ERROR_UNKOWN ISSUE” it is ok to use this cert, click “Add Exception”
- A confirmation window will pop up, confirm the exception by clicking the “Confirm Security Exception”

Record in Item #2 of your report why set up a proxy.

4.5 Verification and Site Navigation

According to OWASP Web Security Testing Guidelines (WSTG), one of the first steps in checking a web site or application is passive testing, also known as site exploration. During passive testing, the tester tries to understand the site or application's logic and will explore the site or application as a user. This process is used to gather information on the target site or application. A common method of passive testing is setting up a HTTP proxy that is used to observe all the HTTP requests and responses. At the end of passive testing, the tester should understand all the access points (gates) of the site or application.

Once the proxy is set up, and you are looking at OWASP Zap, you will need to understand some basic navigation for OWASP ZAP. In the top center of OWASP Zap, this is known as the ‘workspace’ window. This is where requests and responses are displayed. You can also make changes here when a breakpoint has been hit. Above the request tab there is a green circle, if you mouse over it, it states this will be used to set breakpoint. A breakpoint is a pause in communication between a web browser and a web server allowing us to review content. We will be using breakpoints in this lab to verify our exploits. It is important to become familiar with these three main tabs, they are [1]:

- Request tab - This shows the data your browser sends to the application
- Response tab - This shows the data the application sends back to your browser
- Break tab - This allows you to manipulate the data

The tester should perform some basic navigation. Make sure that in OWASP Zap, in the bottom section as the tester navigate the different pages OWASP Zap is recording the pages visited. During site exploration, log in and visit pages that are protected behind a login screen.

Record in Item #3 of your report does OWASP ZAP check for vulnerabilities in each page you visit?

4.6 Exploiting XXE to retrieve files

According to WSTG-INPV-07, XML Injection testing is when a tester tries to inject an XML document into the web application. If the XML parser process the XML file, then the attack is successful. Because it is possible to crash the XML parser, this example will have you try to different methods in which they should both do the same task.

First from the web-xxe workstation, on the desktop there are two files titled “`evil1.xml`” and “`evil2.xml`” view the content of both files so you can see how they are made up and to see how they

differ. Pay attention to the section of code retrieve the passwd file as you will be writing code to extract a file later on. Once you have viewed the code of both files then continue on to the next step.

To perform an XXE from a craft XML file, the file must be uploaded to the server and read. Follow the steps below:

- From the student workstation, log into the web site at `http://192.168.99.100:3000`
- Once logged in, file a complaint. The complaint section is on the right-hand side of the screen once you have logged in.
- In OWASP Zap, turn on breaks, each time you click on a web task, you will need to allow the response to go through. Otherwise the break will prevent the next request from processing.
- Enter a message and in the invoice section click to browse for file upload
- Because the default option is to only accept pdf, please make sure that you select "view all files0", you should see the "`evil1.xml`" select it and click submit
- In OWASP Zap, you should see in the break that you get a message stating that B2B customer complaint…. But you will also get the passwd file contents allow for the response to be sent.
- In OWASP Zap, if successful then the contents of passwd will be displayed.
- In OWASP Zap, save the response as evil1.raw. To do this, right click in Zap in the "Break Section" select "save raw" and you will be saving the "Response body" It is important that you save it "evil1.raw"
- In OWASP Zap, after saving, allow for the response to be sent.
- Repeat the steps and text to see if "`evil2.xml`" displays the same content.

Record in Item #4 of your report did evil1.xml and evil2.xml produce the same result?

4.7 Exploiting a custom XML to exploit XXE

On the student workstation, view the detail of the file title "`evil3.xml`" modify the file to display the file located at "`/displayfile`" if you are not certain of the path, on the student work station, navigate to the location and ensure you can see it and read it. Repeat the steps in section 4.6 to upload it and ensure that the server process it.

Record in Item #5 of your report write the code you would need to display the "displayfile"

Record in Item #6 of your report if you could read data from any file located in the web server?

4.8 Writing a custom exploit using XML

The objective of this section is to allow the learner to write their own custom XML script to pull data out of a file that was identified as potential housing important information. The custom xml code must be saved on the student workstation as "`evil4.xml`" the learner will have to create the eveil4 file on the desktop of the student workstation. The learner knowns that the file is located at etc/ and is titled key.txt. the learner will review the steps above and write the entire XML exploit from examples above to obtain the information on the key file. Repeat the steps in section 4.6 to upload it and ensure that the server process it.

4.9 Exploring SSRF and Document Type Declaration (DTD) Concepts

According to the WSTG section on Testing for XML Injection (OTG-INPVAL-009), there are two additional types of XXE attacks that have not been discussed; they are SSRF and DTD. XXE attacks occurs when untrusted XML input containing a reference to an external entity is processed by a weakly configured XML parser. This type of attack may lead to the disclosure of confidential data (as explored above), denial of service, Server-Side Request Forgery (SSRF), and the ability to reference external services to call down data files (DTD based attacks). However, XML parsers are very finicky and often SSRF and DTD based attacks are not very successful. Review the OWASP XXE Processing guide [2] and answer the following questions.

Record in Item #7 of your report the details of how an SSRF attack is conducted.

Record in Item #8 of your report discuss how a DTD external based attack or reference has impact on a web server.

4.10 Generate Report

In OWASP ZAP once the tester has completed their review of XXE vulnerabilities, in the ZAP application under Report on the top and save a HTML report. Save the report to the home directory of the web-xxe workstation, call the file "`report_zap`"

Record in Item #9 of your report review the report from zap and list at three items of interest relating directly with XXE exploits.

4.11 Review Journal output

This task allows the user to see output that the server would be generating when certain attacks or exploits have run against them. The learner will review journal output on the server.

- On the shell of the web server type the following command:
  - Sudo journalctl -u juice-shop
- Review the output and space bar until the end, record anything of note and if there are any possible exploits that had ran. Support your claim with output from the journal. Record this at the end of your report under question 10.

5 Stop the Labtainer

When the lab is completed, or you'd like to stop working for a while, run

```
stoplab web-xxe
```

from the host Labtainer working directory. You can always restart the Labtainer to continue your work. When the Labtainer is stopped, a zip file is created and copied to a location displayed by the stoplab command. When the lab is completed, send that zip file to the instructor.

References

1. OWASP ZAP UI Overview. Available at the following URL:

   https://www.zaproxy.org/docs/desktop/ui/

2. OWASP XXE Processing Guide. Available at the following URL:

https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing