Offensive Security Web Expert

Exam Report

OSID: XXXX - student@youremailaddress.com

2021-10-05

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

```
The longest recommended line in code blocks is 126 character for the first line and then 122 characters for the following
↪  lines. This is because of limitations in pandoc that doesn't break very long lines, such as Base64 blobs that easily get
↪  very long. The easiest thing to do is to include a space at these points.

Example;
VGhlIGxvbmdlc3QgcmVjb21tZW5kZWQgbGluZSBpbiBjb2RlIGJsb2NrcyBpcyAxMjYgY2hhcmFjdGVyIGZvciB0aGUgZmlyc3QgbGluZSBhbmQgdGhlbiAxMjIgY2hhcmFjdGVycyBmb3IgdGl

VGhlIGxvbmdlc3QgcmVjb21tZW5kZWQgbGluZSBpbiBjb2RlIGJsb2NrcyBpcyAxMjYgY2hhcmFjdGVyIGZvciB0aGUgZmlyc3QgbGluZSBhbmQgdGhlbiAxMjIgY2
↪  hhcmFjdGVycyBmb3IgdGhlIGZvbGxvd2luZyBsaW5lcy4gVGhpcyBpcyBiZWNhdXNlIG9mIGxpbWl0YXRpb25zIGluIHBhbmRvYyB0aGF0IGRvZXNuJ3QgYnJl
↪  YWsgdmVyeSBsb25nIGxpbmVzLCBzdWNoIGFzIEJhc2U2NCBibG9icyB0aGF0IGVhc2lseSBnZXQgdmVyeSBsb25nLiBUaGUgZWFzaWVzdCB0aGluZyB0byBkby
↪  BpcyB0byBpbmNsdWRlIGEgc3BhY2UgYXQgdGhpcyBwb2ludHMuCgpIYWhhIEkgbWFkZSB5b3UgZGVjb2RlIHRoaXMgdG8gY2hlY2sgd2hhdCBpdCBzYXlzIGh1
↪  aD8gTm9zeSBiYXN0YXJkIDsp
```

3.1.3 Vuln

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

3.1.4 Vuln

```python
#!/usr/bin/python

print('Hello World')
```

3.1.5 Vuln

```java
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

```
The longest recommended line in code blocks is 126 character for the first line and then 122 characters for the following
↪  lines. This is because of limitations in pandoc that doesn't break very long lines, such as Base64 blobs that easily get
↪  very long. The easiest thing to do is to include a space at these points.

Example;
VGhlIGxvbmdlc3QgcmVjb21tZW5kZWQgbGluZSBpbiBjb2RlIGJsb2NrcyBpcyAxMjYgY2hhcmFjdGVyIGZvciB0aGUgZmlyc3QgbGluZSBhbmQgdGhlbiAxMjIgY2hhcmFjdGVycyBmb3IgdGhlIGZvbGxvd2luZyBsaW5lcy4gVGhpcyBpcyBiZWNhdXNlIG9mIGxpbWl0YXRpb25zIGluIHBhbmRvYyB0aGF0IGRvZXNuJ3QgYnJlYWsgdmVyeSBsb25nIGxpbmVzLCBzdWNoIGFzIEJhc2U2NCBibG9icyB0aGF0IGVhc2lseSBnZXQgdmVyeSBsb25nLiBUaGUgZWFzaWVzdCB0aGluZyB0byBkbyBpcyB0byBpbmNsdWRlIGEgc3BhY2UgQgdGhpcyBwb2ludHMuCgpIYWhhIEkgbWFkZSB5b3UgZGVjb2RlIHRoaXMgdG8gY2hlY2sgd2hhdCBpdCBzYXlzIGh1aD8gTm9zeSBiYXN0YXJkIDsp

VGhlIGxvbmdlc3QgcmVjb21tZW5kZWQgbGluZSBpbiBjb2RlIGJsb2NrcyBpcyAxMjYgY2hhcmFjdGVyIGZvciB0aGUgZmlyc3QgbGluZSBhbmQgdGhlbiAxMjIgY2
↪  hhcmFjdGVycyBmb3IgdGhlIGZvbGxvd2luZyBsaW5lcy4gVGhpcyBpcyBiZWNhdXNlIG9mIGxpbWl0YXRpb25zIGluIHBhbmRvYyB0aGF0IGRvZXNuJ3QgYnJl
↪  YWsgdmVyeSBsb25nIGxpbmVzLCBzdWNoIGFzIEJhc2U2NCBibG9icyB0aGF0IGVhc2lseSBnZXQgdmVyeSBsb25nLiBUaGUgZWFzaWVzdCB0aGluZyB0byBkby
↪  BpcyB0byBpbmNsdWRlIGEgc3BhY2UgQgdGhpcyBwb2ludHMuCgpIYWhhIEkgbWFkZSB5b3UgZGVjb2RlIHRoaXMgdG8gY2hlY2sgd2hhdCBpdCBzYXlzIGh1
↪  aD8gTm9zeSBiYXN0YXJkIDsp
```

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

```
The longest recommended line in code blocks is 126 character for the first line and then 122 characters for the following
↪  lines. This is because of limitations in pandoc that doesn't break very long lines, such as Base64 blobs that easily get
↪  very long. The easiest thing to do is to include a space at these points.

Example;
VGhlIGxvbmdlc3QgcmVjb21tZW5kZWQgbGluZSBpbiBjb2RlIGJsb2NrcyBpcyAxMjYgY2hhcmFjdGVyIGZvciB0aGUgZmlyc3QgbGluZSBhbmQgdGhlbiAxMjIgY2hhcmFjdGVycyBmb3IgdGhlIGZvbGxvd2luZyBsaW5lcy4gVGhpcyBpcyBiZWNhdXNlIG9mIGxpbWl0YXRpb25zIGluIHBhbmRvYyB0aGF0IGRvZXNuJ3QgYnJlYWsgdmVyeSBsb25nIGxpbmVzLCBzdWNoIGFzIEJhc2U2NCBibG9icyB0aGF0IGVhc2lseSBnZXQgdmVyeSBsb25nLiBUaGUgZWFzaWVzdCB0aGluZyB0byBkbyBpcyB0byBpbmNsdWRlIGEgc3BhY2UgQgdGhpcyBwb2ludHMuCgpIYWhhIEkgbWFkZSB5b3UgZGVjb2RlIHRoaXMgdG8gY2hlY2sgd2hhdCBpdCBzYXlzIGh1aD8gTm9zeSBiYXN0YXJkIDsp

VGhlIGxvbmdlc3QgcmVjb21tZW5kZWQgbGluZSBpbiBjb2RlIGJsb2NrcyBpcyAxMjYgY2hhcmFjdGVyIGZvciB0aGUgZmlyc3QgbGluZSBhbmQgdGhlbiAxMjIgY2
↪  hhcmFjdGVycyBmb3IgdGhlIGZvbGxvd2luZyBsaW5lcy4gVGhpcyBpcyBiZWNhdXNlIG9mIGxpbWl0YXRpb25zIGluIHBhbmRvYyB0aGF0IGRvZXNuJ3QgYnJl
↪  YWsgdmVyeSBsb25nIGxpbmVzLCBzdWNoIGFzIEJhc2U2NCBibG9icyB0aGF0IGVhc2lseSBnZXQgdmVyeSBsb25nLiBUaGUgZWFzaWVzdCB0aGluZyB0byBkbyBpcy
↪  BpcyB0byBpbmNsdWRlIGEgc3BhY2UgQgdGhpcyBwb2ludHMuCgpIYWhhIEkgbWFkZSB5b3UgZGVjb2RlIHRoaXMgdG8gY2hlY2sgd2hhdCBpdCBzYXlzIGh1
↪  aD8gTm9zeSBiYXN0YXJkIDsp
```

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

4.5 Appendix - Highlight testing

This section if purely a reference of all the available highlight languages and should be removed before final report generation. It's generated with the following one-liner;

```
$ (for f in $(pandoc --list-highlight-languages); do echo "\`\`\`$f"; echo '$ echo "some output from '$f'"'; echo "some output
↪  from $f"; echo "# whoami"; echo "root"; echo "\`\`\`"; echo ""; done;) > highlight-languages.md
```

If you want to test all the styles, you can do that with;

```
$ for s in $(pandoc --list-highlight-styles); do pandoc --template eisvogel --highlight-style $s -o highlight-$s.pdf
↪  highlight-languages.md; done;
```

```
$ echo "some output from abc"
some output from abc
# whoami
root
```

```
$ echo "some output from asn1"
some output from asn1
# whoami
root
```

```
$ echo "some output from asp"
some output from asp
# whoami
root
```

```
$ echo "some output from ats"
some output from ats
# whoami
root
```

```
$ echo "some output from awk"
some output from awk
# whoami
root
```

```
$ echo "some output from actionscript"
some output from actionscript
# whoami
root
```

```
$ echo "some output from ada"
some output from ada
# whoami
root
```

```
$ echo "some output from agda"
some output from agda
# whoami
root
```

```
$ echo "some output from alertindent"
some output from alertindent
# whoami
root
```

```
$ echo "some output from apache"
some output from apache
# whoami
root
```

```
$ echo "some output from bash"
some output from bash
# whoami
root
```

```
$ echo "some output from bibtex"
some output from bibtex
# whoami
root
```

```
$ echo "some output from boo"
some output from boo
# whoami
root
```

```
$ echo "some output from c"
some output from c
# whoami
root
```

```
$ echo "some output from cs"
some output from cs
# whoami
root
```

```
$ echo "some output from cpp"
some output from cpp
# whoami
root
```

```
$ echo "some output from cmake"
some output from cmake
# whoami
root
```

```
$ echo "some output from css"
some output from css
# whoami
root
```

```
$ echo "some output from changelog"
some output from changelog
# whoami
root
```

```
$ echo "some output from clojure"
some output from clojure
# whoami
root
```

```
$ echo "some output from coffee"
some output from coffee
# whoami
root
```

```
$ echo "some output from coldfusion"
some output from coldfusion
# whoami
root
```

```
$ echo "some output from commonlisp"
some output from commonlisp
# whoami
root
```

```
$ echo "some output from curry"
some output from curry
# whoami
root
```

```
$ echo "some output from d"
some output from d
# whoami
root
```

```
$ echo "some output from dtd"
some output from dtd
# whoami
root
```

```
$ echo "some output from default"
some output from default
# whoami
root
```

```
$ echo "some output from diff"
some output from diff
# whoami
root
```

```
$ echo "some output from djangotemplate"
some output from djangotemplate
# whoami
root
```

```
$ echo "some output from dockerfile"
some output from dockerfile
# whoami
root
```

```
$ echo "some output from doxygen"
some output from doxygen
# whoami
root
```

```
$ echo "some output from doxygenlua"
some output from doxygenlua
# whoami
root
```

```
$ echo "some output from eiffel"
some output from eiffel
# whoami
root
```

```
$ echo "some output from elixir"
some output from elixir
# whoami
root
```

```
$ echo "some output from elm"
some output from elm
# whoami
root
```

```
$ echo "some output from email"
some output from email
# whoami
root
```

```
$ echo "some output from erlang"
some output from erlang
# whoami
root
```

```
$ echo "some output from fsharp"
some output from fsharp
# whoami
root
```

```
$ echo "some output from fortran"
some output from fortran
# whoami
root
```

```
$ echo "some output from gcc"
some output from gcc
# whoami
root
```

```
$ echo "some output from glsl"
some output from glsl
# whoami
root
```

```
$ echo "some output from gnuassembler"
some output from gnuassembler
# whoami
root
```

```
$ echo "some output from m4"
some output from m4
# whoami
root
```

```
$ echo "some output from go"
some output from go
# whoami
root
```

```
$ echo "some output from html"
some output from html
# whoami
root
```

```
$ echo "some output from hamlet"
some output from hamlet
# whoami
root
```

```
$ echo "some output from haskell"
some output from haskell
# whoami
root
```

```
$ echo "some output from haxe"
some output from haxe
# whoami
root
```

```
$ echo "some output from ini"
some output from ini
# whoami
root
```

```
$ echo "some output from isocpp"
some output from isocpp
# whoami
root
```

```
$ echo "some output from idris"
some output from idris
# whoami
root
```

```
$ echo "some output from fasm"
some output from fasm
# whoami
root
```

```
$ echo "some output from nasm"
some output from nasm
# whoami
root
```

```
$ echo "some output from j"
some output from j
# whoami
root
```

```
$ echo "some output from json"
some output from json
# whoami
root
```

```
$ echo "some output from jsp"
some output from jsp
# whoami
root
```

```
$ echo "some output from java"
some output from java
# whoami
root
```

```
$ echo "some output from javascript"
some output from javascript
# whoami
root
```

```
$ echo "some output from javascriptreact"
some output from javascriptreact
# whoami
root
```

```
$ echo "some output from javadoc"
some output from javadoc
# whoami
root
```

```
$ echo "some output from julia"
some output from julia
# whoami
root
```

```
$ echo "some output from kotlin"
some output from kotlin
# whoami
root
```

```
$ echo "some output from llvm"
some output from llvm
# whoami
root
```

```
$ echo "some output from latex"
some output from latex
# whoami
root
```

```
$ echo "some output from lex"
some output from lex
# whoami
root
```

```
$ echo "some output from lilypond"
some output from lilypond
# whoami
root
```

```
$ echo "some output from literatecurry"
some output from literatecurry
# whoami
root
```

```
$ echo "some output from literatehaskell"
some output from literatehaskell
# whoami
root
```

```
$ echo "some output from lua"
some output from lua
# whoami
root
```

```
$ echo "some output from mips"
some output from mips
# whoami
root
```

```
$ echo "some output from makefile"
some output from makefile
# whoami
root
```

```
$ echo "some output from markdown"
some output from markdown
# whoami
root
```

```
$ echo "some output from mathematica"
some output from mathematica
# whoami
root
```

```
$ echo "some output from matlab"
some output from matlab
# whoami
root
```

```
$ echo "some output from maxima"
some output from maxima
# whoami
root
```

```
$ echo "some output from mediawiki"
some output from mediawiki
# whoami
root
```

```
$ echo "some output from metafont"
some output from metafont
# whoami
root
```

```
$ echo "some output from modelines"
some output from modelines
# whoami
root
```

```
$ echo "some output from modula2"
some output from modula2
# whoami
root
```

```
$ echo "some output from modula3"
some output from modula3
# whoami
root
```

```
$ echo "some output from monobasic"
some output from monobasic
# whoami
root
```

```
$ echo "some output from mustache"
some output from mustache
# whoami
root
```

```
$ echo "some output from ocaml"
some output from ocaml
# whoami
root
```

```
$ echo "some output from objectivec"
some output from objectivec
# whoami
root
```

```
$ echo "some output from objectivecpp"
some output from objectivecpp
# whoami
root
```

```
$ echo "some output from octave"
some output from octave
# whoami
root
```

```
$ echo "some output from opencl"
some output from opencl
# whoami
root
```

```
$ echo "some output from php"
some output from php
# whoami
root
```

```
$ echo "some output from povray"
some output from povray
# whoami
root
```

```
$ echo "some output from pascal"
some output from pascal
# whoami
root
```

```
$ echo "some output from perl"
some output from perl
# whoami
root
```

```
$ echo "some output from pike"
some output from pike
# whoami
root
```

```
$ echo "some output from postscript"
some output from postscript
# whoami
root
```

```
$ echo "some output from powershell"
some output from powershell
# whoami
root
```

```
$ echo "some output from prolog"
some output from prolog
# whoami
root
```

```
$ echo "some output from protobuf"
some output from protobuf
# whoami
root
```

```
$ echo "some output from pure"
some output from pure
# whoami
root
```

```
$ echo "some output from purebasic"
some output from purebasic
# whoami
root
```

```
$ echo "some output from python"
some output from python
# whoami
root
```

```
$ echo "some output from qml"
some output from qml
# whoami
root
```

```
$ echo "some output from r"
some output from r
# whoami
root
```

```
$ echo "some output from relaxng"
some output from relaxng
# whoami
root
```

```
$ echo "some output from relaxngcompact"
some output from relaxngcompact
# whoami
root
```

```
$ echo "some output from roff"
some output from roff
# whoami
root
```

```
$ echo "some output from ruby"
some output from ruby
# whoami
root
```

```
$ echo "some output from rhtml"
some output from rhtml
# whoami
root
```

```
$ echo "some output from rust"
some output from rust
# whoami
root
```

```
$ echo "some output from sgml"
some output from sgml
# whoami
root
```

```
$ echo "some output from sml"
some output from sml
# whoami
root
```

```
$ echo "some output from sql"
some output from sql
# whoami
root
```

```
$ echo "some output from sqlmysql"
some output from sqlmysql
# whoami
root
```

```
$ echo "some output from sqlpostgresql"
some output from sqlpostgresql
# whoami
root
```

```
$ echo "some output from scala"
some output from scala
# whoami
root
```

```
$ echo "some output from scheme"
some output from scheme
# whoami
root
```

```
$ echo "some output from stata"
some output from stata
# whoami
root
```

```
$ echo "some output from tcl"
some output from tcl
# whoami
root
```

```
$ echo "some output from tcsh"
some output from tcsh
# whoami
root
```

```
$ echo "some output from texinfo"
some output from texinfo
# whoami
root
```

```
$ echo "some output from mandoc"
some output from mandoc
# whoami
root
```

```
$ echo "some output from typescript"
some output from typescript
# whoami
root
```

```
$ echo "some output from vhdl"
some output from vhdl
# whoami
root
```

```
$ echo "some output from verilog"
some output from verilog
# whoami
root
```

```
$ echo "some output from xml"
some output from xml
# whoami
root
```

```
$ echo "some output from xul"
some output from xul
# whoami
root
```

```
$ echo "some output from yaml"
some output from yaml
# whoami
root
```

```
$ echo "some output from yacc"
some output from yacc
# whoami
root
```

```
$ echo "some output from zsh"
some output from zsh
# whoami
root
```

```
$ echo "some output from dot"
some output from dot
# whoami
root
```

```
$ echo "some output from noweb"
some output from noweb
# whoami
root
```

```
$ echo "some output from rest"
some output from rest
# whoami
root
```

```
$ echo "some output from sci"
some output from sci
# whoami
root
```

```
$ echo "some output from sed"
some output from sed
# whoami
root
```

```
$ echo "some output from xorg"
some output from xorg
# whoami
root
```

```
$ echo "some output from xslt"
some output from xslt
# whoami
root
```