Offensive Security Web Expert

Exam Report

OSID: XXXX - student@youremailaddress.com

2020-07-25

Contents

1 OSWE Exam Report

1.1 Introduction

The Offensive Security OSWE exam report 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 against 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

In the 48 hours between [2020-01-01 10:00] to [2020-01-03 10:00] the student was tasked with performing a white-box penetration test towards Offensive Security Exam containing [#] applications.

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. The student's 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, the student was able to gain access to multiple machines, primarily due to design flaws and implementation errors. On [#] out of [#] servers, full shell access was achieved. These systems as well as a brief description on how access was obtained are listed below:

- **Application (192.168.x.x)** - Short summary of the exploit path
- **Application (192.168.x.x)** - Short summary of the exploit path
- **Application (192.168.x.x)** - Short summary of the exploit path

Full details can be found in the Whitebox audit section and scripts to automatically exploit the identified vulnerabilities can be found amongst the Appendixes.

2.1 Recommendations

We 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, the student was able to successfully gain full access to X out of the Y systems.

3.1 Application (192.168.x.x)

The application is a custom web application written in [language]. The application [provides the following functionality and users].

During the testing, the student was provided with unauthenticated access to the application. [The server was configured with debug functionality]. A number of vulnerabilities were found in the application, ranging from [XSS] to [RCE], allowing the student to achieve full control of the application and underlaying server.

Each found vulnerability is described in detail below, a script to automatically exploit the server can be found in appendix *Appendix - Full script for [application] exploitation*.

3.1.1 Proof of exploitation

The following sensitive files were extracted from the server, as proof of successful exploitation;

`local.txt` - `MakeSureToEndLineWithTwoSpaces`
`proof.txt` - `OrElseItWillEndUpOnOneLine`

Figure 3.1: local.txt

Figure 3.2: proof.txt

3.1.2 Vuln

```
Some code.

Check the FAQ if you have issues with long lines.
```

3.1.3 Vuln

```
<?php echo 'Hello World'; ?>
```

3.1.4 Vuln

```
#!/usr/bin/python

print('Hello World')
```

3.1.5 Vuln

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

3.1.6 Recommendations

-
-
-

3.2 Application (192.168.x.x)

The application is a custom web application written in [language]. The application [provides the following functionality and users].

During the testing, the student was provided with unauthenticated access to the application. [The server was configured with debug functionality]. A number of vulnerabilities were found in the application, ranging from [XSS] to [RCE], allowing the student to achieve full control of the application and underlaying server.

Each found vulnerability is described in detail below, a script to automatically exploit the server can be found in appendix *Appendix - Full script for [application] exploitation*.

3.2.1 Proof of exploitation

The following sensitive files were extracted from the server, as proof of successful exploitation;

`local.txt` - `MakeSureToEndLineWithTwoSpaces`
`proof.txt` - `OrElseItWillEndUpOnOneLine`

Figure 3.3: local.txt

Figure 3.4: proof.txt

3.2.2 Vuln

```
Some code.

Check the FAQ if you have issues with long lines.
```

3.2.3 Vuln

```
<?php echo 'Hello World'; ?>
```

3.2.4 Vuln

```
#!/usr/bin/python

print('Hello World')
```

3.2.5 Vuln

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

3.2.6 Recommendations

-
-
-

3.3 Application (192.168.x.x)

The application is a custom web application written in [language]. The application [provides the following functionality and users].

During the testing, the student was provided with unauthenticated access to the application. [The server was configured with debug functionality]. A number of vulnerabilities were found in the application, ranging from [XSS] to [RCE], allowing the student to achieve full control of the application and underlaying server.

Each found vulnerability is described in detail below, a script to automatically exploit the server can be found in appendix *Appendix - Full script for [application] exploitation*.

3.3.1 Proof of exploitation

The following sensitive files were extracted from the server, as proof of successful exploitation;

`local.txt` - `MakeSureToEndLineWithTwoSpaces`
`proof.txt` - `OrElseItWillEndUpOnOneLine`

Figure 3.5: local.txt

Figure 3.6: proof.txt

3.3.2 Vuln

```
Some code.

Check the FAQ if you have issues with long lines.
```

3.3.3 Vuln

```
<?php echo 'Hello World'; ?>
```

3.3.4 Vuln

```
#!/usr/bin/python

print('Hello World')
```

3.3.5 Vuln

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

3.3.6 Recommendations

-
-
-

4 Appendixes

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

4.1 Appendix - Exam 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

4.2 Appendix - Full script for [application] exploitation

4.2.1 Execution steps

Document requirements and all steps to get it running …

4.2.2 Script

4.3 Appendix - Full script for [application] exploitation

4.3.1 Execution steps

Document requirements and all steps to get it running …

4.3.2 Script

4.4 Appendix - Full script for [application] exploitation

4.4.1 Execution steps

Document requirements and all steps to get it running …

4.4.2 Script