HONEYPOT AUTOMATED ATTACK DETECTING INTERFACE

PROJECT REPORT

5/2/2020

Team members:
Khang Le, Kenneth Maguire, Yarely Ochoa,
Nghi Phan, Dominic Vu, Gordon Zhong.
Project Supervisor:
Dr. Haadi Jafarian
Kuntal Das

TABLE OF CONTENTS

1. INTRODUCTION

In this day and age, cyber-attacks are very prevalent. For the people working on the defense side of Cyber Security, it is a game of knowing your enemy. With this in mind, we have built a standalone system for controlling honeypots, which capture cyber-attacks and relay that information to the network admin. Dockers are used to scale multiple honeypots across the network. We have created a dashboard tool to enable user's better functionality to start and stop honeypots. We designed and developed the tools for analyzing logs to identify and visualize potential attacks and intrusion attempts.

1.1 Summary of Project

Using containers to isolate software from its base environment and ensuring it works uniformly across different instances. This enables users the flexibility to capture attacks on their network easily. We want to capture network attacks and find a trend as to which attacks are most common. Using the data capture tool, we can develop a trend of popular attacks which can be used to improve defenses for future attacks.

2. Assumption and Constraint

Assumptions:

- Need to host program on a Linux system
- Have MongoDB, Altair, Flask and Docker technologies configure

Constraints:

- Available hardware resources for testing and development.
- Compatibility of frameworks with the architecture.

3. Scope Management

These are the requirements that need to be met while developing Honeypot Automated Attack Detecting Interface.

- The framework must provide real-time statistics about status of dockers
- The framework must allow for a GUI-based management of dockers
- The framework must have a visualization dashboard that uses graph and chart to identify attack patterns and visualize them.
- Needs to identify and visualize potential attacks and intrusion attempts.

4. Work Structure Breakdown

Task 1 - Designing Docker Management Framework

A Web-based GUI that has the following capabilities

- Browsing through the docker images, and allowing for run/stopping a docker image on an IP/port
- Allowing for observing and customization of dockers configuration. For example, external location for log storage.
- Allowing for customization of other important configuration including assigned memory, CPU, etc.
- A component for observing current status of dockers (down, up, faulty, etc).
- Admin authentication (just one user is enough).
- TLS-based access to GUI

Task 2 - Building a repository of honeypots

- Having 3 tested and function honeypots that can generate logs
- Converting these honeypots into docker images
- Determining default configuration for customization, especially external logging of events.

Task 3 - Log Synthesis and Visualization Engine

The Web-based GUI that used the log files to visualize data

- A visualizations tool that includes a live threat map based on source IP address to determine the destination of the attack
- Bar graph to visualize attack time and attacker pattern

4.1 Deployment Plan

The final local webpage will have a group of honeypots that runs within a system and can be hosted locally and generate logs. There will be a logging tool to extract the data from the honeypots. It will track traffic and captures different attacks on the system. The system will have a UI that can start and stop the docker containers and keep tracks of which dockers failed. There will be graphs to visualize these data.

5. Quality Management

We will be dependent on Docker, Altair, Flask and MongoDB tools available on the Internet. There will be open-sources tools that will need to be debug and quality check. We need to quality check both low and high interaction honeypot in the systems to ensure the functionality is consistent. The logging and data management tools will be developed by students in this team. The locally hosted UI will also be design by students of this team to ensure a smooth user experience.

6. DISCUSSION

We were able to reach our objectives for implementing this honeypot project. Each individual task was meet for building the overall honeypot deployment product. We were able to utilize Flask to implement the interface to control the deployment of dockers. This interface displays aggregated data from the deployment honeypot and provides information such as CPU usage, RAM usage and free disk space available. There are 3 honeypots that were fully tested and are functional to generate logs. We converted these honeypots into their own Docker container and tested these Docker images to ensure the functionality of the file. Utilizing Altair, we also parsed the logs and displayed graphs based on the data captured from the attacks. The final product is a webpage hosted locally that contains all these functionalities.

7. FUTURE IMPLEMENTATION

For future deployment, we would want to deploy this framework in a real-world setup, potentially inside CU Denver school network, for collecting and visualizing internal intrusions and attacks. This will be helpful in helping a system track and see which attacks are most common and help develop a tool that can help potential attacks from happening. We want to optimize the UI by allowing users to add and customize honeypots with various parameters such as port and IP address instead of just the given parameter. We believe this will help cater to individual user experience and provide a better interface to work with. We think that this project can be improve with the implementation of machine learning models that analyze attack trends. This can be useful because it can help train the system to become smarter and more efficient in detecting attacks.