TED University / CMPE492

Senior Project

Test Plan Report

Date

04-26-2022

Website of the Project:

https://ayse-nur-safak.github.io/HoneyTED/honeyted/index.html

GROUP MEMBERS	SUPERVISORS
AYŞE NUR ŞAFAK	EMIN KUĞU
KAYRA EGE ARDA	VENERA ADANOVA
BARAN AKIN	TOLGA KURTULUŞ ÇAPIN
EBRU KILIÇ	

Contents

1. INTRODUCTION

This document describes the plan for testing our honeypot system. Testing the systems we have installed for our cybersecurity research is one of the most important steps for the result and product. It is reevaluated according to how accurate the work done in the test step is and whether the features give the desired results.

1.1. SCOPE

Since our research consists of several steps, we need to test many structures. We need to test all the steps and features because all the steps and systems are interconnected with each other. The scope of the test is as follows:

A. Honeypot is installed and running smoothly.
B. To be in a state that is opened to the internet.
C. Accepting SSH and Telnet connections of attacker's machines to the system.
D. The accepted username and passwords must be from the created white list.
E. Saving the id and password parts of incoming connection attempts with details like time via SSH and Telnet.
F. Logging the commands written after the connection is established, along with the response of it.
G. Logging the moments when the connection is established and finished.
H. Checking the proper functioning of the established website to help it attract attackers.
I. Performance and stress testing of the website.
J. Parsing the collected log files according to their IP.
K. The results obtained give an output as a CSV file.
L. Uploading these CSV files to a database.
M. Using an API to get detailed information about the resulting IPs.
N. Identification of the found results with each IP and collection of these results in a database.
O. Statistical calculation of the results obtained.
P. The system runs fast.
Q. Absence of security vulnerabilities other than the desired security vulnerabilities of the system.
R. The system's reliability is measured by the uptime average of the system over time.

1.1.1. IN SCOPE

Usability is a very important attribute of the project. Installing the honeypot and running it without any obstacle is one of our priorities.

For security, we will check the accepted username and the password (authorization). Therefore, other unexpected people cannot access our system.

Reliability is another important issue. Testing the stress of the website and the performance of the project will increase the durability. We will look up the system's speed (but not about the hardware).

1.1.2. OUT SCOPE

Database Testing and Hardware Testing are out-scope testings. Since this project does not need too many hardware requirements, we do not need to test it. Average computer hardware can provide the system.

On the other hand, we do not have concerns about data loss. Therefore, for now, we are not going to do database testing.

Statistical calculations of the results are also out of the scope requirements. It is not something critical for the project's safety or dangerous for quality attributes.

1.2. REQUIREMENTS FOR TEST

The items (use cases, functional requirements, non-functional requirements) listed below have been designated as testing objectives. This is a list of what will be tested.

1.2.1. Data and Database Integrity Testing

The results which have been stored as CSV files shall be uploaded to a database.

1.2.2 Function Testing

Honeypot is installed and running smoothly.
To be in a state that is opened to the internet.
Accepting SSH and Telnet connections of attacker’s machines to the system.
The Honeypots shall take the log files which include the attacker’s information.

1.2.3 Performance Testing

The system runs fast.
The Honeypots that emulates a vulnerable web server shall catch up attacker activity.

1.2.4 Security and Access Control Testing

The system shall not be exploitable.
The honeypot needs to be reliable.
The honeypot should be self contained (it cannot become or create a vulnerability to the host system).
An attacker would only have limited access to the operating system. Because it is a much more static environment and 'low interaction' means that the attacker will not be able to interact with our fake system in any detail.

1.2.5 Configuration Testing

The honeypot shall run on a server that hosts an unix environment.

1.2.6 Load Testing

We are going to test the system by ddos attacking our own system. That will test the durability.

1.3. ROLES & RESPONSIBILITIES

This section describes the suggested resources for testing the installed honeypot system, as well as their primary tasks and knowledge or skill set.

Human Resources		
Role	Minimum Resources Recommended	Specific Responsibilities
Test Manager	Ayşe Nur Şafak	Provides management oversight Responsibilities: ● Managing the whole testing process. ● Providing all the needed resources for the testing activities
Test Designer	Ebru Kılıç	Identifies, prioritizes, and implements test cases Responsibilities: ● Generate test plan ● Evaluate effectiveness of test effort
System Tester	Baran Akın	Executes the tests Responsibilities: ● Execute tests ● Log results ● Recover from errors ● Document defects
Test System Administrator	Kayra Ege Arda	Ensures the test environment and assets are managed and maintained. Responsibilities: ● Administer test management system ● Install / manage worker access to test systems
Database Administration/Database Manager	Ebru Kılıç Ayşe Nur Şafak Baran Akın Kayra Ege Arda	Ensures test data (database) environment and assets are managed and maintained.

		Responsibilities: ● Administer test data (database)
Implementer	Ebru Kılıç Ayşe Nur Şafak Baran Akın Kayra Ege Arda	Implements the specified testing Responsibilities: ● Test the system

2. TEST METHODOLOGY

The Test Methodology describes the suggested technique for testing the installed honeypot system. The preceding section on Requirements for Test detailed what would be tested; this part outlines how this will be tested.

2.1. OVERVIEW

Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs and improving performance.The project must have been tested both separately and as a whole. The fact that the individual parts that make up the systems give the desired result, as a result of their working together, causes the system to give us the desired result. Therefore, it is a great convenience for us to test all these singularities and their work together with a specific combination, and to detect the errors and problems that arise, and to provide us with a desired and reliable result. Currently, many methods and tools are used to test it. These are divided into the following headings, and the tool/s that will be used for this test are mentioned in their descriptions.

2.2. TEST LEVELS

We will check our application one by one according to the 6 different test methods given here, as well as the combination of these units and as a whole primarily. These tests depend on the scope of the project, time, and budget constraints.

2.2.1. Unit Testing

Since this is a research project, we cannot test extra subunit code in the system. Honeypot already has working codes inside of it. But about the website, we did unit testing on HTML codes. We set up a unit with input, make it do its duty and after all, we check the output of it. It is basically where the input could be a function parameter, and the output just the return value. We found some bugs inside of the code that are not showing on the website but standing behind the background. Bugs have been reported and fixed.

2.2.2. API Testing

We will test our APIs like they are a black box, because the most important point for our APIs will be their security, so with that in mind we are going to be performing an black box penetration test and a fuzz test as a worst case denial of service attack.

2.2.3. Integration Testing

Since our system is small and we nearly have all the modules to test we have decided to use the big bang approach for our integration testing methodology, in this test methodology we test all the components together, since the system is small the disadvantage of this method is eliminated, with the small complexity of the added components and with the timesavings that this method brings we have decided to use this approach.

2.2.4. System Testing

For our system testing, because everything has been built on top of an existing project we have little worries about the interactions between the few newly added components and the system itself, our main concern here is the throughput, which will be tested by an pseudo loading of the system, by swarming our own system with “users” we will have an idea on how does the integration handle many request and the general throughput of the system.

2.2.5. Performance & Stress Testing

Performance testing measures response times, and other time sensitive requirements. The goal of Performance testing is to verify and validate the performance requirements have been achieved. There are several tools in order to perform these types of tests. The most popular ones are JMeter ve Slowloris. For these tests, these tools can be used for both performance and stress tests.

2.2.6. Beta Testing

Since this is a research project and will not be sold or deployed commercially and is for the sole purpose of gaining new insights in this industry and furthermore because it has already been deployed partially to gain information we will not have an beta version.

3. RISK & ISSUES

Our Risks are system downtime and exploitation of the secure component. system downtime should not occur more than %10 percent of the total runtime and it is not a big issue, since we have physical access to the system even if it goes down we can always boot it up or replace hardware if needed. (hardware replacements should be extremely rare).
But the exploitation of the secure part of the machine is a big problem if it happens and for the occurrence probability we cannot be certain, depending on the methods and the aim of the malicious actors who do gain access to the system we could lose data or our project wiped all together. To combat this we would take backups using the 3 2 1 method. (3 backups 2 on site one of them is on an seperate machine and 1 off site.) This is the minimal requirement if you need to keep backups properly. Other than that and monitoring the traffic on that separate network there is not much action that can be taken if this event were to occur. All we can do is try to discover the threat, if we catch it, stop it. (manual disconnection works really well since it is not an critical application and we have access to the machine itself) and take mitigation steps to reduce the losses in the event's occurrence (ex; backups included but not limited to).

4. TEST ENVIRONMENT

The tests will be run on the same machine and the environment, that will be the environment that we will ship the product on and do our research, so we would not need to worry about compatibility issues. and because we are essentially testing a live system we would not need to worry about migrating or calculating overheads on a new system. Furthermore, because this is a research project the application and the server itself is not critical. Thus, the test environment is for our benefit.

5. TEST SCHEDULE

Testing the…	Timeline	Current status
Honeypot is installed and running smoothly.	01/04/2022 - 03/04/2022	tested
To be in a state that is opened to the internet.	01/04/2022 - 04/04/2022	tested
Accepting SSH and Telnet connections of attacker's machines to the system.	05/04/2022 - 09/04/2022	tested
The accepted username and passwords must be from the created white list.	08/04/2022	tested
Saving the id and password parts of incoming connection attempts with details like time via SSH and Telnet.	10/04/2022 - 12/04/2022	tested
Logging the commands written after the connection is established, along with the response of it.	15/04/2022 - 18/04/2022	tested
Logging the moments when the connection is established and finished.	20/04/2022 - 24/04/2022	tested
The results obtained give an output as a CSV file.	20/04/2022 - 24/04/2022	tested
Testing the website if it is working or not.	20/04/2022 - 24/04/2022	tested
Identification of the found results with each IP and collection of these results in a database.	29/04/2022 - 05/05/2022	not tested
Statistical calculation of the results obtained.	05/05/2022 - 08/05/2022	not tested
The system runs fast.	05/05/2022 - 08/05/2022	not tested

Absence of security vulnerabilities other than the desired security vulnerabilities of the system.	05/05/2022 - 08/05/2022	not tested
The system's reliability is measured by the uptime average of the system over time.	09/05/2022 - 16/05/2022	not tested
Uploading these CSV files to a database.	09/05/2022 - 16/05/2022	not tested
Using an API to get detailed information about the resulting IPs.	09/05/2022 - 16/05/2022	not tested
Checking the proper functioning of the established website to help it attract attackers.	09/05/2022 - 16/05/2022	not tested
Performance and stress testing of the website.	16/05/2022 - 23/05/2022	not tested
Parsing the collected log files according to their IP.	16/05/2022 - 23/05/2022	not tested

6. PROCEDURES

Developing the test plan and seeing the path. Keep the content varied and supported as much as possible.
Test the software. Firstly check the functional testing by using Unit Testing, Integration Testing, System Testing, Acceptance Testing, API Testing, and Performance & Stress Testing.
Testing the non-functional requirements like usability, and performance.
Test the software design. Design should be better to be effective and efficient.
Execute all and record the result that we have.
Reporting the test results. This allows us to see where we are wrong.
Fixing the bugs and other stuff that we want to improve.
Test the software changes. After fixing the mistakes, we will check again to be sure if there is another bug.
Evaluate test effectiveness.

7. TESTED / NOT TESTED FEATURES

7.1. TESTED FEATURES

Honeypot is installed and running smoothly.
To be in a state that is opened to the internet.
Accepting SSH and Telnet connections of attacker’s machines to the system.

The accepted username and passwords must be from the created white list.
Saving the id and password parts of incoming connection attempts with details like time via SSH and Telnet.
Logging the commands written after the connection is established, along with the response of it.
Logging the moments when the connection is established and finished.
The results obtained give an output as a CSV file.
Testing the website if it is working or not.

7.2. NOT TESTED FEATURES

Identification of the found results with each IP and collection of these results in a database.
Statistical calculation of the results obtained.
The system runs fast.
Absence of security vulnerabilities other than the desired security vulnerabilities of the system.
The system's reliability is measured by the uptime average of the system over time.
Uploading these CSV files to a database.
Using an API to get detailed information about the resulting IPs.
Checking the proper functioning of the established website to help it attract attackers.
Performance and stress testing of the website.
Parsing the collected log files according to their IP.