TED University/ CMPE491

Senior Project

High-Level Design Report

Date

01-16-2022

GROUP MEMBERS

AYŞE NUR ŞAFAK

KAYRA EGE ARDA

BARAN AKIN

EBRU KILIÇ

SUPERVISOR

EMIN KUĞU

JURIES

VENERA ANDANOVA

TOLGA KURTULUŞ ÇAPIN

CONTENTS

1. INTRODUCTION

The rising vulnerabilities are one of the most regarding situations of the day. Therefore, every day, billions of attacks on small, medium, and large networks appear around the world. Some detection tools have been developed to protect networks by gathering information from previous malicious attacks. It is necessary to approach this problem with many different solutions because the habits of attackers, the way they attack, and their goals are changing every day. Therefore, a honeypot is one of these tools which traps attackers by using fake data that a hacker will attempt to steal from. In this way, it offers an option that we can analyze them, except that we have only tricked and caught them. In this project, we will design a low-interaction, SSH honeypot. It will be installed on a public server to investigate cyber-attack methods. This will gain the protectability and help the people who defend their system.

1.1. PURPOSE OF THE SYSTEM

The purpose of the system is to collect data about the behaviors of real-world attackers, using what is essentially a “trap” and by recording how they react when they fall into this trap, we can generate useful data about attack patterns and attack vectors. Later, by analyzing and processing this data we can predict future behaviors of malicious actors.

1.2. DESIGN GOALS

The goal of the design is such that to design a system that will be exploited “purposely” the trap and have a secure part of the system that holds the data for later analysis and usage. Ideally, we want the data to be secure and safe, and potentially offload the data to another device in between arbitrary intervals. Therefore, we know for certain that the data that is collected cannot be used for nefarious purposes by third parties.

1.3. PROJECT CONSTRAINTS

The constraints of this project are to have a system to “be hacked” and for people to roam around in such a system so that the behaviors can be saved. Finding companies or hosting services to host such a project has proven difficult to say the least. Another constraint is the security of the data that is collected. We must segment the system carefully to have a part of the system that will be vulnerable, and the data storage part be safe and secure.

1.4. PROFESSIONAL AND ETHICAL RESPONSIBILITIES

Our professional responsibility lies in the security and the anonymity of the data that is collected. We must be certain that the data that is collected cannot be accessed by third parties, unless the law states otherwise. If the data, we keep is de-anonymized and is accessed by third parties we might be doing acts that are against data protection laws and this would both be non-professional and unethical. Never mind the illegality of such an act.

2. CURRENT SOFTWARE ARCHITECTURE

Kojoney is a low interaction SSH honeypot. It is developed in Python and is implemented on the Twisted and Conch libraries, which provide SSH server and TCP/IP functionality. Kojoney generates a very realistic SSH server on a host system; whenever an attacker authenticates to Kojoney, they are trapped in the honeypot instead of being directed to a shell. This shows that once an attacker signs in, they can only communicate with Kojoney, not the actual host system. In addition to these, it may be configured to allow any number of user accounts and passwords. It will store all attempts to authenticate, including unsuccessful login attempts, and will keep track of the attacker's IP address and the accounts examined.

Once authenticated in, an attacker may submit a strict subset of instructions, to which Kojoney will react with pre-defined text. The curl and wget commands are prominent exceptions to this rule. When an attacker performs one of these instructions, Kojoney will download the specified files into a predetermined location for analysis. However, the attacker is never able to see or interact with files obtained in this manner. The downloads are securely stored so that administrators may inspect them, but attackers cannot access them.

We may simply increase Kojoney's "interactivity" to make it more resemble a high interaction honeypot while preserving the safety of using a low interaction honeypot by upgrading it.

3. PROPOSED SOFTWARE ARCHITECTURE

Cowrie is a medium to high interaction SSH, and Telnet honeypot designed to log brute force attacks and the shell interaction performed by the attacker. In medium interaction mode (shell) it emulates a UNIX system in Python, in high interaction mode (proxy) it functions as an SSH and telnet proxy to observe attacker behavior to another system.

3.1 OVERVIEW PROPOSED SOFTWARE ARCHITECTURE

Cowrie is a medium to high interaction SSH, and Telnet honeypot designed to log brute force attacks and the shell interaction performed by the attacker. In medium interaction mode (shell) it emulates a UNIX system in Python. Also, it basically records the movements of the attacker who is allowed to connect to the system and saves them in a database. It can also display this information with a graphical interface optionally.

Software requirements of the cowrie:

The software is needed to run locally:

Python 3.7+

python-virtualenv

some python dependencies which are:

appdirs==1.4.4

attrs==21.4.0

bcrypt==3.2.0

configparser==5.2.0

cryptography==36.0.1

packaging==21.3

pyasn1_modules==0.2.8

pyopenssl==21.0.0

pyparsing==3.0.6

python-dateutil==2.8.2

service_identity==21.1.0

tftpy==0.8.2

treq==21.5.0

twisted==21.7.0

Why do we decide to change our current system?

The current system which is Kojoney is a very old library and it has not been upgraded in a long while. It uses python 2.7 which is a very old release and can lead to some system vulnerabilities. We met some problems when we configurate the Kojoney to our system. Therefore, we decided to change our system as cowrie which is an upgraded version of Kippo. It is based on python too. Also, it is better suited for our goal, which is to collect data on the actions of nefarious users. It is also more modular than Kojoney, and this modularity allows us to complete our tasks and requirements easier.

3.2 SUBSYSTEM DECOMPOSITION

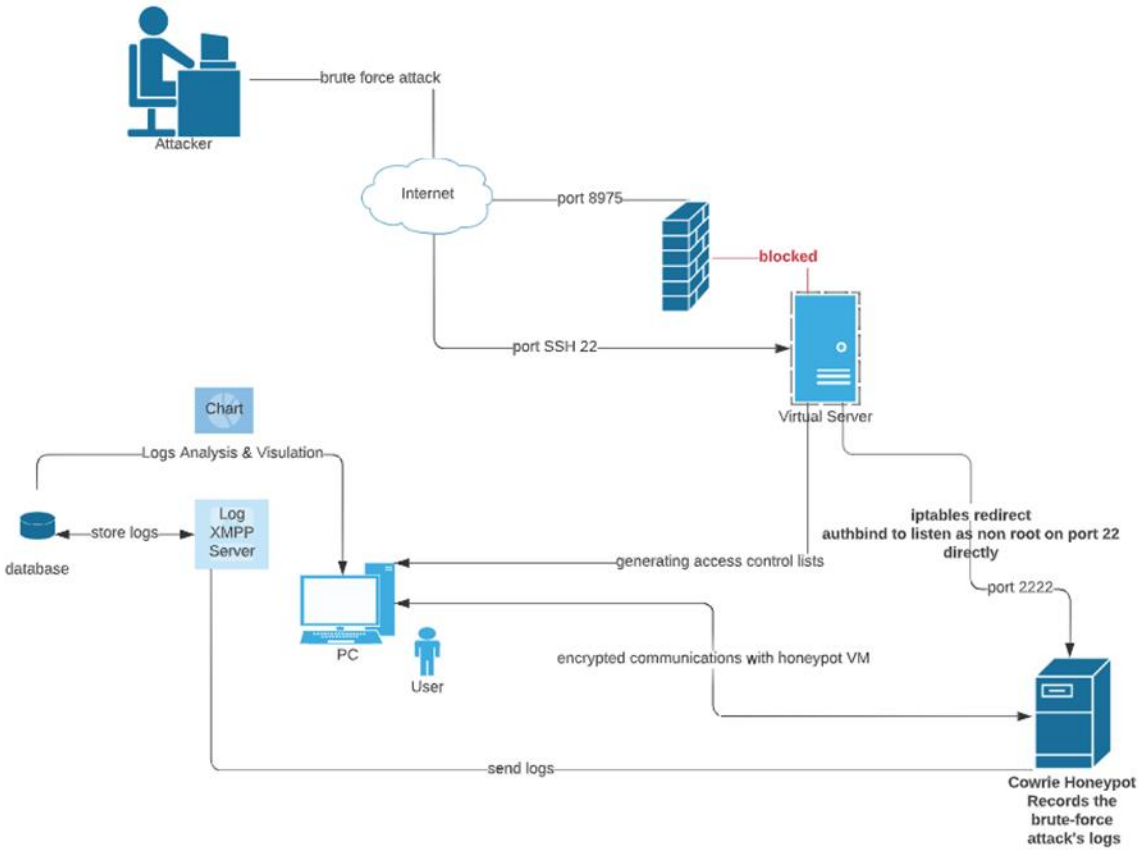
Long term data storage: This subsystem is responsible for the holding of the gathered data for a long-term storage, and the security of the said data.

Tracking: This sub system is designed to track the trapped bad actors and record their actions in our system to be later turned into data to be analyzed.

Trapping: The subsystem designed to purposely be exploitable by bad actors to trap them in our system to be analyzed.

Analysis: This subsystem has the responsibility to detect or find activities which are considered an attack. It is responsible for monitoring the status of the system also. We will use some plugins to convert data to a machine readable and a human readable form.

3.3 HARDWARE/SOFTWARE MAPPING



3.4 PERSISTENT DATA MANAGEMENT

We would like to offload the data to another server with irregular intervals to minimize the data that will be on a potentially compromised machine. the irregularity of the intervals is due to the fact that nefarious actors might be on the lookout for the pattern of the data leaving the server and then they might try to escalate their privilege to the secure servers. and this is the reason also that we would not want a linked database between two servers or a constant connection between them.

3.5 ACCESS CONTROL AND SECURITY

The admin access will be given to our group members and other collaborators of the project dependent on their degree of involvement with the project. the decision to give access to the systems and unanalyzed data will be withheld by the creators unless the law states otherwise. Other than that, all the reports that we will generate will be anonymized from all personal details including but not limited to; Ip addresses, mac addresses, location data. Access control and

security describe the user model of the system in terms of an access matrix. This section also describes security issues, such as the selection of an authentication mechanism, the use of encryption, and the management of keys.

3.6 GLOBAL SOFTWARE CONTROL

In general, there are two fundamentals of the subsystems of the project. The first one is the Analysis subsystem. This system has the responsibility to detect or find activities which are considered an attack. It is responsible for monitoring also.

Another significant subsystem is the Reaction subsystem. It has a responsibility for all the actions which are captured after detection by the Analysis subsystem. It shows the actions of the hacker which were monitored and recorded by the Analysis subsystem. Therefore, both subsystems are working in synchronization.

To further clarify, after the attack, the Analysis subsystem listens for events. After cloning the data, it stores process information into the monitoring list. After all these, Reaction Analysis takes the data from Analysis and reports it to us.

3.7 BOUNDARY CONDITIONS

Honeypot has some limitations in legitimate systems. It may not be able to detect these security breaches. It may fail to track the hacker. Also, there is another risk. Having exploited the honeypot, an attacker can escalate his privilege or make lateral attacks on the system itself. These attacks can be used to infiltrate the real production network. We must be sure that, Honeypot is adequately isolated.

The startup procedure should also be isolated, give warnings and do not start if a critical error exists. should check the checksums of the files before restarting again. (For delayed logic bombs or kernel attacks.) and also should start the honeypot service on bootup.

For the shutdown procedure it should reset the system status to the original which is read only to keep it from corrupting. which is why it is crucial that we pull the data from the system frequently. (Restoring to the start status further hardens the system against possible overtakes.)

For the error behavior of the system, the system should ideally be frozen and require manual intervention, that is because the system might have been overtaken or it might require our intervention. Also, in the future we are planning to automate more of the steps and intervention requirements to further streamline the usage of the honeypot.

4. SUBSYSTEM SERVICES

Although there are many subsystems that we will use, the subsystem services that we will determine for the moment may be incorrect and inconsistent because it has not yet been

completely clarified. Because of the system change we made; we have not yet been able to determine an interface for our subsystems. After the SDD part has been fully decided, it will be updated in the future reports. It has been left empty for the moment for these reasons.

5. GLOSSARY

In this section, there are explanations of the terms related to the project or system mentioned in the report.

Networks: A network is a technology that connects various devices, mainly for the purpose of data sharing.

Honeypot: Honeypots are trap servers that serve to collect information about attackers or users who have unauthorized access to information systems.

Malicious Attacks: They are attacks and software that are used to disrupt the functions of computers and mobile devices, collect critical information, provide access to special computer systems, and display unwanted ads.

Low-Interaction: Low-interaction honeypots use fewer resources; they collect basic information about the level, type, and where the threat is coming from.

SSH: SSH, or Secure Shell, is a remote management protocol that allows users to control and edit their servers over the Internet.

Attack Vectors: It is defined as the technique by which unauthorized access to a device or network can be provided by hackers for unfair purposes.

Offload the Data: Data offloading is the use of complementary network technologies to deliver targeted data for networks.

SDD: Software Design Document.

Vulnerable: vulnerability; errors that make a system or application vulnerable to cyber-attacks are called.

Anonymized: This means that data cannot be associated with an identified or identifiable natural person, even if it is matched with other data.

De-anonymized: The reverse process of anonymization.

Anonymity: It is the unidentified, unreachable, untraceable, and unknowable identity of a person or object.

Cowrie: Cowrie is a medium interaction SSH and Telnet honeypot library.

Python: The high-level language. The version 3.7+ will be used.

6. REFERENCES

1. [1] S. Russell and P. Norvig, Virtual honeypots: from botnet tracking to intrusion detection, 3rd ed. Addison-Wesley Professional, 2007.
2. ACM Code of Ethics and Professional Conduct.
3. The Software Engineering Code of Ethics, IEEE Computer Society.
4. IEEE Code of Ethics.
5. Computer and Information Ethics, Stanford Encyclopedia of Philosophy
6. https://github.com/cowrie/cowrie
7. https://cowrie.readthedocs.io/en/latest/index.html