

**REALIZE & REACT SECURITY**

**SECURITY ASSESSMENTS FINDINGS REPORT FOR ForEver25**

**Date: April 17th, 2022**

**Project: Python**

**BUSINESS CONFIDENTIAL**

Table of Contents

[I Confidentiality Statement 3](#_Toc101123566)

[II Disclaimer 3](#_Toc101123567)

[III Contact Information 4](#_Toc101123568)

[1 Assessment Overview 5](#_Toc101123569)

[1.1 Assessment Components 6](#_Toc101123570)

[1.1.1 Internal Penetration Test 6](#_Toc101123571)

[1.2 Findings Severity Ratings 7](#_Toc101123572)

[1.3 Scope 8](#_Toc101123573)

[1.3.1 Scope Exclusions 8](#_Toc101123574)

[Client Allowances 8](#_Toc101123575)

[1.4 Executive Summary 9](#_Toc101123576)

[1.5 Summary of Tasks 9](#_Toc101123577)

[1.5.1 Introduction to Python 9](#_Toc101123578)

[1.5.2 Hello World 9](#_Toc101123579)

[1.5.3 Mathematical Operators 10](#_Toc101123580)

[1.5.4 Variables and Data Types 11](#_Toc101123581)

[1.5.5 Logical and Boolean Operators 12](#_Toc101123582)

[1.5.6 Shipping Project 12](#_Toc101123583)

[1.5.7 Loops 13](#_Toc101123584)

[1.5.8 Bitcoin project 13](#_Toc101123585)

[1.5.9 Files 14](#_Toc101123586)

[1.5.10 Imports 14](#_Toc101123587)

[1.6 Conclusion 15](#_Toc101123588)

# I Confidentiality Statement

This document is the exclusive property of ForEver25 and REALIZE & REACT Security (REALIZE & REACTS). This document contains proprietary and confidential information. Duplication, redistribution, or use, in whole or in part, in any form, requires consent of both FOREVER25 and REALIZE & REACTS.

REALIZE & REACTS may share this document with auditors under non-disclosure agreements to demonstrate penetration test requirement compliance.

# II Disclaimer

A penetration test is considered a snapshot in time. The findings and recommendations reflect the information gathered during the assessment and not any changes or modifications made outside of that period.

Time-limited engagements do not allow for a full evaluation of all security controls. REALIZE & REACTS prioritized the assessment to identify the weakest security controls an attacker would exploit. REALIZE & REACTS recommends conducting similar assessments on an annual basis by internal or third-party assessors to ensure the continued success of the controls

# III Contact Information

|  |  |
| --- | --- |
|  |  |

|  |  |  |
| --- | --- | --- |
| Name | Title | Contact Information |
| **Demo Company** | | |
| Jonathan Helmus | Instructor | Office: (555) 555-5555  Email: Jonathan.helmus@metrostate.edu |
|  | | |
| Edajei Alvine | Lead Penetration Tester | Office: (555) 555-5555  Email: vu8547fr@go.minnstate.edu |

# 1 Assessment Overview

From April 4th to April 17th, Realize & React engaged ForEver25 to evaluate the security posture of its infrastructure compared to current industry best practices that included an internal penetration test using python. All testing performed is based on requirements stated in tryhackme room, Python.

Phases of penetration testing activities include the following:

* Planning – Customer goals are gathered, and rules of engagement obtained.
* Discovery – Perform scanning and enumeration to identify potential vulnerabilities, weak areas, and exploits.
* Attack – Confirm potential vulnerabilities through exploitation and perform additional discovery upon new access.
* Reporting – Document all found vulnerabilities and exploits, failed attempts, and company strengths and weaknesses.

Diagram

Description automatically generated

# 

# Assessment Components

## 1.1.1 Internal Penetration Test

An internal penetration test is meant to identify what could be accomplished by an attacker who has internal access to your network. A Realize & React Engineer performs scanning and enumeration to identify potential vulnerabilities in hopes of exploitation.

## 1.2 Findings Severity Ratings

The following table defines levels of severity and corresponding CVSS score range that are used throughout the document to assess vulnerability and risk impact.

| Severity | CVSS V3 Score Range | Definition |
| --- | --- | --- |
| Critical | 9.0-10.0 | Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately. |
| High | 7.0-8.9 | Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible. |
| Moderate | 4.0-6.9 | Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved. |
| Low | 0.1-3.9 | Vulnerabilities are non-exploitable but would reduce an organization’s attack surface. It is advised to form a plan of action and patch during the next maintenance window. |
| Informational | N/A | No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation. |

# 1.3 Scope

|  |  |
| --- | --- |
| Assessment | Details |
| Internal Penetration Test | [Python Room](https://tryhackme.com/room/pythonbasics) |

* Full scope information provided in “**ForEver25.. Full Findings.xslx”**

## 1.3.1 Scope Exclusions

Per client request, REALIZE & REACT Security did not perform any Denial-of-Service attacks during testing.

## Client Allowances

FOREVER25 did not provide any allowances to assist the testing.

# 1.4 Executive Summary

REALIZE & REACTS evaluated FOREVER25’s internal security posture through an internal network penetration test from April 4th to April 17th. By leveraging a series of attacks, REALIZE & REACTS found critical level vulnerabilities that allowed full internal access to the FOREVER25 user accounts. It is highly recommended that FOREVER25 addresses these vulnerabilities as soon as possible as the vulnerabilities are easily found through basic reconnaissance and exploitable without much effort.

## 1.5 Summary of Tasks

## 1.5.1 Introduction to Python

In the code editor was the statement “print (“Learn security with TryHackMe!”)

After pressing the “run code” button the code output showed “Learn security with TryHackMe!”.

The word “print” before the brackets therefore tells the editor to output what is in quotes in-between the brackets.

Text

Description automatically generated fig1

1.5.2 Hello World

The code editor was used to print hello world using the line;

print(“Hello World”)

Output: Hello World

The print statement sends to the console all that is within the brackets right after the word “print”.

Graphical user interface, text, application

Description automatically generated fig2

## 1.5.3 Mathematical Operators

Mathematical operators work normally as they are, however unlike printing a sentence like that in fig1, if we want the operation to be performed, then the operation is put in the brackets without the use of quotes surrounding the numbers which are being operated in order to output the result

1. 21 + 43 == print(21 +43)

Output: 64

Graphical user interface, text, application

Description automatically generated fig3

1. 142-52 == print(142-52)

Output: 90

Graphical user interface, text, application

Description automatically generated fig4

1. Multiplication

print(10\*3420)

Graphical user interface, text, application

Description automatically generated fig5

1. Exponent: The exponent operates a little differently as it requires two multiplication operators for the output to return a correct answer

Therefore, any number p raised to the power of a number q will need to be entered as p\*\*q.

So 5 squared is entered as

print(5\*\*2)

Graphical user interface, application

Description automatically generated fig6

## 1.5.4 Variables and Data Types

Also, in the place of numbers being put directly into a print statement they can be printed from a variable. A variable can ease editing in case a single figure had to be changed at several places within code or there needed to be an update of a previous variable.

height=200

adding 50 to the height

height+=50

this way 50 is added to whatever number the height is set to be above and not being added to the 200.

To print the height now

print(height)

Also, unlike fig 1 & 2 above, to output a variable there need not be quotes surrounding it within the brackets.

print(height) will output the most updated value of height in order of occurrence from the top. So, it prints 250 because height was initialized at the top and then there was another occurrence of height plus 50.

Graphical user interface, text, application

Description automatically generated fig7

## 1.5.5 Logical and Boolean Operators

Logical and operators are if, else, elif (else if), and Boolean operators are true or false

These work together as the statement that follows a logical operator could be true or false, based on which other statements could be printed. For example, with the height from above, logical, and Boolean operators could be used to do something else if height was less or greater than 200.

## 1.5.6 Shipping Project

This makes use of the logical and Boolean operators, creating,

updating, and printing updated variables based on the conditions of the logical operators

Graphical user interface, text, application, website

Description automatically generatedfig8

When customer\_basket\_cost is changed to 101 and the code is rerun, the output now changes as customer cost becomes more than 100. The use of a variable also makes updating the value and output easier.

Graphical user interface, text, application

Description automatically generatedfig9

## 1.5.7 Loops

The loop does as action several with increments

For i in range (51):

print (i)

This outputs all numbers from 1 to 50. The number in bracket is the limit and the output will always be from 0 to the number in bracket (n) minus one i.e. (n-1) except otherwise a starting point is indicated. But in this case just stating print(i), the printing will start from 0 until 50-1.

Graphical user interface, text, application

Description automatically generatedfig10

## 1.5.8 Bitcoin project

This project uses a function. The function takes 2 parameters and makes the calculation using mathematical operators. I can tell that a function with the help of variables can make doing a repetitive action even easier.

Graphical user interface, text, application, chat or text message

Description automatically generatedfig11

When the value of the variable for bitcoin value is changed, the Boolean result changes and the function runs giving another output.

Graphical user interface, text, application

Description automatically generatedfig12

A function can make use of variables, logical and Boolean operators, and print statements to give output.

## 1.5.9 Files

This task opens a given file name. The file name must be exactly what the file was named to be with any special characters included.

Text

Description automatically generated with medium confidencefig12

The “r” must be indicated telling the compiler we are trying to read the file contents, so it opens the file in read-only mode.

Text

Description automatically generatedfig13

## 1.5.10 Imports

The “import” keyword helps make use of libraries within python. For example, importing the datetime library helps to print the current date and time when the code is executed.

Graphical user interface, text

Description automatically generatedfig14

## 1.6 Conclusion

Tasks were straight forward and provided a good understanding of how the python language works.

Also, printing a statement Is straight forward as just typing the print and the statement we want to print in a pair of parentheses after the statement. In some cases, like variables printing doesn’t require a pair of quotes as it does for strings.

Different tasks could be performed with a combination of loops, conditional statements (if, else, elif) in a function like in the shipping and bitcoin tasks. Files could be open in read, or write mode using a python read or write statement (f.read() or f.write()) as shown in fig 13 and finally the import statement helps us use already existing libraries like the date and time as shown in fig14.