Low Interaction Honeypot for Network Traffic-Monitoring

Leah T. Powell
Dept. of Computer Science
Embry-Riddle Aeronautical University
Daytona Beach, USA
powelll7@my.erau.edu

Colton D. Underwood
Dept. of Computer Science
Embry-Riddle Aeronautical University
Daytona Beach, USA
underwc9@my.erau.edu

Abstract— **In cybersecurity, a honeypot is essentially a bait-trap designed as a mimic of a real target to attract malicious cyber attackers into attacking it in order to gain information about the attackers and how they operate throughout the attack. The honeypot in this project is classified as a low-interaction honeypot where it only gives the attacker very limited access to the mimicked operating system, such as they will be unable to interact in-depth past a small emulation of internet protocols and network services. This choice of allowing them to be deceived and no more was chosen after an assessment of the project timeline scope and skills of the assigned developers within the project. The goal was ultimately to engage in a hands-on learning opportunity where developers can better understand how low-interaction honeypots are effective in capturing the network traffic of attackers. The tools used in this project were Wireshark for traffic analysis and Visual Studio Code for coding in python and CSS, as well as ubuntu Linux and a SEEDlabs virtual environment. As a result, the simulated website created by developers was able to efficiently expose and log the actions and internet behavior of the simulated attacker host.**

Keywords—Honeypot, Network Security, Cybersecurity, Scapy, Wireshark, SEEDlabs

I. Introduction

Cybersecurity honeypots were initially deployed in the 1980s and 1990s and were inspired by human honey traps where people would use romantic guises to gain information, benefits, or otherwise enact political motives. In cyber security, the approach is the same and involves traps and decoys placed around seemingly critical systems [2]. The envisioned pipeline is the system is propped up as a real system, the attacker finds purposefully placed weaknesses and attempts to penetrate them, and these actions of penetration are logged and later analyzed by engineers who can use the data for strengthening the real system without any disruption to the data center or actual system performance. In that regard, an important aspect of honeypots is complete isolation from a real, vulnerable source of data or access point via the fake honeypot.
In this low interaction honeypot project, the goal is for it to be deployed in a controlled environment where pretend attacker hosts are run by the project developers during a simulation. In order to do that, we chose Wireshark for network analysis so we could go a step further in the simulation and have network traffic proof of what activity was flagged as malicious, not flagged at all, or deemed suspicious. In order to stay within a semester, scope the website was designed as a face-value website with only the log-in home screen available for view. The website was designed around an older version of a honeypot which would have been deployed in the 2000s, so the website has been modeled after images from 2000s social media websites. These websites were often a target of those attacks since they often used HTTP instead of HTTPS. In a larger scope, this project would have enacted a fuller-built website with additional recreations of a typical 2000s social website including clickable links, additional text boxes, and fake imitations of user blog posts or media sharing. By designing a honeypot that would have been enacted in the 2000s we are able to efficiently display why the http design was weak for authentication purposes and otherwise gain insight on how the basis of these attacks by simulated attackers might still be relevant in https attacks today. The reason for the honeypot being completely isolated and simulated both in user information and the attacker role is to prevent any real exposure to the developer's networks and computer systems in the case that the honeypot is attacked by truly malicious parties that surpass a student's skills in designing a fully isolated honeypot.

.

II. System Architecture

For this project the honeypot was built to ensure proof of concept. The honeypot we created here is created through a .html file that holds the contents of the webpage to start off with. Then with Python we're able to allocate a server to that specific webpage so whenever the IP and port number are input, we get a webpage showing a login and password. Through this log in and password we built a few more .html files. These files include Accessdenied.html and Accessaccepted.html. For these access pages we included low level information about the site and what just happened, if you had happened to log in with the wrong username/password, you'd be taken to the access denied page in which you can try again to log in. If you log in correctly and have the same IP and MAC as the whitelisted ones within the csv, then you'll be accepted. While this is all going on, the server is documenting all activity within the pages that gets logged in a .txt page which shows user activity. With all of this, the Server VM starts the HTTP server, then the Victim VM logs into the Server through its IP and port number and then during the demonstration the Attacker VM starts to initiate its process. The Attacker VM then takes a python script that includes Scapy libraries and beings to ARP spoof the Victim VM [3]. This makes it so that when the attack is initiated, all packets thrown to the Victim are given to the Attacker VM instead so that when Victim begins to login, the login information is shown through Wireshark [1] on the Attackers side in which it uses the PORT constraint to see the login username and password. Since the Server has a whitelisted IP and MAC address, it knows that the new login from a different address is suspicious in which it only allows the Attacker to get as far as we want it to. So, in this case we allowed the Attacker to get through the login page but once it tried to access more than just that, we flagged it for suspicious activity. Although we flagged it early in this honeypot, we could have set up more fake information that looks true but isn't so the attacker would think the attack worked.

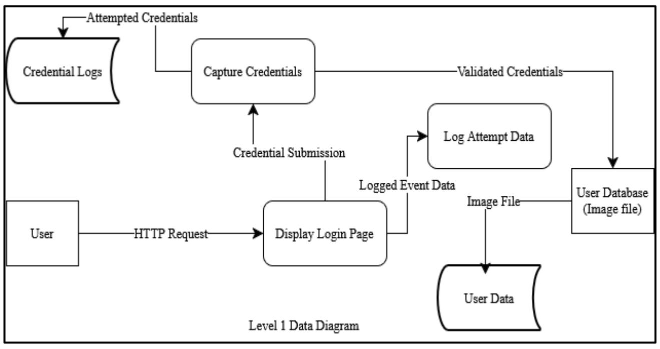

Figure 1: Data Flow Diagram level 1

A. DFD Level 1 Elements

1. Entities
    a. User: represents both victim host and attacker host
    b. User Database (image file)
2. Processes
    a. Capture Credentials: process the credentials prompted from user
    b. Log Attempt Data: adds to log file per log-in attempt
    c. Display Login Page: displays default log-in page with user and password prompts
3. Data Stores
    a. Credential Logs: log file of valid (fabricated for simulation) credentials
    b. User Data: file containing vulnerable data per user (Image file for simulation)
4. Data Flows
    a. HTTP Request
    b. Credential Submission
    c. Logged Event Data
    d. Image File
    e. Attempted Credentials
    f. Validated Credentials

III. Discussion

The results of this project were conclusive in demonstrating the effectiveness of a low-interaction honeypot specifically designed for network traffic monitoring within a controlled environment. The simulated attacks were successful in displaying one particular way an attacker would interact and receive exposure from the fake website system. By intentionally designing using a security system used in the 2000s and otherwise rendered obsolete modern cyber security standards, such as unencrypted http login forms, the honeypot effectively captured the specific traffic that would otherwise remain covert. These particular observations are validating for the education value these types of simulations can demonstrate within the legacy-styled web systems, specifically on how intrusion techniques operate and how sensitive data is exposed within the network.

IV. Conclusion

This project met its goals in successfully demonstrating how a low-interaction honeypot can be used for network traffic monitoring with the aim of analyzing logs of simulated unauthorized access attempts in a controlled environment. The choice of designing the project with obsolete website http designs of 2000s era social media blogs, which use unencrypted communication, the honeypot made the simulated attacks identifiable and able to be observed via that setup for the network traffic sniffing. The authentication process allowed for a hands-on experience in gaining educational value for understanding why these designs were rendered obsolete and how they have impacted modern social web designs. The simple structure of our system ensured developer's computer systems no risk while still allowing intruder behavior to be simulated during testing. Despite goals being achieved and educational value being met, there are still some improvements that would be implemented in future versions of the same project. To stay within a scope that uses obsolete 2000s-level web security, the website could further it's depth and realistic appeal by having more links, text, and additional imitated user accounts. Additionally, the type of attack used within this simulation could not only be improved but multiple types could be carried out. In some versions, the usage of AI models for an attacker's behavior might be beneficial for furthermore exhaustive testing for the effectiveness of the honeypot website's security. Suggested simulated attacks for future implementation would be SQL injections, credential brute forces, URL tampering (which would be most effective if hyperlinks and buttons are included in that honeypot design), or cookie manipulation. Other studies and simulations similar to this project have been conducted with much more in-depth results upon experimenting with similar obsolete designs and more than a single type of attack at once [5].

".

Acknowledgment

Thank you to Professor Kandel for her support and assistance in the project and to the T.A. Kappers, H. for his feedback on design.

REFERENCES

[1] Wireshark Foundation, "Wireshark: Network Protocol Analyzer," 2025. [Online]. Available: https://www.wireshark.org/

[2] Proofpoint, "Honeypots: What They Are & How They Work," Proofpoint Threat Reference, [Online]. Available: https://www.proofpoint.com/us/threat-reference/honeypot.

[3] SEED Labs, "ARP Cache Poisoning Attack Lab," 2024. [Online]. Available:https://seedsecuritylabs.org/Labs_20.04/Networking/ARP_Attack/ (Accessed: Dec. 9, 2025).

[4] Akamai, "High-Interaction Honeypot Versus Low-Interaction Honeypot — Comparison," Akamai Blog, Jan. 3, 2019. [Online]. Available: https://www.akamai.com/blog/security/high-interaction-honeypot-versus-low-interaction-honeypot-comparison (accessed Dec. 9, 2025).

[5] Z. Morić et al., "Advancing Cybersecurity with Honeypots and Deception Strategies," Information, vol. 12, no. 1, 2025. [Online]. Available: https://www.mdpi.com/2227-9709/12/1/14