**PROJECT PROPOSAL**

*HoneyWeb*

Table of Contents

[Project Abstract 3](#_Toc113300636)

[Conceptual Design 3](#_Toc113300637)

[Proof of Concept 3](#_Toc113300638)

[Background 3](#_Toc113300639)

[Required Resources 3](#_Toc113300640)

## Project Abstract

This document proposes a Honeypotting software that will recursively extend itself, creating a “web” of Pots and Tokens to trap an attack. The idea is essentially forcing an attacker to “Dig in the wrong place”. When an attack engages with a honeypot, it will spawn a new honeypot with unique characteristics and honeytokens to deceive the attacker. This will continue for as long as the attack continues. It will provide ample time to track an IP and port, as well as learn about the attack. If the attacker disconnects from the honeypot, it will spawn new honeypots with unique tokens to continue the attempt to deter the attack. The HoneyWeb is not a security system like a firewall, it is designed to observe attacks and take full advantage of the honeypot's capability to confuse and slow down attacks.

## 

## Conceptual Design

Language: Python

Libraries: Flask,Honeypot, threading, Random,time

Structure

## Proof of Concept

https://github.com/JoeStellaTU/HoneyWeb

## Background

<https://github.com/paralax/awesome-honeypots>

<https://github.com/honeytrap/honeytrap>

## Required Resources

### Hardware Resources:

1. **Server/Host Machine:**
   * Depending on the scale and complexity of your honeypot network, you may need a dedicated server or host machine to deploy and manage the honeypot instances. This could be a physical machine or a virtual server.
2. **Network Infrastructure:**
   * Ensure access to a network infrastructure where you can deploy and monitor the honeypots. This may involve setting up a local lab environment or collaborating with a network administrator to simulate realistic network conditions.

### Software Resources:

1. **Operating System:**
   * Choose a suitable operating system for your honeypot environment. Linux distributions like Ubuntu Server or specialized honeypot-focused OS such as Honeyd or Modern Honey Network (MHN) are common choices.
2. **Honeypot Software:**
   * Select and install honeypot software. Examples include Honeyd, Cowrie (SSH honeypot), Dionaea (malware honeypot), or Glastopf (web application honeypot). Ensure compatibility with your chosen operating system.
3. **Event Handling Library (Optional):**
   * If you opt for an event-driven architecture (using Python's **threading.Event**), ensure your chosen programming language and environment support it.
4. **Security Tools:**
   * Install security tools for monitoring and analysis. This may include intrusion detection systems (IDS) like Snort or Suricata, packet capture tools like Wireshark, and log analysis tools.
5. **Research Datasets (Optional):**
   * If your project involves training models or conducting data-driven analysis, you may need datasets related to cybersecurity threats. Sources like the National Vulnerability Database (NVD) or datasets provided by organizations like Kaggle could be valuable.
6. **Documentation and Reporting Tools:**
   * Use tools for documenting your project, such as Markdown for documentation and Jupyter Notebooks for interactive analysis. Reporting tools may include graphing libraries (Matplotlib, Seaborn) for visualizing results.
7. **Version Control System:**
   * Utilize a version control system (e.g., Git) for tracking changes in your codebase and collaborating with team members.