OWASP Insufficient Logging and Monitoring 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 explores Insufficient logging and monitoring. When a hacker infiltrates a network, IT systems will generate traffic which usually doesn't correspond to the normal one, unless you are dealing with highly skilled hackers who have time and money to go after your IT infrastructure. If you can't detect this abnormal behavior as soon as possible, you are essentially giving them enough time to achieve their goal.

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-inslog
```

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.**

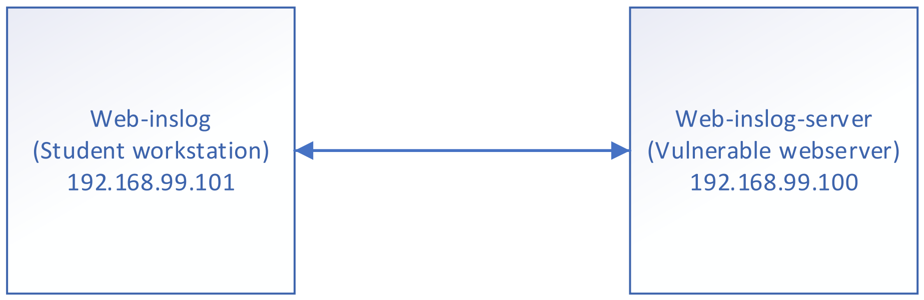
The resulting virtual terminal is connected to the student workstation; you will have OWASP ZAP and Firefox located on this workstation.

There are several accounts that are configured

Username	Password	Systems
admin@juice.org	admin123	Admin for web login
jim@juice.org	ncc-1701	web login
Ubuntu	ubuntu	Student workstation
Ubuntu	ubuntu	vulnerable server

3 Network Configuration

The student workstation (`web-inslog`) is configured to have IP address `192.168.99.101` while the vulnerable webserver (`web-inslog` -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

The objective of this task is to set up OWASP ZAP to be function and to allow the capture of traffic from the web-xss student workstation. 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 Accessing Restricted Areas

The objective of this section is to see what access is allowed and to determine even if you can access a restricted area if you will be allowed to do anything. According to OWASP WSTG-ATHZ-01, one of the first tasks when looking at a web site is to see what URLs and directories you have access to. Conducting a basic site survey using a web site crawler is the preferred method for data gathering on a web site.

Example 1 – Web Survey
The following will allow OWASP Zap to crawl a website to determine if what paths are accessible.

- Open OWASP Zap and perform a site scan on the IP address:
  `192.168.99.100:3000`
- Under alerts you should see a section titled Path Traversal, see if you can see any paths that you would think are password protect or secured.
- Do you see any links to administrative pages?
- Please save your scan results from OWASP Zap, save it to the desktop and title it “traversal.html”

Record in Item #3 of your report how would you prevent access to an administrative page?

4.6 Accessing Server Logs

The objective of this task is to see if the learner can access server logs or other metric data that the server generates. According to OWASP WSTG-CRYP-03, sensitive data must be protected when it is transmitted or even at rest. If data is transmitted over HTTPS or encrypted in another way the protection mechanism must not have limitations or vulnerabilities. If data is at rest, there needs to be a control in place to secure sensitive data. As a rule of thumb if data must be protected when it is stored, this data must also be protected during transmission. sensitive data is typically defined as passwords or authentication-based content, but it can also include server logs and other information used to compromise a system or service.

The first example below will have the learner look through some left-over information from an IT server team member to aid in identifying were log files may be. The second example will let the learner explore the site survey map and see if they can find metric location.

Example 1 – Gaining Access to Server Logs

- While completing the tasks above, and reviewing content in the ftp directory, a file titled "`incident-support.kdbx`" should be reviewed.
- Upon inspection of the file a directory should stand out, and that is the location the support team saves log files.
- You can also review task 4.5 site survey and determine if there is a support directory listed. View the continence of the following URL

```
https://192.168.99.100:3000/support/logs
```

- Can you access the server logs? Save the logs as "logs.txt"

Record in Item #4 of your report why would someone want to gain access to access logs of a server?

4.7 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.
- Review task 4.6, check the system logs, are there any indication that you are running an XML exploit against the server?

Record in Item #5 of your report are XXE exploits still common? How can they be prevented?

4.8 Persistent XSS without Front-End Access

The objective of this section is to perform a persistent XSS exploit without accessing the front end of the web server using curl. Follow the proceeding steps to identify product and try to exploit the server using a PUT request. The goal of this task is to see if the learner can modify or updating products.

- Search for a product, the user should see that the URL doesn't change between the different product. The user can view all product detail by navigating to the following URL:
    - `https://192.168.99.100:3000/api/Products/`
- This gives the learner a list of product IDs and will be used in modifying data in the proceeding steps. Take note of "Orange Juice" which is product ID2 and pay attention to the description.
- From the student workstation terminal, the learner will use curl to see if the options for a specific product. Type the following command at the terminal:

    -
      ```
      curl -X OPTIONS -D –
      'http://192.168.99.100:3000/api/Products/2'
      ```
- Pay attention to the Access-Control-Methods, if the learner is unsure of the different types then a review of HTTP requests will need to be reviewed. The fact that it accepts PUT, should mean that we can push updates via the terminal.
- From the student workstation terminal, type the following command

    -
      ```
      curl -X PUT
      "http://192.168.99.100:3000/api/Products/2" -H
      "Content-Type: application/json" --data-binary
      '{"description":"TEST"}'
      ```
- check the Products page to see if the description was updated. The product URL is:
    - https://192.168.99.100:3000/api/Products/
- Modify the curl PUT command to check the description to include an XXS error
- Review task 4.6, check the system logs, are there any indication that you are running an XML exploit against the server?

Record in Item #6 of your report Why is allowing a PUT request from external sources a bad idea?

4.9 User Login Exploit using Injection

According to OWASP WSTG-INPV-05, an SQL injection testing checks if it is possible to inject data into the application so that it executes a user-controlled SQL query in the database. Testers find a SQL injection vulnerability if the application uses user input to create SQL queries without proper input validation. A successful exploitation of this class of vulnerability allows an unauthorized user to access or manipulate data in the database.

An SQL injection attack consists of insertion or "injection" of either a partial or complete SQL query via the data input or transmitted from the client (browser) to the web application. A successful SQL injection attack can read sensitive data from the database, modify database data (insert/update/delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file existing on the DBMS file system or write files into the file system, and, in some cases, issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands.

OWASP outlines the steps in testing for SQL injections by a few injections based attacks. First is to perform a standard or classic SQL injection attack which will be completed in example 1. OWASP further explains that you can also capture data and inject modified version in transit, which will be review in example 2. In the last example the learner will combine what they have learned in the first to examples and will be performing injection attacks to bypass user authentication, this will be examined in example 3.

Example 1 – Classic SQL Injection
Simple SQL injection would be trying to use the command below in the user login section to see if anything comes back. SQL basic injections such as the ones listed below may not always work. If that is that case it is ok to document which ones you tried and if any of them worked. Command is single quote and two hyphens as seen below.

```
'--
```

- The above code may yield some information
- The next type of basic SQL injection will be using a combination sequence. The try logging in with the following command in the email with any password
  ```
  'a or 1=1
  ```
  - `This may not work as some exploits get patched.`
- The 'a allows for login with the administration account, using the user list below are any of the other account able to be logged in with just using their first letter and the 1=1 command?
- The above sections focused on users, let's see if the product or search page are vulnerable. The point will focus on the following URL:
  ```
  http://192.168.99.100:3000/#/search?q=
  ```
- The q= allows us to manipulate data. Try adding 1=1 after the equals sign, does this do anything? In older version of SQL this might have returned an error, but this site is new enough, so it does.
- Try modifying your search to use the following characters or combination of characters:
  ```
  ' ) ( - = "
  ```
- Below are two different URLs compare the two and see if there are any differences
  ```
  http://192.168.99.100:3000/#/search?q='))--
  http://192.168.99.100:3000/#/search
  ```
- Review task 4.6, check the system logs, are there any indication that you are running an SQL injection exploit against the server?

Record in Item #7 of your report why is "1=1" a common SQL injection exploit?

4.10 Accessing Restricted Areas

The objective of this section is to see what access is allowed and to determine even if you can access a restricted area if you will be allowed to do anything. According to OWASP WSTG-ATHZ-01, one of the first tasks when looking at a web site is to see what URLs and directories you have access to. Conducting a basic site survey using a web site crawler is the preferred method for data gathering on a web site.

Example 1 – Path Traversal
The following will allow OWASP Zap to crawl a website to determine if what paths are accessible.

- Open OWASP Zap and perform a site scan on the IP address:
  `192.168.99.100:3000`
- Do you see any links to administrative pages?
- Please save your scan results from OWASP Zap, save it to the desktop and title it "traversal.html"

Example 2 – feedback modification

- In Firefox navigate to the following URL
  `http://192.168.99.100:3000/#/administatration`
- Once you are at the administrative page are you able to remove feedback? Remove all 1-star feedback

Review task 4.6, are there any indication of users accessing restricted portions of the website or being able to modify user views?

Record in Item #8 of your report why is path transversal such an issue on web server?

4.11 Generate Report

In OWASP ZAP once the tester has found the different 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-inject workstation, call the file "`report_zap`"

Record in Item #9 of your report any interesting finding of the report. Find three areas of interest and explain why they were important and how they may have an impact on the security of this website.

4.12 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-inslog
```

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