Offensive Security Web Expert Exam Report

OSWE Exam Report

student@youremailaddress.com, OSID: XXXX

2020-07-25

Contents

1 OSWE Exam Report

1.1 Introduction

The Offensive Security OSWE exam documentation contains all efforts that were conducted in order to pass the Offensive Security Web Expert exam. This report will be graded from a standpoint of correctness and fullness to all aspects of the exam. The purpose of this report is to ensure that the student has the technical knowledge required to pass the qualifications for the Offensive Security Web Expert certification.

1.2 Objective

The objective of this assessment is to perform a white-box penetration test the Offensive Security Exam network. The student is tasked with following methodical approach in obtaining access to the objective goals. This test should simulate an actual white-box penetration test with Proof of Concept and how you would start from beginning to end, including the overall report.

1.3 Requirements

The student will be required to fill out this exam documentation fully and to include the following sections:

- Methodology walkthrough and detailed outline of steps taken
- Each finding with included screenshots, walkthrough, sample code, and proof.txt if applicable.
- Any additional items that were not included

2 High-Level Summary

I was tasked with performing a white-box penetration test towards Offensive Security Exam. A white-box penetration test is sifting through the massive amount of data available to identify potential points of weakness. The focus of this test is to provide a comprehensive assessment of both internal and external vulnerabilities. My overall objective was to evaluate the application, identify vulnerabilities, and write automated exploit while reporting the findings back to Offensive Security.

When performing the white-box penetration test, there were several critical vulnerabilities that were identified on Offensive Security's network. When performing the attacks, I was able to gain access to multiple machines, primarily due to design flaws and implementation errors. During the testing, I had a shell access to multiple systems. All systems were successfully exploited and access granted. These systems as well as a brief description on how access was obtained are listed below:

- 192.168.x.x - app_name - Short summary of the exploit path
- 192.168.x.x - app_name - Short summary of the exploit path

2.1 Recommendations

I recommend patching the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

3 Whitebox audit

The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems. During this penetration test, I was able to successfully gain access to **X** out of the **2** systems.

3.1 192.168.XX.XX - app_name [language]

3.1.1 Local.txt & Proof.txt

Provide screenshots of Burp and your browser showing a successful login as the administrative user on the actual target machine with the value of local.txt visible.

local.txt: `xxx`

Figure 3.1: local.txt

Provide a screenshot `id` and `ip a` command and the contents of proof.txt.

proof.txt: `xxx`

Figure 3.2: proof.txt

3.1.2 Debug setup

Provide your debug setup.

3.1.3 Vulnerability 1 - vulnerability_name

Provide the method and code used to find the vulnerability 1.

3.1.4 Vulnerability 2 - vulnerability_name

Provide the method and code used to find the vulnerability 2.

3.1.5 Vulnerability X - vulnerability_name

Provide the method and code used to find the vulnerability X.

3.1.6 Steps of exploit writing

Provide a detailed account of your methodology in creating the exploits. The steps taken should be able to be easily followed and reproducible if necessary.

3.1.7 PoC Code

To install the dependencies required for PoC execution:

```
$ package_manager install dependency1 dependency2
```

Provide the final proof of concept code used to gain access to the server.

```
#!/usr/bin/env ruby

puts 'My best PoC'
```

3.2 192.168.XX.XX - app_name [language]

3.2.1 Local.txt & Proof.txt

Provide screenshots of Burp and your browser showing a successful login as the administrative user on the actual target machine with the value of local.txt visible.

local.txt: xxx

Figure 3.3: local.txt

Provide a screenshot `id` and `ip a` command and the contents of proof.txt.

proof.txt: xxx

Figure 3.4: proof.txt

3.2.2 Debug setup

Provide your debug setup.

3.2.3 Vulnerability 1 - vulnerability_name

Provide the method and code used to find the vulnerability 1.

3.2.4 Vulnerability 2 - vulnerability_name

Provide the method and code used to find the vulnerability 2.

3.2.5 Vulnerability X - vulnerability_name

Provide the method and code used to find the vulnerability X.

3.2.6 Steps of exploit writing

Provide a detailed account of your methodology in creating the exploits. The steps taken should be able to be easily followed and reproducible if necessary.

3.2.7 PoC Code

Provide the final proof of concept code used to gain access to the server.

To install the dependencies required for PoC execution:

```
$ package_manager install dependency1 dependency2
```

```
#!/usr/bin/env ruby

puts 'My best PoC'
```

4 Additional Items

This section is placed for any additional items that were not mentioned in the overall report.

4.1 Appendix - Proof.txt, Local.txt, and Machines summary

Key	Machine 1
IP (Hostname)	192.168.x.x
Name	app_name
Language	x
Local.txt Contents	xxx
Proof.txt Contents	xxx

Key	Machine 2
IP (Hostname)	192.168.x.x
Name	app_name
Language	x
Local.txt Contents	xxx
Proof.txt Contents	xxx