TED University/ CMPE491

# Senior Project

## Project Analysis Report


| GROUP MEMBERS |
| --- |
| AYŞE NUR ŞAFAK 16271073704 |
| KAYRA EGE ARDA 51508242928 |
| BARAN AKIN 36194302446 |
| EBRU KILIÇ 1444920268 |

| SUPERVISOR |
| --- |
| EMİN KUĞU |

| JURIES |
| --- |
| VENERA ANDANOVA |
| TOLGA KURTULUŞ ÇAPIN |


Date
12-1-2021

# PROJECT ANALYSIS REPORT

## 1. Introduction

The rising vulnerabilities are one of the most regarding situation of the day. So 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. A honeypot is one of these tools which traps attackers by using fake data that a hacker will attempt to steal from. In this project, we will design a low-interaction, SSH honeypot. It will be installed on a public server in order to investigate cyber-attack methods. This will gain the protectablity and help to the people who defend their system.

## 2. Current System

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 of 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 hone ypot while preserving the safety of using a low interaction honeypot by upgrading it.

## 3. Proposed System

For now, because this application is running on the server side, we are planning to use a cli (command line interface) as our way to interact with the application, and the reports will be in xml format. we may or may not implement a GUI (graphic user interface) as the project advances.

### 3.1 Overview

This project is basically for tracking movements of the attacker in the system. We will do it with Honeypot which is behaving like servers. They focus on detecting vulnerabilities in internal networks. Thus, it deceives hackers. So that, we will research potential attacks and analyze them. We will watch the actions and report how they behave.

# PROJECT ANALYSIS REPORT

We will use Kojoney for interaction honeypot which is using Phyton. We are going to use our collected data to assist the manifestation or improvement on some IDS and IPS systems. These data help us to predict and learn the behaviors and steps of bad actors.

## 3.2 Functional Requirements

- SSH honeypot shall perform as a server in a Linux environment.
- The Honeypots that emulate a vulnerable web server shall catch up attacker activity.
- The Honeypots shall take the log files which includes attacker's information.
- Attackers' connection than shall either dropped
- Attackers shall see the network admins messages.
- The Honeypot shall keep the commands that attackers execute in the system
- Honeypot shall give the report of the actions and information of the attacker.

## 3.3 Nonfunctional Requirements

- The system shall be stable.
- The system shall not be exploitable.
- An attacker shall 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.
- The honeypot shall need to be reliable.
- The honeypot should be secure (it cannot become or create a vulnerability to the host system).
- The Honeypot shall not use significant resources to maintain.

## 3.4 Pseudo Requirements

For this project, mainly C language or Phyton are using. We choose to do our project with phyton-based. One of the main reasons is Phyton has better library, and this make it more usable. Also, it has better server-client base. It is more long-lasting than C language because Phyton is still developing. It is supportable in more area and easier to access.

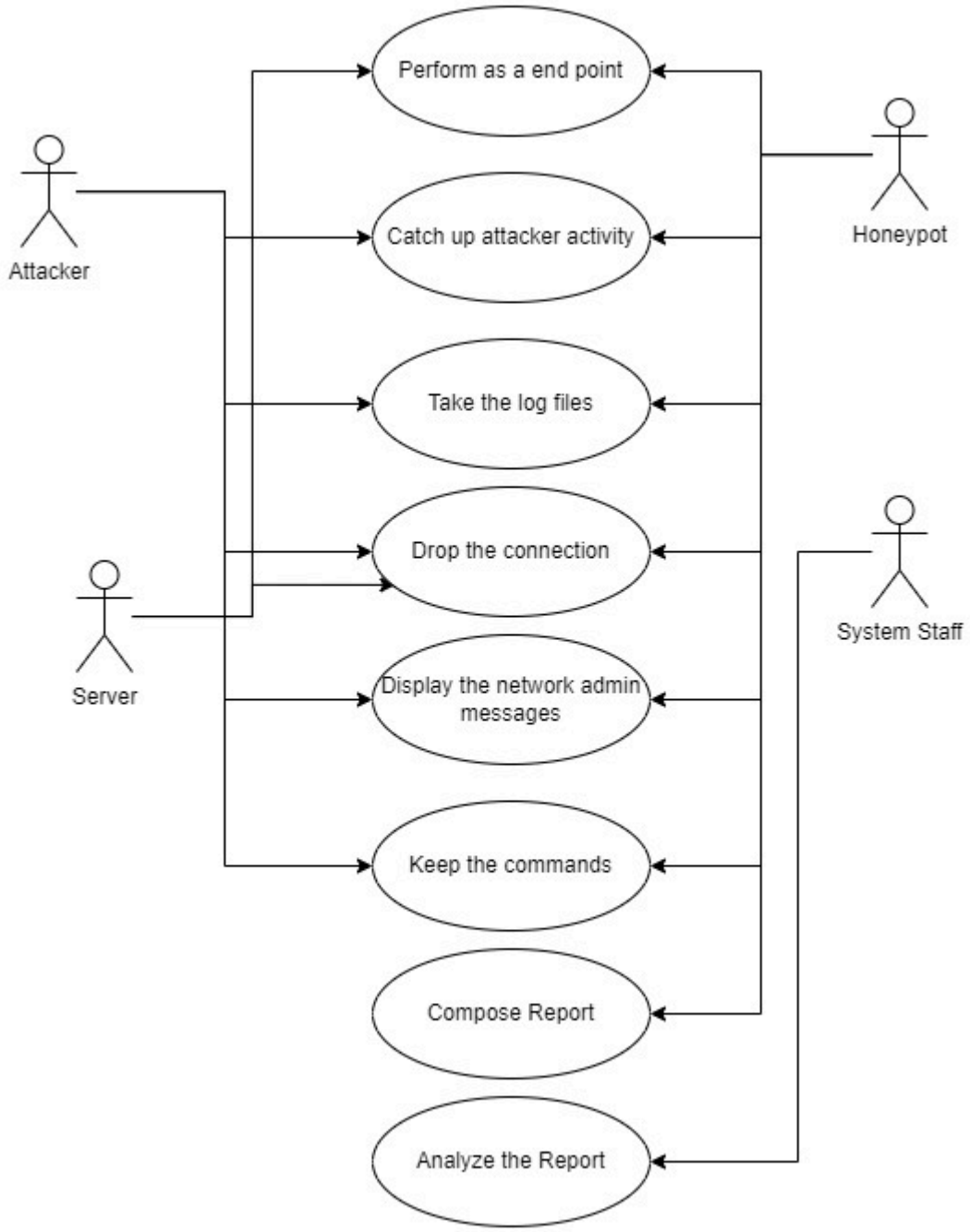
We are using Kojoney for interaction honeypot. It has high interaction, and it can change more comfortable. Kojoney is open to development. It is changeable. Another significant point is, the curl and wget commands are prominent exceptions. When an attacker performs one of these instructions, Kojoney will download the specified files into a predetermined location for analysis. It is SSH based, and it has GNO license. On the other hand, it is usable in Linux, some others may not use in Linux.

## 3.5 System models

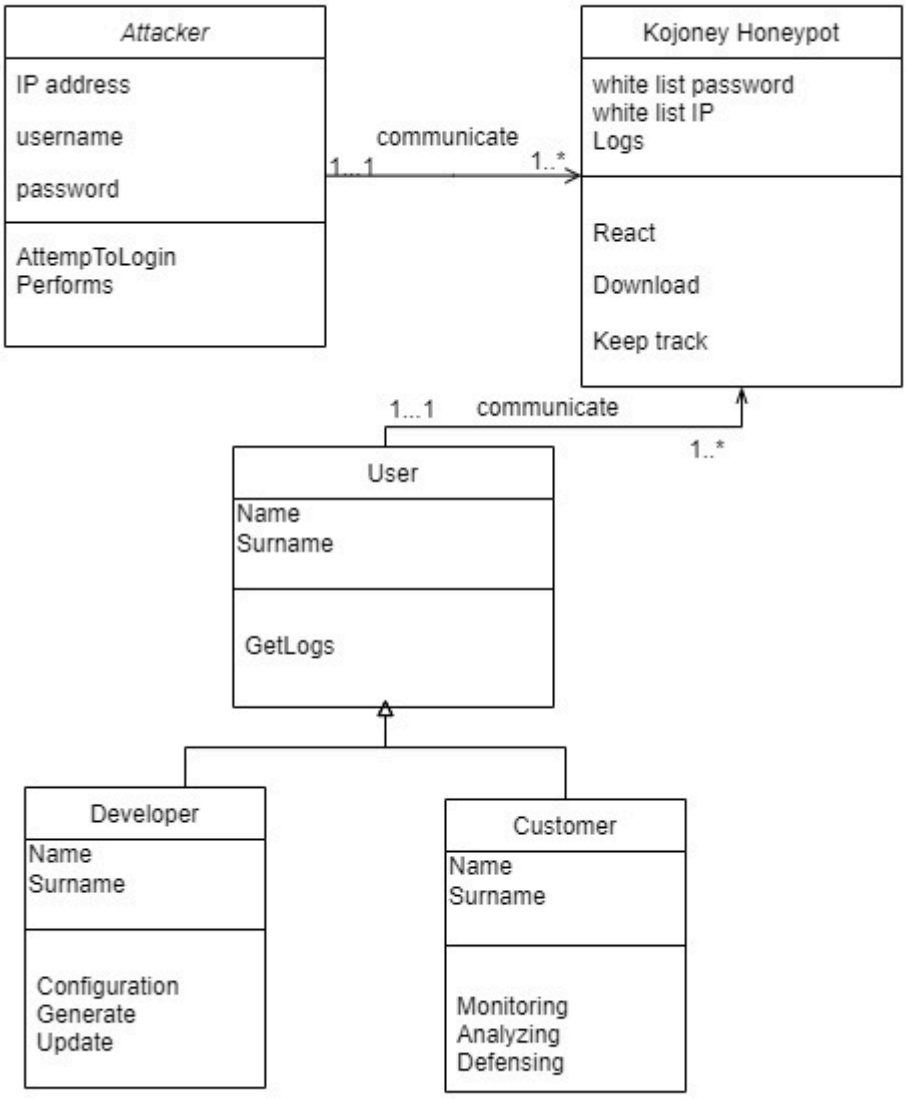
### 3.5.1 Scenarios

We start a timer for each connection to be cut off eventually and send a notification in the event of a connection, but if it is necessary and we want to observe that attacker some more we can manually override the timer to keep the connection up
If they try to get a reverse shell using an upload exploit, we quarantine the file and save it for further inquiry.

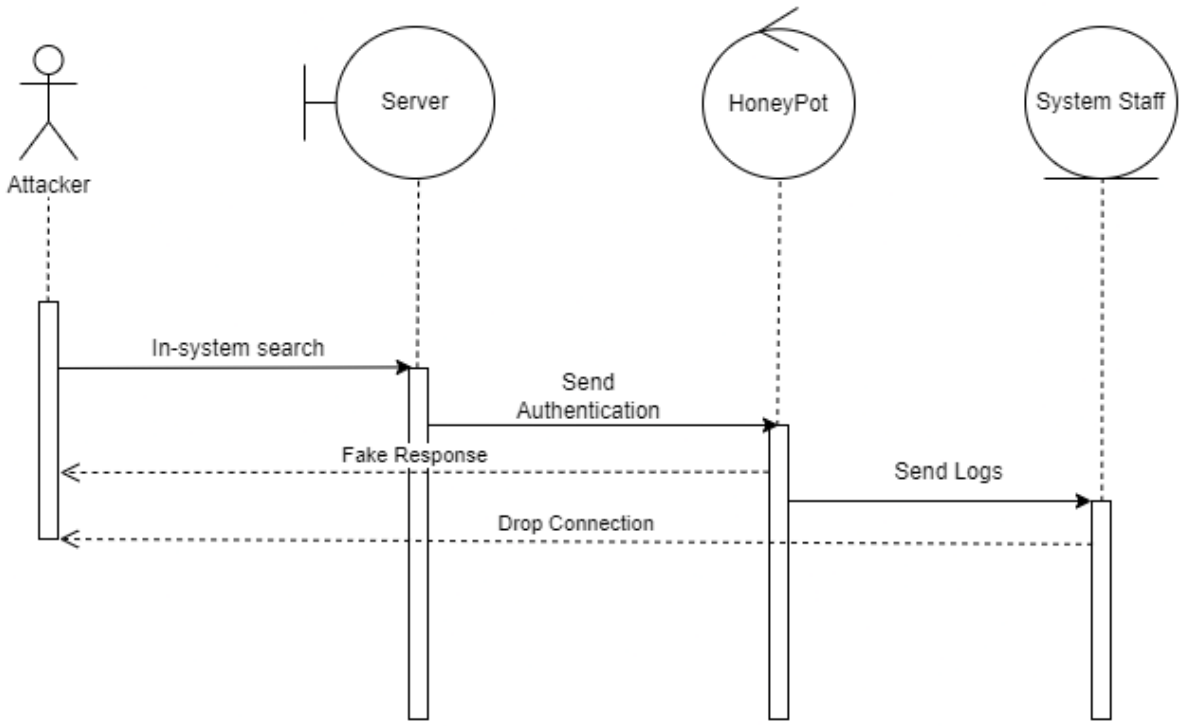
### 3.5.2 Use Case Model



### 3.5.3 Object and Class model



### 3.5.4 Dynamic models




### 3.5.5 User interface - navigational paths and screen mock-ups

For now, because this application is running on the server side, we are planning to use an CLI as our way to interact with the application, and the reports will be in XML format. We may or may not implement a GUI as the project advances.

## 4. Glossary

- Honeypot: We can summarize this term as a trap for hackers, it has a vulnerability that we want the hackers to exploit so that we can record their actions in our system.
- SSH: Is an encrypted network protocol for operating and accessing network services securely over an unsecured network.
- Kojoney: Is a honeypot library that has been implemented using python and is the backbone of this project.
- GNO: Stands for General Public License which gives the freedom to run study share and modify the existing software.
- Wget: Is a computer program that is used to retrieve content from web servers.
- Curl: Is a command line tool to exchange data to or from the server.
- Attacker: The hacker or malicious actors that we are trying to observe, their goal is to “hack” a vulnerable server, which in this context is our honeypot.

- Vulnerabilities: Being able to take advantage of and exploitable by hackers, an error in security, though in this context it is paramount for the system to work and is created deliberately.
- Low interaction: In the context it means that the hacker will not be able to deeply interact with the system which is a decoy.
- IDS: Stands for Intrusion Detection System. It detects when an unauthorized user or a malicious actor has entered the system.
- IPS: Stands for Intrusion Prevention System. It prevents unauthorized users or malicious users from entering the system in the first place.
- Reverse Shell: It is used to execute commands on the behalf of the hacker, if a hacker creates a Reverse Shell, it means that we have a major breach in our system.

- GUI: Graphical User Interface
- CLI: Command Line Interface

## 5. References


- [1] S. Russell and P. Norvig, Virtual honeypots: from botnet tracking to intrusion detection, 3rd ed. Addison-Wesley Professional, 2007.
- ACM Code of Ethics and Professional Conduct
- The Software Engineering Code of Ethics, IEEE Computer Society
- IEEE Code of Ethics
- Computer and Information Ethics, Stanford Encyclopedia of Philosophy
- http://kojoney.sourceforge.net/
- https://github.com/madirish/kojoney2