Project Report

Implementing SSH HoneyPot to Study Potential Attackers for Network Security

Sudhanshu Mukherjee
Shubham Singh
Vinil Harsh

Introduction to the SSH Honeypot

A security tool known as an SSH honeypot is made to identify, thwart, or investigate efforts to obtain unauthorized access to information systems via the Secure Shell (SSH) protocol. Secure SSH access is essential to preventing unwanted access and potential security breaches since SSH is frequently used for secure remote system administration. It makes it possible to examine malevolent behavior in a regulated setting. Our goal is to construct a realistic SSH honeypot that will draw in potential attackers and allow us to examine their techniques.

The question arises why we have chosen to build SSH Honeypot

The major reason is to build that it identifies and monitor potential threats and attack patterns. After the whole analysis it comes to an end that after doping so, we can study the behavior of the attackers and will be able to understand the emerging threats.

The SSH Honeypot can benefits in many ways and some of them are:

1. **Detection of Attackers:** SSH honeypots are used to detect and draw in malevolent parties looking to enter a system without authorization. The honeypot attracts attackers by mimicking a weak or alluring SSH server, which enables security experts to research their strategies.

2. **Able to understand the Attack Patterns:** SSH honeypots assist security professionals in examining the methods and approaches used by attackers. This knowledge is useful for creating stronger defenses and enhancing security protocols.

3. **Policy Enforcement:** SSH honeypots detect and stop unwanted access attempts, which can assist enterprises in enforcing security standards. This is especially crucial for businesses with stringent access control procedures in place.

4. **Legal and Ethical Research:** SSH honeypots can be used by organizations and security researchers for morally and legally sound research. Staying ahead of developing threats requires an understanding of hostile actors' methods.

Talking about what are the functionalities of SSH Honeypot

The functionalities of the SSH Honeypot are divided into two sub categories:

SSH Honeypot Script

1. **Emulates an SSH server to attract potential attackers.**
   When an SSH honeypot "emulates an SSH server to attract potential attackers," it means that the honeypot is designed to mimic the behaviour of a legitimate SSH server to entice and interact with individuals or automated tools attempting to gain unauthorized access.

2. **Monitors and logs commands sent by attackers.**
   It indicates that its purpose is to record and intercept any activities made by possible bad actors who try to access the system without authorization by using the SSH protocol.

3. **Detects predefined honeytokens to identify malicious activity.**
   It is intended to deceive adversaries into interacting with purposefully inserted false information. After then, the honeypot examines these exchanges to spot and report any possibly dangerous activities. By taking a proactive stance, security experts can better recognize and counter possible security threats by gaining insight into the strategies and techniques used by attackers.

Flask Web Interface

1. **Establishes communication with the honeypot script.**
   The purpose of the Flask web interface is to communicate with the honeypot script. Through this communication, the web interface can instruct the honeypot script to change its settings or transmit commands, as well as receive updates or information from the script.

2. **Allows users to interact with the honeypot through simulated connections.**
   The Flask web interface enables users, such as security professionals or system administrators, to interact with the SSH honeypot. This interaction may include tasks such as monitoring the activity within the honeypot, viewing logs, analyzing attempted connections, and adjusting configuration settings. The term "simulated connections" describes the construction of fictitious or dummy exchanges that imitate genuine SSH connections. These simulated connections are intended to mimic the actions of authentic users or possible attackers trying to gain access to the system within the framework of an SSH honeypot.

3. **Provides a user-friendly interface for monitoring real-time logs.**
   The Flask web interface provides a visually intuitive platform for monitoring real-time logs, which improves the usefulness of the SSH honeypot. This enhances the

effectiveness and ease of use for security experts who are responsible for examining and addressing behaviours in the honeypot setting.

The implementation of Frontend HTML

The technology used to implement the frontend of the SSH Honeypot are HTML, CSS, and JavaScript.
The process of designing a user interface that closely resembles the look and feel of a computer terminal. Terminals are text-based user interfaces that are frequently used for command-line operations, system administration, and server interaction.

In frontend development, user interaction refers to giving users a fun and easy-to-use interface when they interact with the program. This covers operations such as interacting with the UI, submitting forms, and clicking buttons.

In frontend development, responsive layout refers to creating an interface that can adjust to different screen sizes and devices. This guarantees a unified and intuitive user experience on tablets, smartphones, and PCs.

The implementation of SSH Honeypot Script

The SSH Honeypot build in such a way that it is user-friendly and the development of interface is sober and simple to understand and one can operate easily.

Utilizes Python Threading for concurrent execution.
Parallelism in a program is achieved by concurrent execution using Python threading. A program's various components can operate independently and concurrently thanks to threading. The threading module in Python offers a method for managing and creating threads.
Some of the major reasons why we used python threading:

- **Responsive User Interface:** Threading is frequently used in graphical user interface (GUI) programs to maintain the interface's responsiveness while carrying out background operations. The main thread is kept free to process user input and update the GUI by dividing time-consuming tasks into different threads, which keeps the application from becoming sluggish.

- **Concurrency in I/O-Bound Tasks:** For I/O-bound operations, where a large amount of time is spent waiting for input or output procedures to finish, threading works effectively. Because several threads can execute I/O tasks simultaneously, the software can utilize its resources more effectively.

- **Asynchronous Programming:** In asynchronous programming, threading is frequently used to control concurrent execution without the requirement for explicit parallelism. Threading is used by asynchronous frameworks like asyncio to provide concurrency without using blocking calls.

Log rotation ensures effective log management.
Effective log management in SSH honeypots and other systems requires log rotation. It entails the archiving, compression, and deletion of log files on a regular basis to guarantee that historical data is kept for analysis and that the logs don't take up too much disk space. A vital component of log management in SSH honeypots is log rotation, which reduces the possibility of attacker-caused log modification while preserving system health, performance, and historical data.

Implementation of Flask Web Interface

Flask Routes for various functionalities: By designing routes that correlate to various capabilities or aspects inside the application, the goal is to organize the Flask application. Every route has a unique URL endpoint that it is connected to, and when a user visits that endpoint, the functionality that goes along with it is activated.

Separating Simulated Connections and Real-time Log Retrieval: Using this method, routes are arranged according to whether they manage real-time log retrieval or simulated connections, or interactions with the honeypot. When it comes to simulated connections, this could involve logging in to the honeypot using SSH and retrieving logs in real-time.

Implementation of Real-Time Log Updates

JavaScript and AJAX (Asynchronous JavaScript and XML) are used in web development to accomplish dynamic updates in real-time, which allows for log updates to be completed without the need for a page refresh. This approach makes use of asynchronous data fetching and updating technologies to create an immersive experience, enabling users to watch actions quickly. In order to ensure effective resource usage and minimal latency in real-time updates, optimization strategies are essential for seamless functioning. By giving consumers quick feedback and enabling them to stay updated about ongoing activities without the need for manual involvement, this improves the user experience. Moreover, these streamlined updates support effective resource usage, lower server load, and scalability even with an increasing user base. Thus, the combination of optimization techniques, AJAX, and JavaScript creates a responsive and scalable online environment, making more beneficial for an user’s engagement and operational efficiency.

Main aspects related to Security Measures

Security Deployment: Install the SSH honeypot in a secured area with limited access. To facilitate monitoring, keep links to reliable sources to a minimum. Utilize network segmentation and firewalls to separate the honeypot and stop illegal access to other vital systems. This containment technique aids in reducing any possible risks related to the installation of honeypots.

Regular Updates: To handle new security risks, make sure the dependencies and honeypot script are updated on a regular basis. Use the most recent security updates and patches to

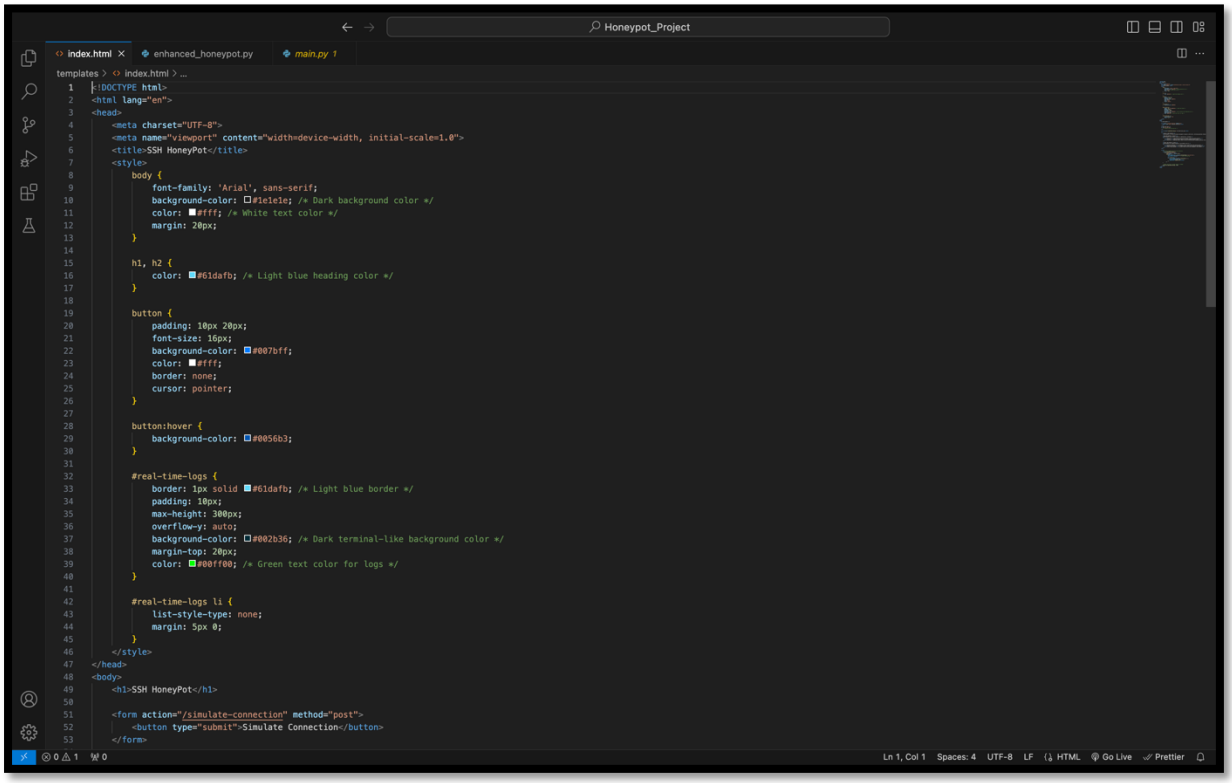
get rid of vulnerabilities. Frequent upkeep improves the honeypot environment's overall security posture and lowers the danger of exploitation.

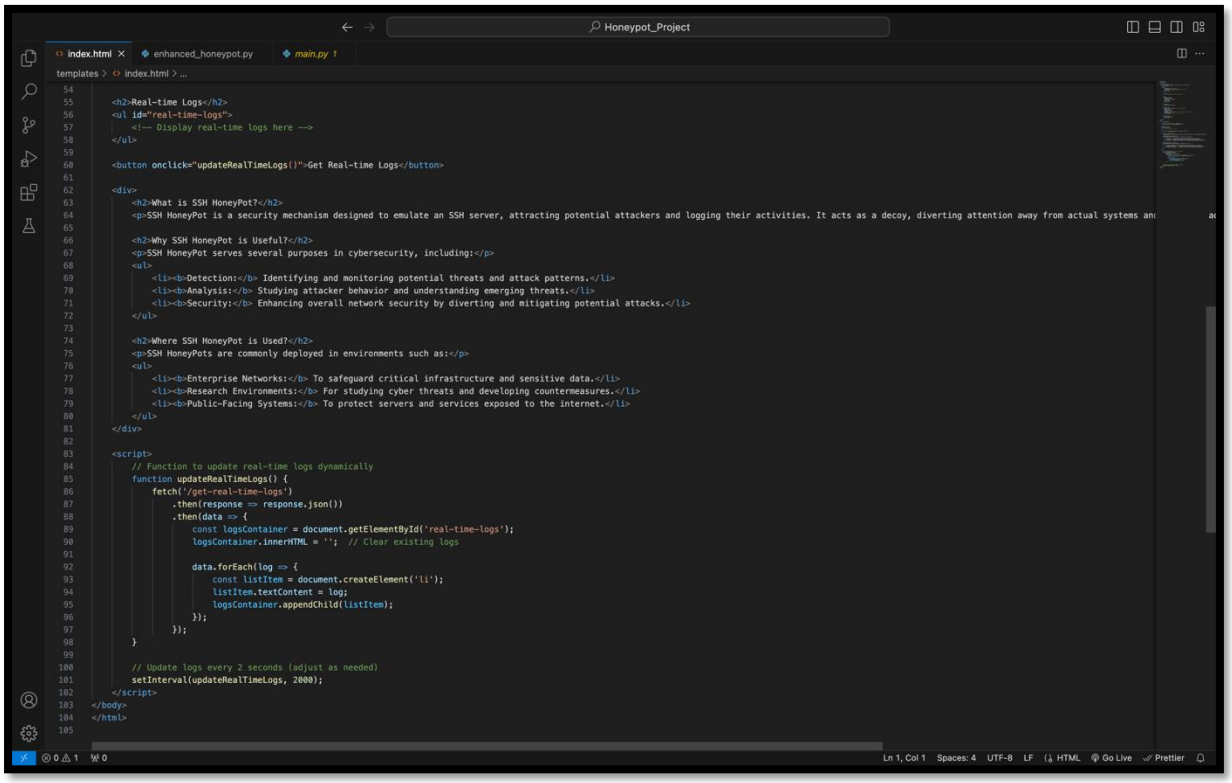
Logging and Analysis: Analyze honeypot logs on a regular basis to spot trends and possible dangers. Examine log data to learn about the tactics and actions of attackers. To improve threat detection efficiency, use automated analytical techniques like machine learning algorithms or intrusion detection systems (IDS). Promptly address detected patterns in order to fortify defenses and avert possible security breaches.

Integration with Security System: To take use of the advantages of an all-encompassing security approach, integrate the honeypot into the current security infrastructure. Provide data to security monitoring systems so they can do a thorough investigation. By ensuring that the honeypot's findings are included into a larger security plan, this integration makes it possible to mount a more effective defense against changing threats.

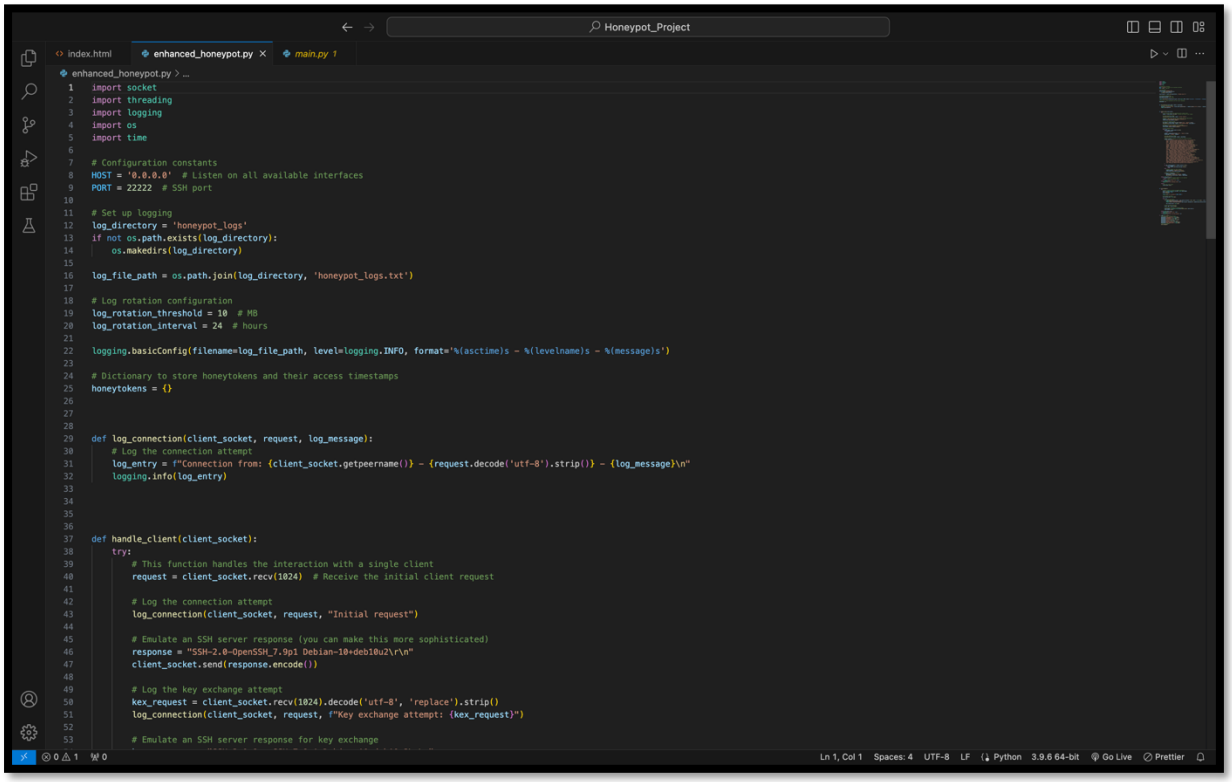
Snippets of the Code

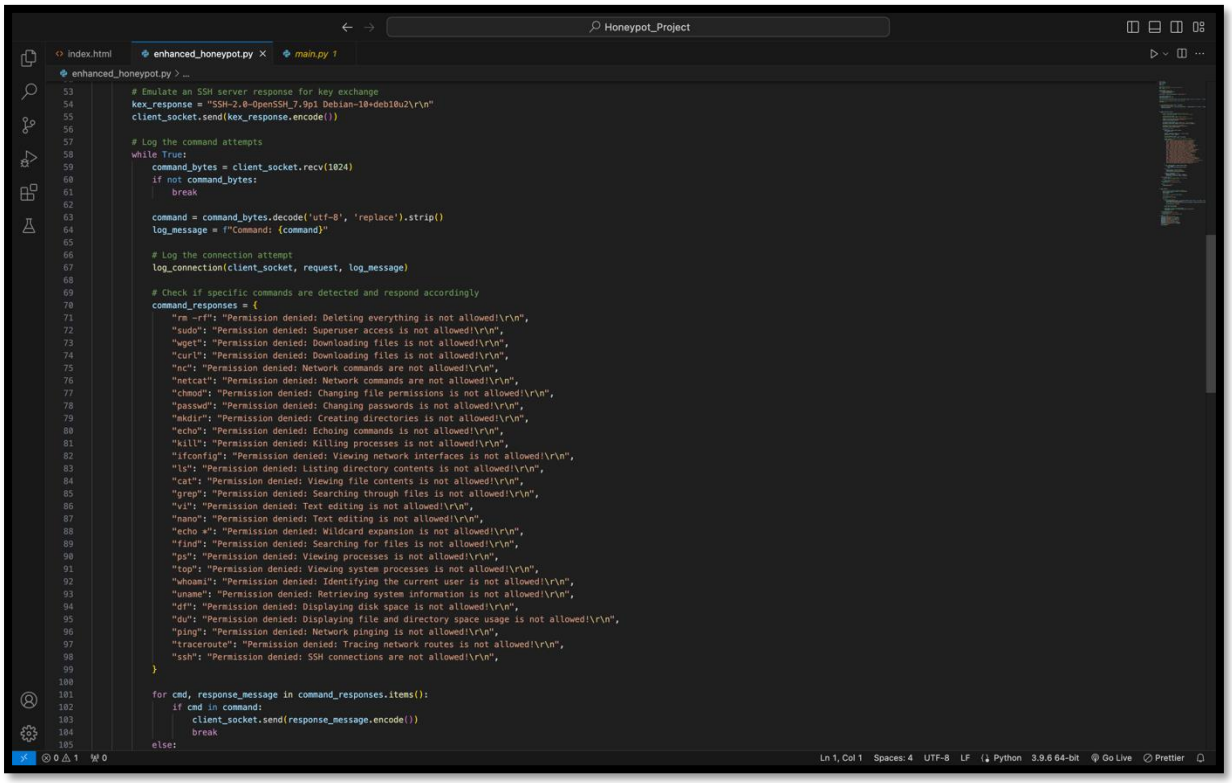
HTML file for the frontend development:

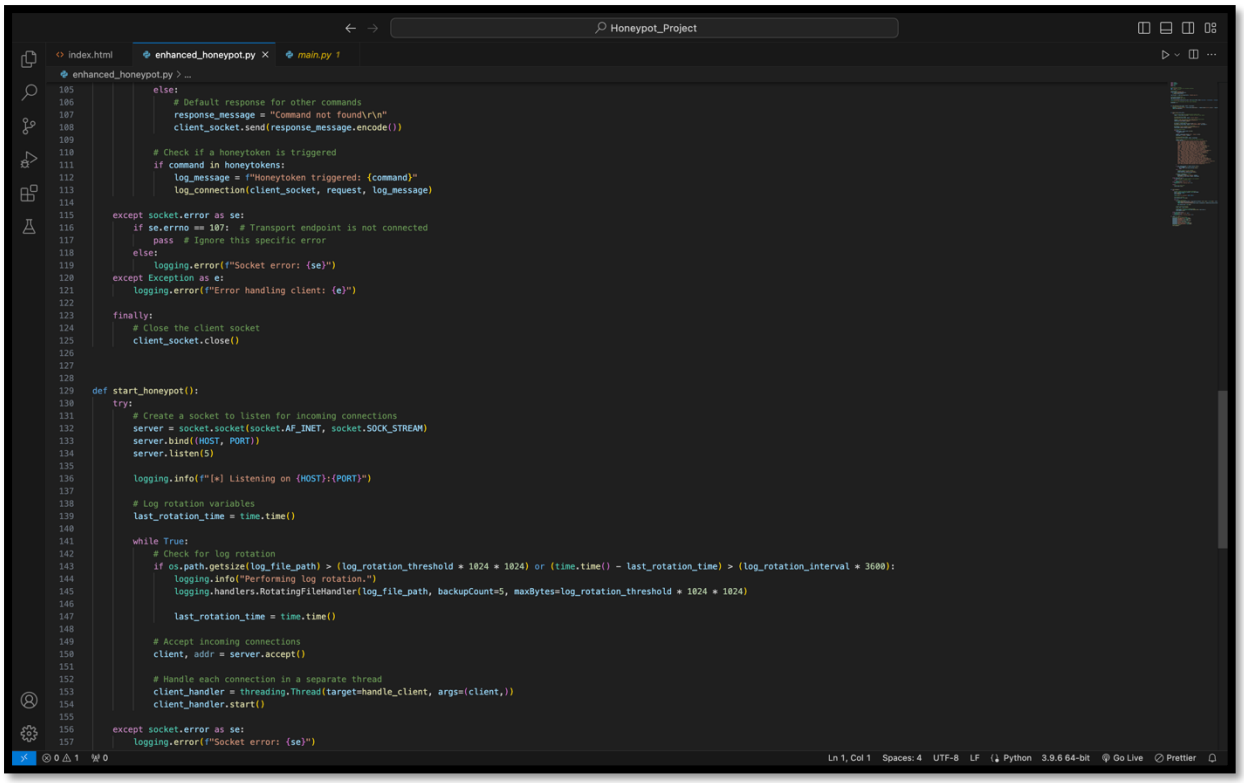




Python file with enhanced honeypot functionality:



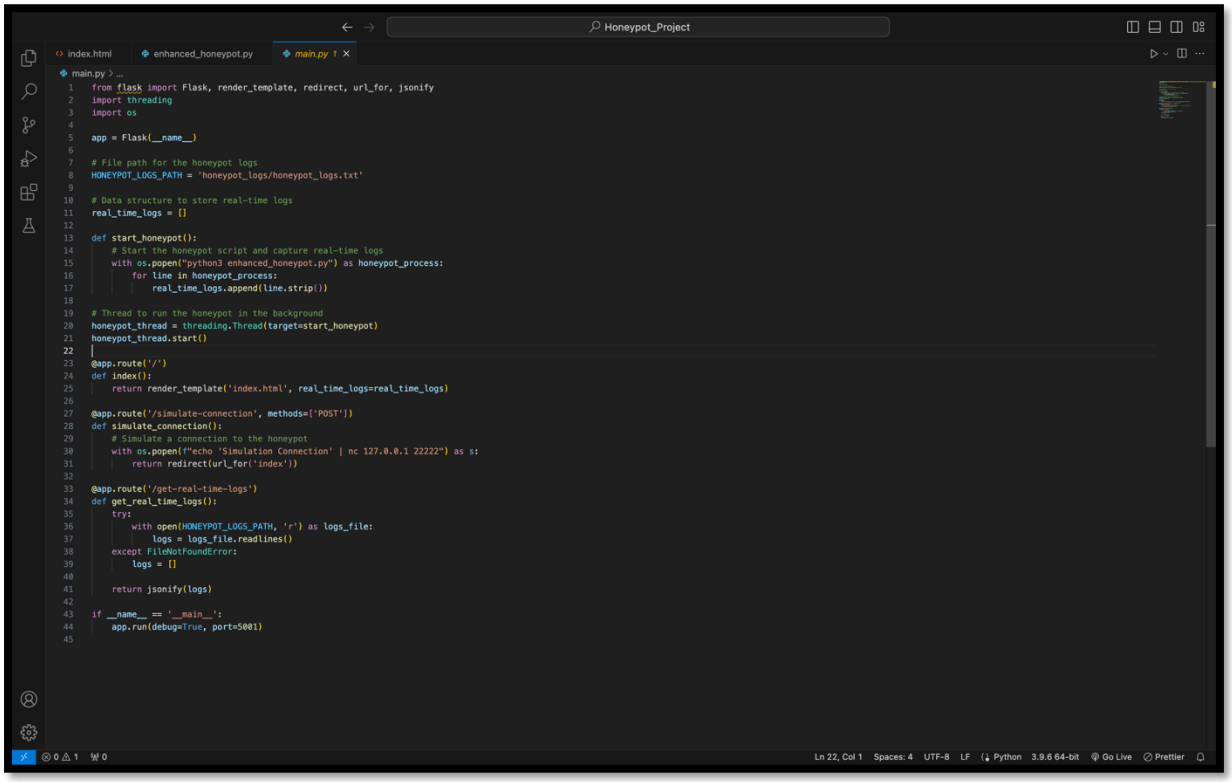




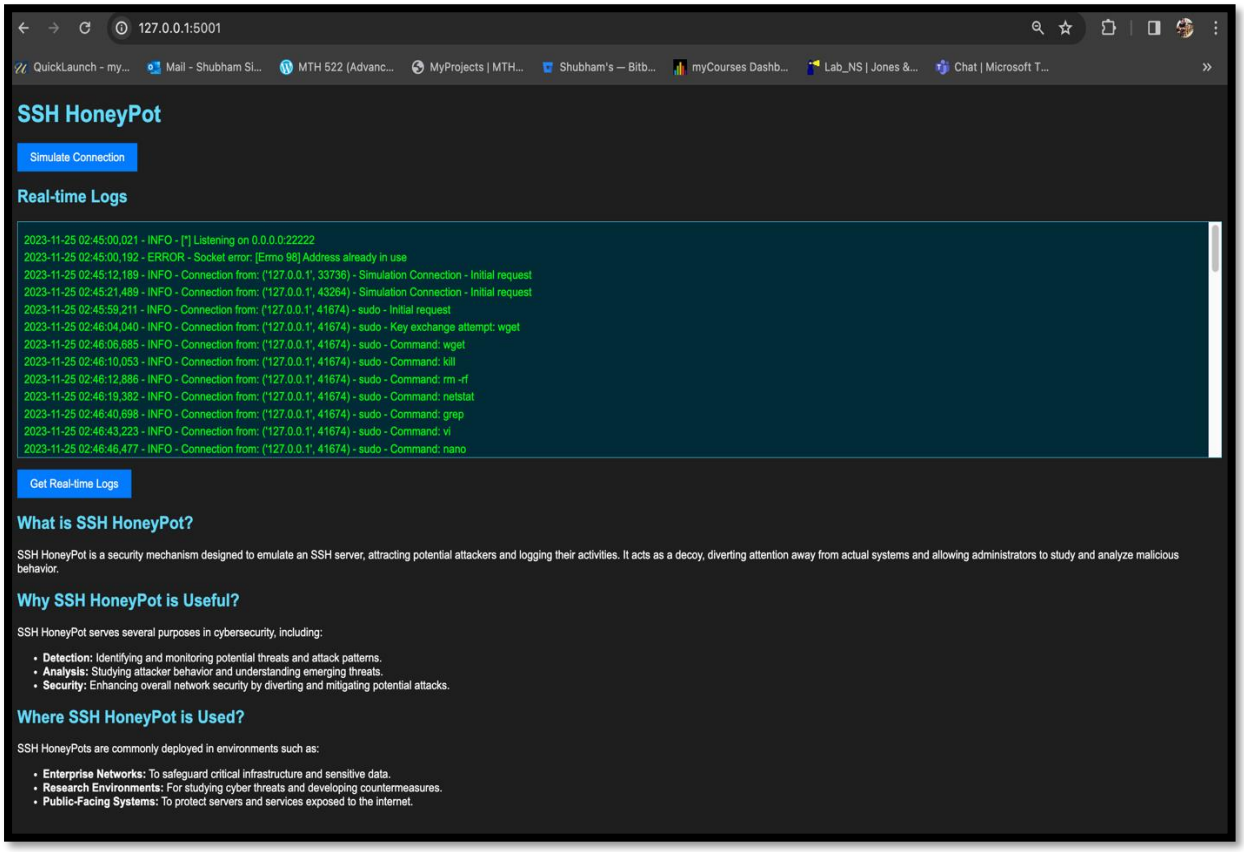

```
    except socket.error as se:
        logging.error(f"Socket error: {se}")
    except Exception as e:
        logging.error(f"Error starting honeypot: {e}")

if __name__ == '__main__':
    # Add honeytokens to the dictionary
    honeytokens["fake_password"] = time.time()
    honeytokens["admin@admin.com"] = time.time()
    honeytokens["hacker123"] = time.time()
    honeytokens["secretpassword"] =  time.time()
    honeytokens["root"] = time.time()
    honeytokens["backdoor_key"] = time.time()
    honeytokens["malicious_payload"] = time.time()
    honeytokens["fake_credit_card"] = time.time(),
    # Start the honeypot
    start_honeypot()
```

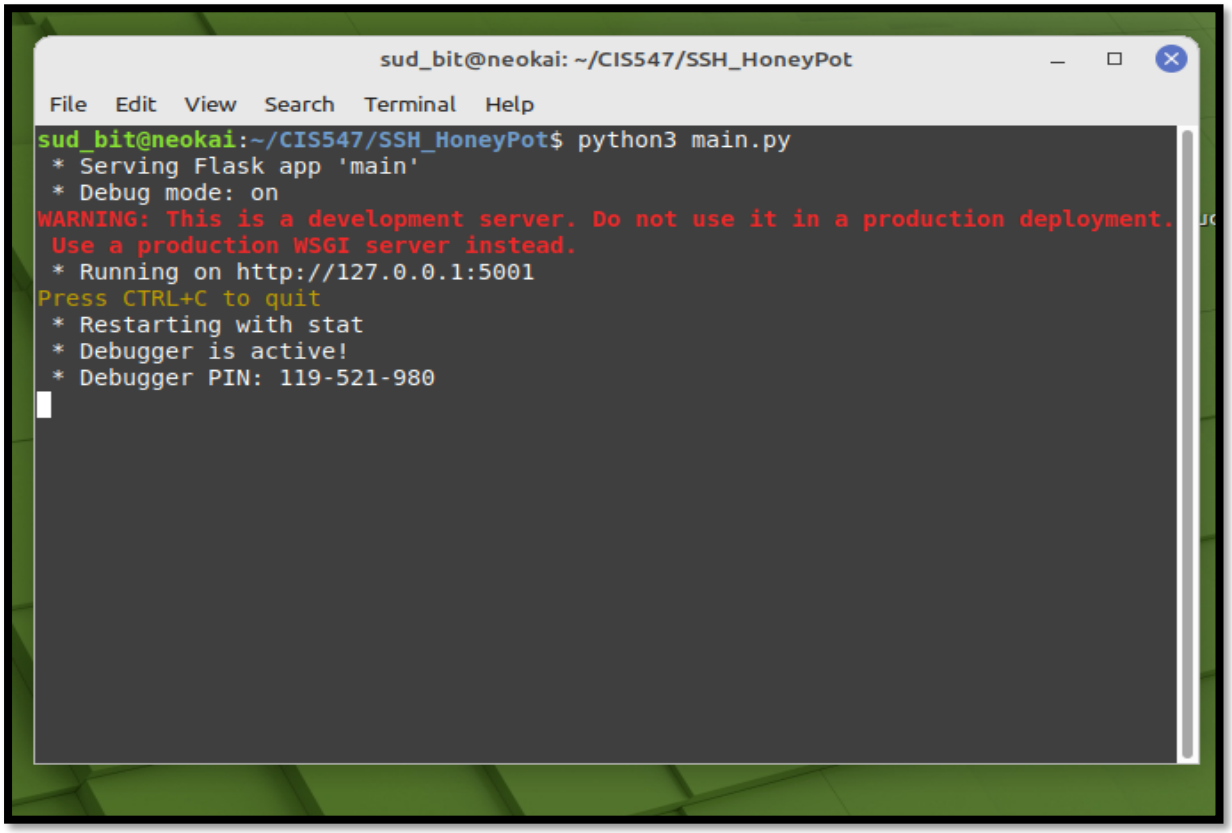
Main python file in which threading and flask are built for the web interface:

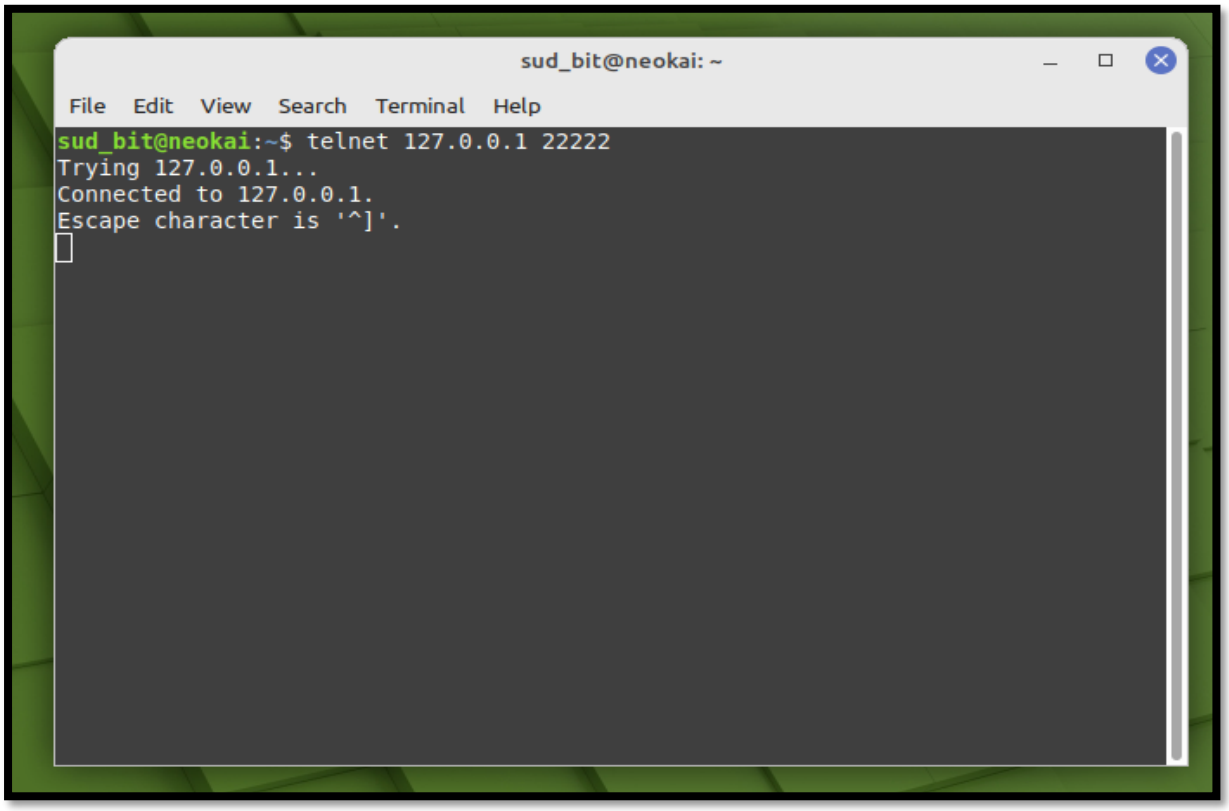


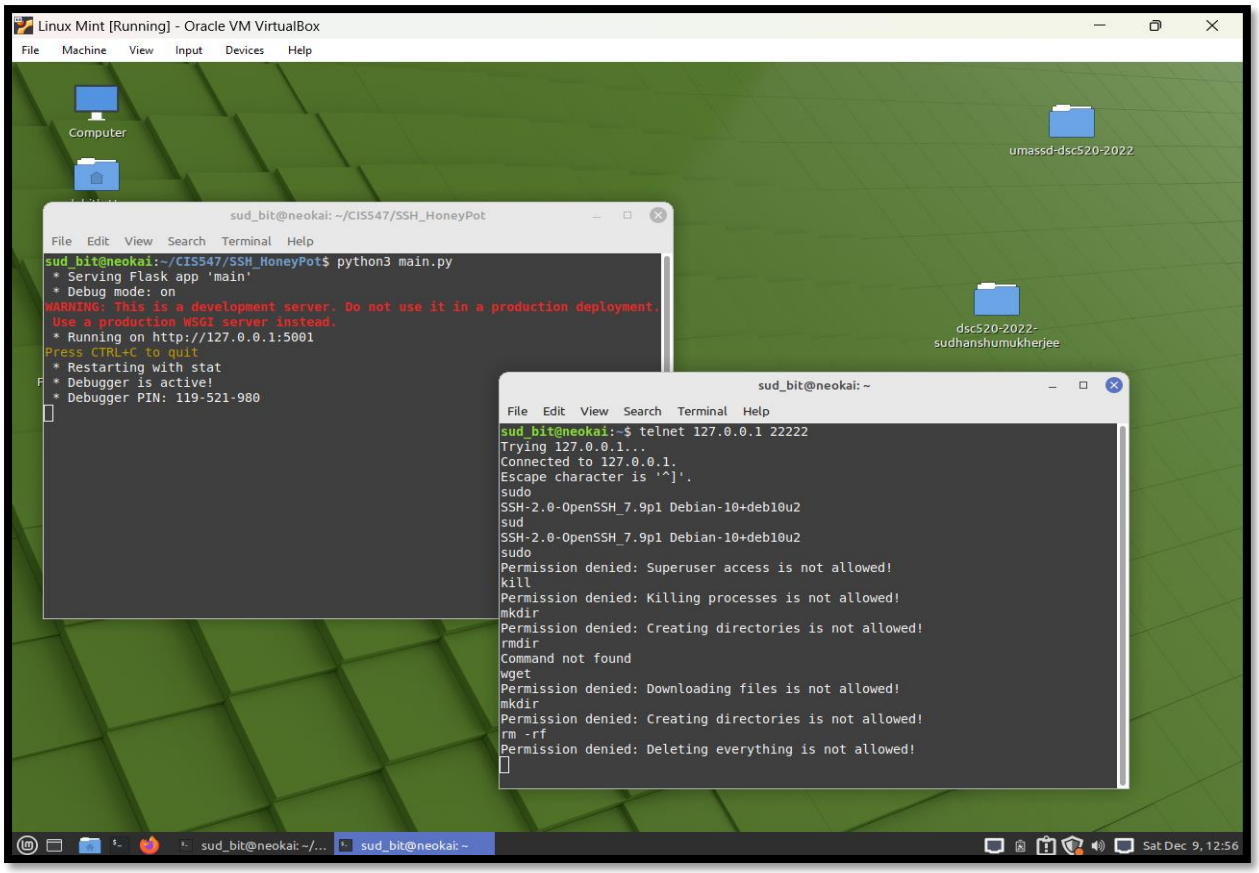
Snippets of an User Interface



Snippets for an Output







Conclusion

In the realm of cybersecurity, understanding the tactics employed by potential attackers is crucial for fortifying our defenses. Our SSH HoneyPot project has been a journey into the world of deception and detection, offering a proactive approach to identifying and thwarting malicious activities.

Through the deployment of our SSH HoneyPot, we've created an environment that entices would-be attackers with simulated vulnerabilities and false assets. By closely monitoring the interactions with our decoy system, we gain valuable insights into the methods and intentions of malicious actors. The project's success lies in its ability to act as an early warning system, allowing us to detect, analyze, and respond to security threats before they can escalate.

The key features of our SSH HoneyPot include a dynamic response mechanism to various commands, a robust logging and analysis system, and the integration of HoneyTokens for an added layer of deception. The enhanced honeypot not only logs connection attempts but also responds strategically to specific commands, limiting the potential actions of attackers and providing valuable intelligence on their intent.

By incorporating elements like HoneyTokens, we've introduced a level of sophistication to our deception strategy. These fake pieces of information, strategically placed within our honeypot,

serve as tripwires, alerting us to unauthorized access attempts and potential security breaches. The project's success is not only measured by the incidents it detects but also by the deterrent effect it imposes on attackers, who may be hesitant to distinguish between real and decoy systems.

As we conclude our SSH HoneyPot project, we recognize that the cybersecurity landscape is dynamic and ever-evolving. Our honeypot serves as a testament to the importance of proactive defense measures, leveraging deception as a powerful tool to stay one step ahead of adversaries. By continuously refining and expanding our honeypot's capabilities, we contribute to the collective effort of securing digital landscapes and fostering a safer online environment.

Our SSH HoneyPot project stands as a foundation for future endeavors in understanding, mitigating, and adapting to the evolving threat landscape.