WinnieThePooh Honeypot Project Report

1. Introduction

The **WinnieThePooh Honeypot** is a cybersecurity project designed to simulate vulnerable services (SSH & HTTP) in order to **attract, log, and analyze malicious activity**.

It provides:

- A **fake SSH service** (using Paramiko) that records attacker login attempts and keystrokes.
- A **fake HTTP service** (using Flask) to log web-based scans or exploit attempts.
- A **database backend** for storing logs of attacks.
- A **dashboard** for visualizing attack data.
- A **brute-force simulator** to test honeypot resilience.

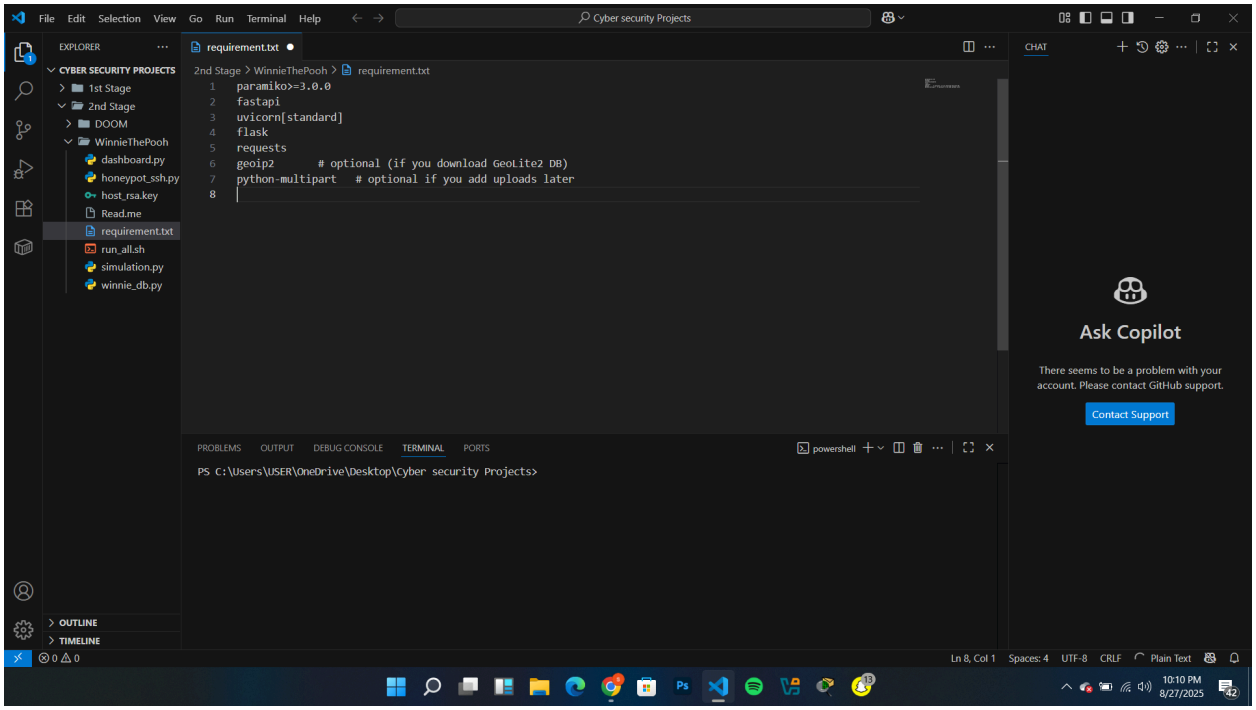
By deploying this project, defenders can learn how attackers behave, test intrusion detection methods, and raise awareness of password brute-forcing threats.

2. Project Requirements

The following dependencies were installed via `requirements.txt`:

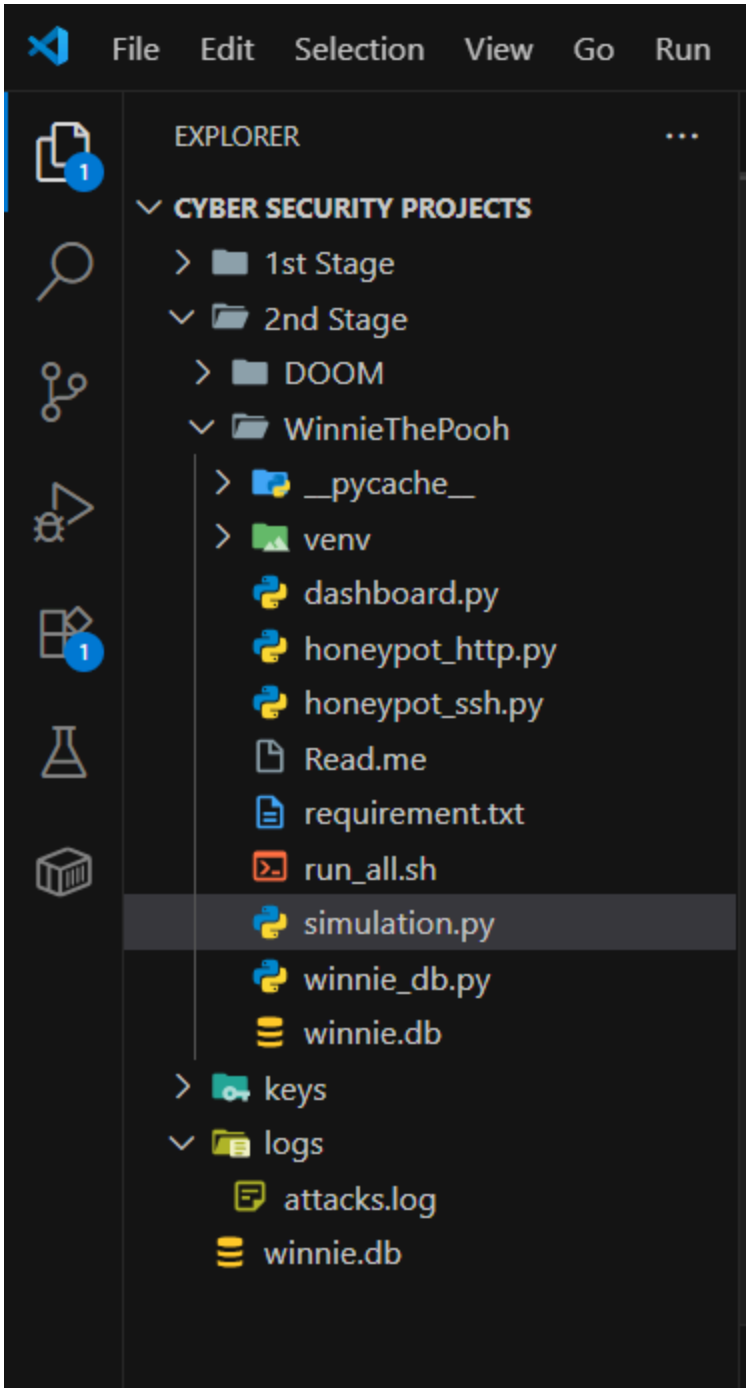
```
paramiko>=3.0.0
fastapi
uvicorn[standard]
flask
requests
geoip2                # optional (for geolocation of IPs)
python-multipart      # optional (for uploads in HTTP honeypot)
```

screenshot of requirements.txt in VS Code



3. Project Structure

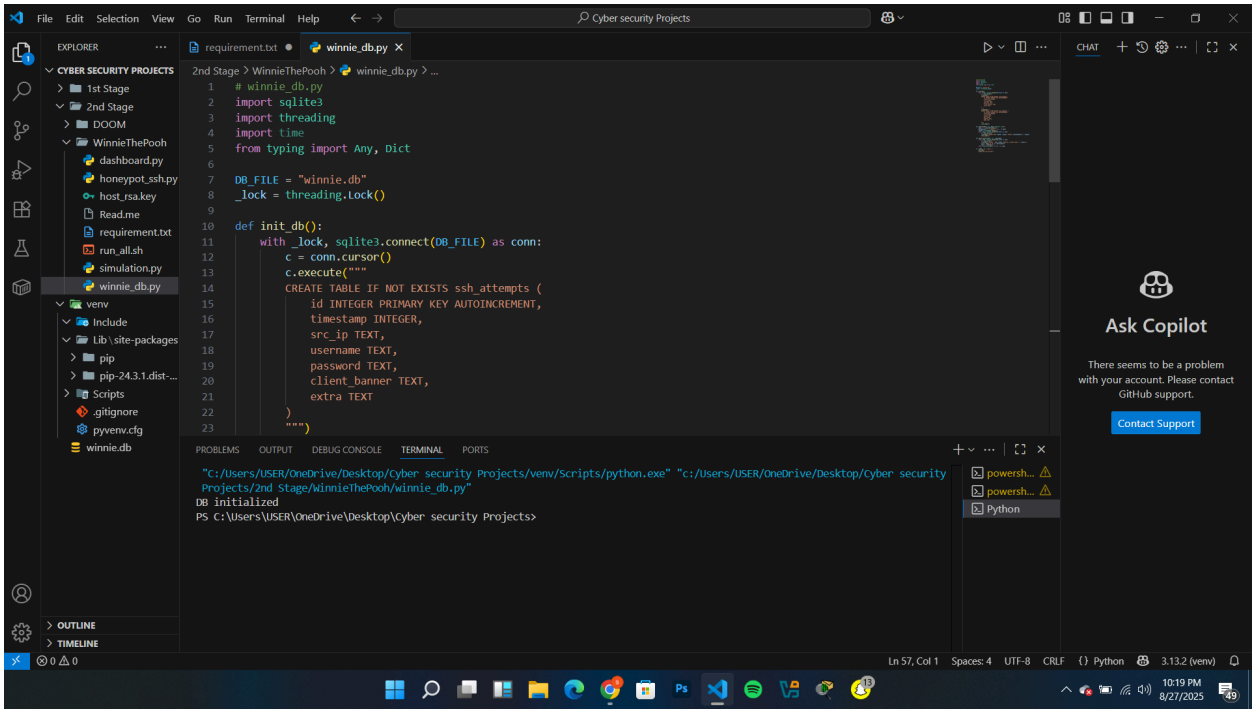
```
winniethepooh/
├─ README.md
├─ requirements.txt
├─ host_rsa.key              # SSH private key for honeypot
├─ winnie_db.py              # Database logic (attack logs)
├─ honeypot_ssh.py           # SSH honeypot service
├─ honeypot_http.py          # HTTP honeypot service
├─ dashboard.py              # Flask dashboard for logs
├─ simulate_ssh_bruteforce.py # Script to simulate attacks
└─ run_all.sh                # Script to launch all components
```


File
Edit
Selection
View
Go
Run
EXPLORER
CYBER SECURITY PROJECTS
1st Stage
2nd Stage
DOOM
WinnieThePooh
__pycache__
venv
dashboard.py
honeypot_http.py
honeypot_ssh.py
Read.me
requirement.txt
run_all.sh
simulation.py
winnie_db.py
winnie.db
keys
logs
attacks.log
winnie.db

4. Steps Taken

Step 1: Database Setup

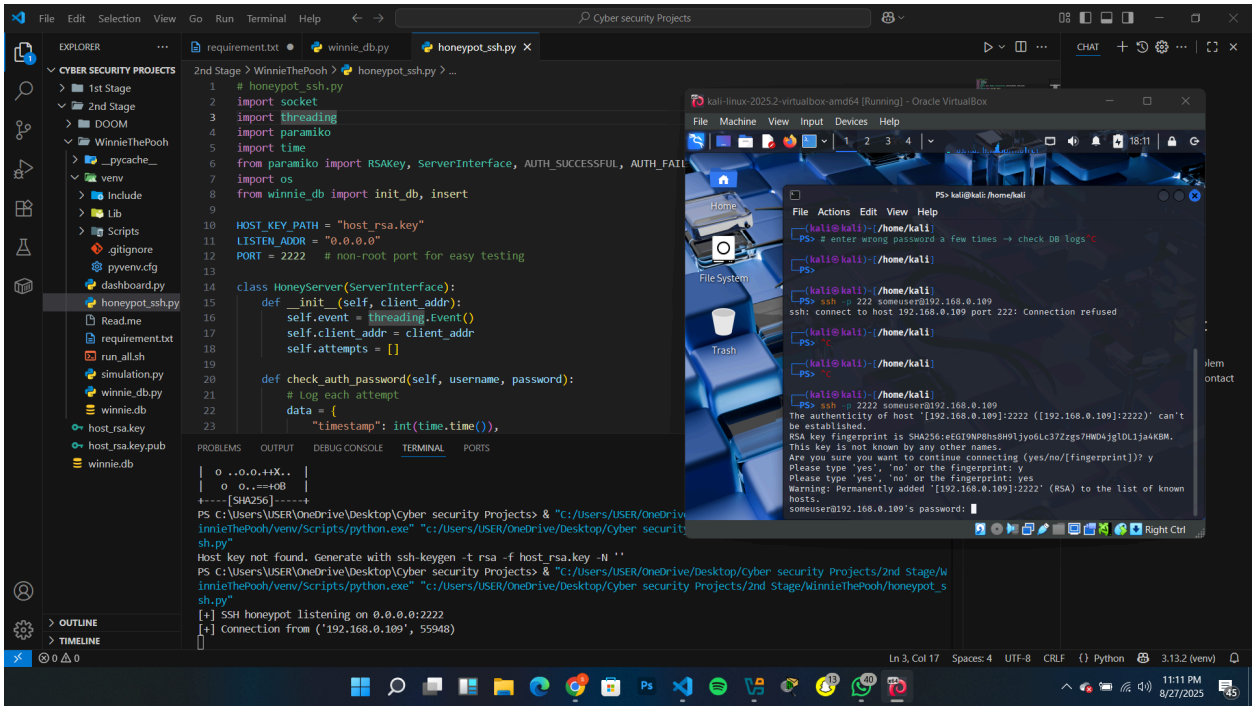
- Created `winnie_db.py` to initialize and manage the database.
- The DB stores attack attempts, IP addresses, timestamps, and captured data.

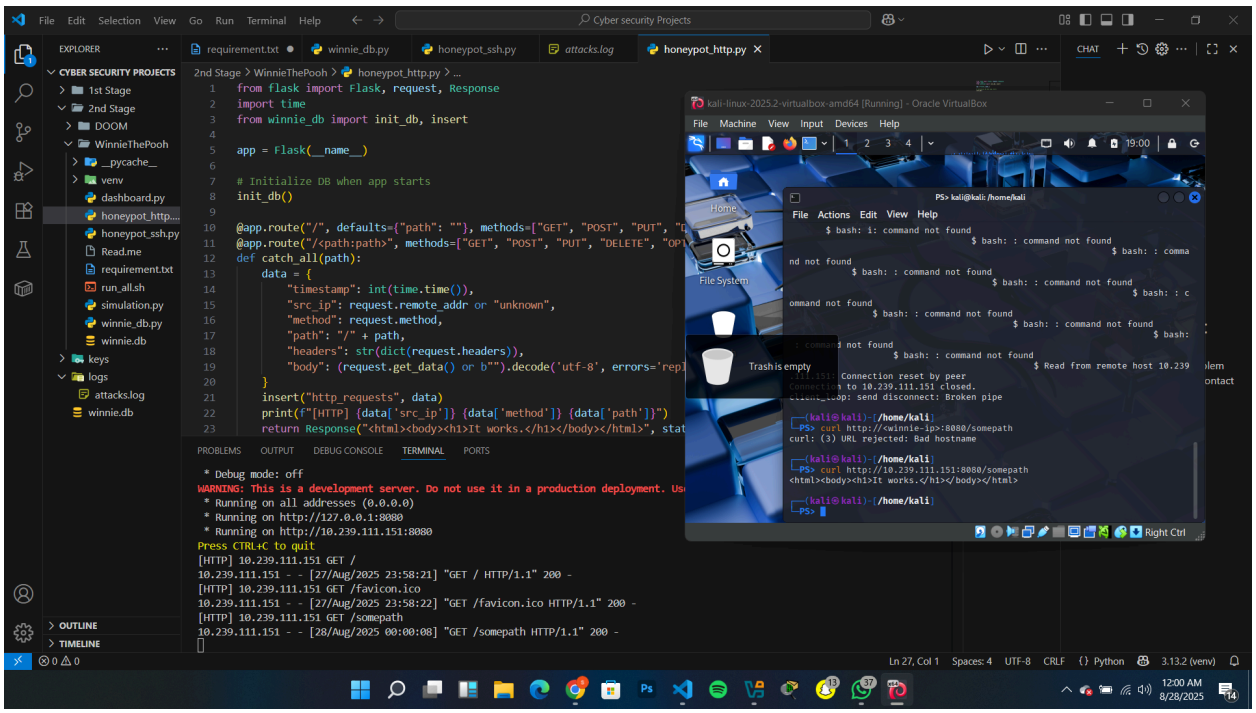


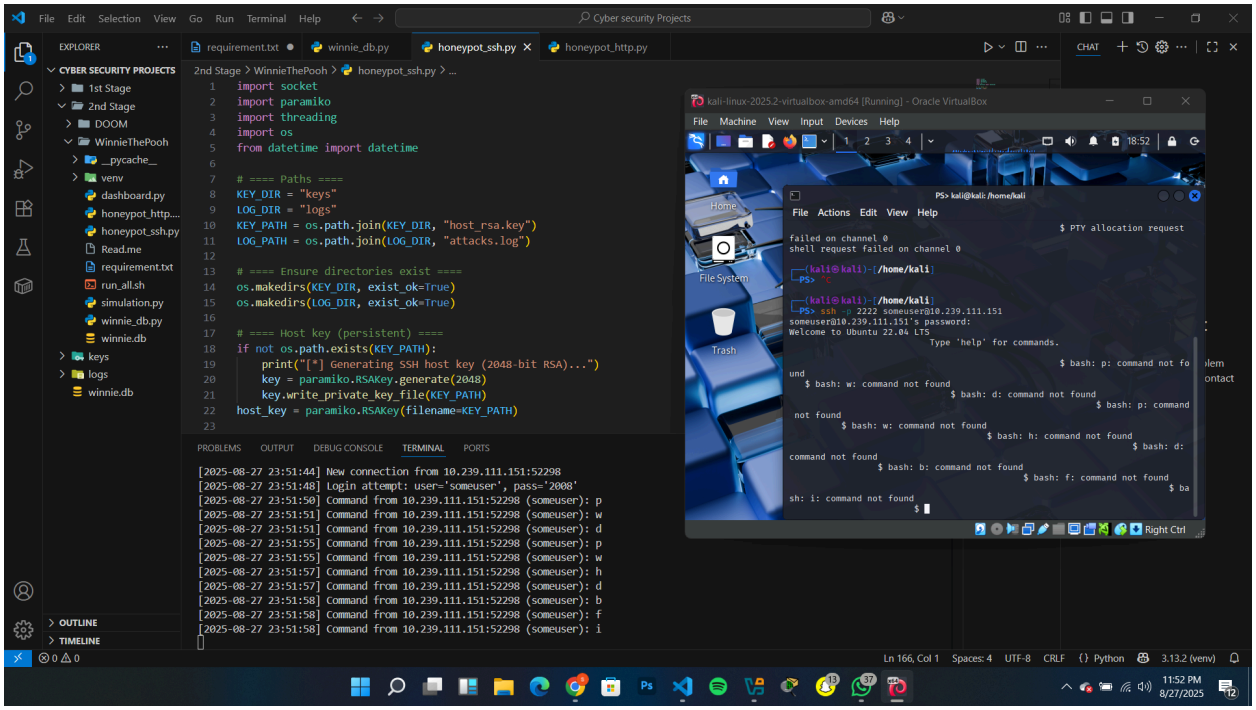
Step 2: SSH Honeypot

- Implemented in `honeypot_ssh.py` using **Paramiko**.
- The honeypot accepts connections but instead of providing a real shell, it **records attacker keystrokes** (keylogger).
- SSH keys were generated with:

```
ssh-keygen -t rsa -f host_rsa.key
```


File Edit Selection View Go Run Terminal Help
Cyber security Projects
EXPLORER
CYBER SECURITY PROJECTS
1st Stage
2nd Stage
DOOM
WinnieThePooh
__pycache__
venv
Include
Lib
Scripts
.gitignore
pyvenv.cfg
dashboard.py
honeypot_ssh.py
Read.me
requirement.txt
run_all.sh
simulation.py
winnie_db.py
winnie.db
host_rsa.key
host_rsa.key.pub
winnie.db
requirement.txt
winnie_db.py
honeypot_ssh.py
2nd Stage > WinnieThePooh > honeypot_ssh.py > ...
# honeypot_ssh.py
import socket
import threading
import paramiko
import time
from paramiko import RSAKey, ServerInterface, AUTH_SUCCESSFUL, AUTH_FAIL
import os
from winnie_db import init_db, insert
HOST_KEY_PATH = "host_rsa.key"
LISTEN_ADDR = "0.0.0.0"
PORT = 2222 # non-root port for easy testing
class HoneyServer(ServerInterface):
def __init__(self, client_addr):
self.event = threading.Event()
self.client_addr = client_addr
self.attempts = []
def check_auth_password(self, username, password):
# Log each attempt
data = {
"timestamp": int(time.time()),
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
| o ..o.o.++X.. |
| o o..==+oB |
+----[SHA256]-----+
PS C:\Users\USER\OneDrive\Desktop\Cyber security Projects> & "C:/Users/USER/OneDrive/Desktop/Cyber security Projects/2nd Stage/WinnieThePooh/venv/Scripts/python.exe" "c:/Users/USER/OneDrive/Desktop/Cyber security Projects/2nd Stage/WinnieThePooh/honeypot_ssh.py"
Host key not found. Generate with ssh-keygen -t rsa -f host_rsa.key -N ''
PS C:\Users\USER\OneDrive\Desktop\Cyber security Projects> & "C:/Users/USER/OneDrive/Desktop/Cyber security Projects/2nd Stage/WinnieThePooh/venv/Scripts/python.exe" "c:/Users/USER/OneDrive/Desktop/Cyber security Projects/2nd Stage/WinnieThePooh/honeypot_ssh.py"
[+] SSH honeypot listening on 0.0.0.0:2222
[+] Connection from ('192.168.0.109', 55948)
OUTLINE
TIMELINE
Ln 3, Col 17 Spaces: 4 UTF-8 CRLF {} Python 3.13.2 (venv)
11:11 PM
8/27/2025
kali-linux-2025.2-virtualbox-amd64 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Home
File System
Trash
PS> kali@kali: /home/kali
File Actions Edit View Help
PS> # enter wrong password a few times → check DB logs^C
PS> ssh -p 222 someuser@192.168.0.109
ssh: connect to host 192.168.0.109 port 222: Connection refused
PS> ^C
PS> ^C
PS> ssh -p 2222 someuser@192.168.0.109
The authenticity of host '[192.168.0.109]:2222 ([192.168.0.109]:2222)' can't be established.
RSA key fingerprint is SHA256:eEGI9NP8hs8H9ljyo6Lc37Zzgs7HWD4jglDL1ja4KBM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '[192.168.0.109]:2222' (RSA) to the list of known hosts.
someuser@192.168.0.109's password:
Right Ctrl


File Edit Selection View Go Run Terminal Help
Cyber security Projects
EXPLORER
CYBER SECURITY PROJECTS
1st Stage
2nd Stage
DOOM
WinnieThePooh
__pycache__
venv
dashboard.py
honeypot_http....
honeypot_ssh.py
Read.me
requirement.txt
run_all.sh
simulation.py
winnie_db.py
winnie.db
keys
logs
attacks.log
winnie.db
requirement.txt
winnie_db.py
honeypot_ssh.py
attacks.log
honeypot_http.py
2nd Stage > WinnieThePooh > honeypot_http.py > ...
from flask import Flask, request, Response
import time
from winnie_db import init_db, insert
app = Flask(__name__)
# Initialize DB when app starts
init_db()
@app.route("/", defaults={"path": ""}, methods=["GET", "POST", "PUT",
@app.route("/<path:path>", methods=["GET", "POST", "PUT", "DELETE",
def catch_all(path):
data = {
"timestamp": int(time.time()),
"src_ip": request.remote_addr or "unknown",
"method": request.method,
"path": "/" + path,
"headers": str(dict(request.headers)),
"body": (request.get_data() or b"").decode('utf-8', errors='rep
}
insert("http_requests", data)
print(f"[HTTP] {data['src_ip']} {data['method']} {data['path']}")
return Response("<html><body><h1>It works.</h1></body></html>", stat
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Us
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:8080
* Running on http://10.239.111.151:8080
Press CTRL+C to quit
[HTTP] 10.239.111.151 GET /
10.239.111.151 - - [27/Aug/2025 23:58:21] "GET / HTTP/1.1" 200 -
[HTTP] 10.239.111.151 GET /favicon.ico
10.239.111.151 - - [27/Aug/2025 23:58:22] "GET /favicon.ico HTTP/1.1" 200 -
[HTTP] 10.239.111.151 GET /somepath
10.239.111.151 - - [28/Aug/2025 00:00:08] "GET /somepath HTTP/1.1" 200 -
OUTLINE
TIMELINE
Ln 27, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.2 (venv)
12:00 AM
8/28/2025
kali-linux-2025.2-virtualbox-amd64 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Home
File System
Trash is empty
PS> kali@kali: /home/kali
File Actions Edit View Help
$ bash: i: command not found
$ bash: : command not found
$ bash: : command not found
$ bash: : command not found
$ bash: : command not found
$ bash: : command not found
$ bash: : command not found
$ bash: : command not found
$ bash: : command not found
$ Read from remote host 10.239
Connection reset by peer
Connection to 10.239.111.151 closed.
client_loop: send disconnect: Broken pipe
PS> curl http://<winnie-ip>:8080/somepath
curl: (3) URL rejected: Bad hostname
PS> curl http://10.239.111.151:8080/somepath
<html><body><h1>It works.</h1></body></html>
Right Ctrl

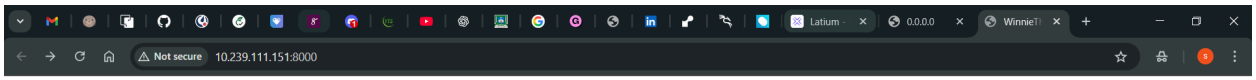


Step 3: HTTP Honeypot

- Built using **Flask** in `honeypot_http.py`.
- Simulates a vulnerable website where GET/POST requests are logged.
- Malicious scans and uploads can be captured.

Step 4: Web Dashboard

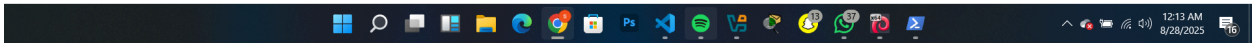
- Created `dashboard.py` in Flask to visualize attack logs from the DB.
- Displays attacker IPs, geolocation (if GeoLite2 DB is integrated), and types of attempts.

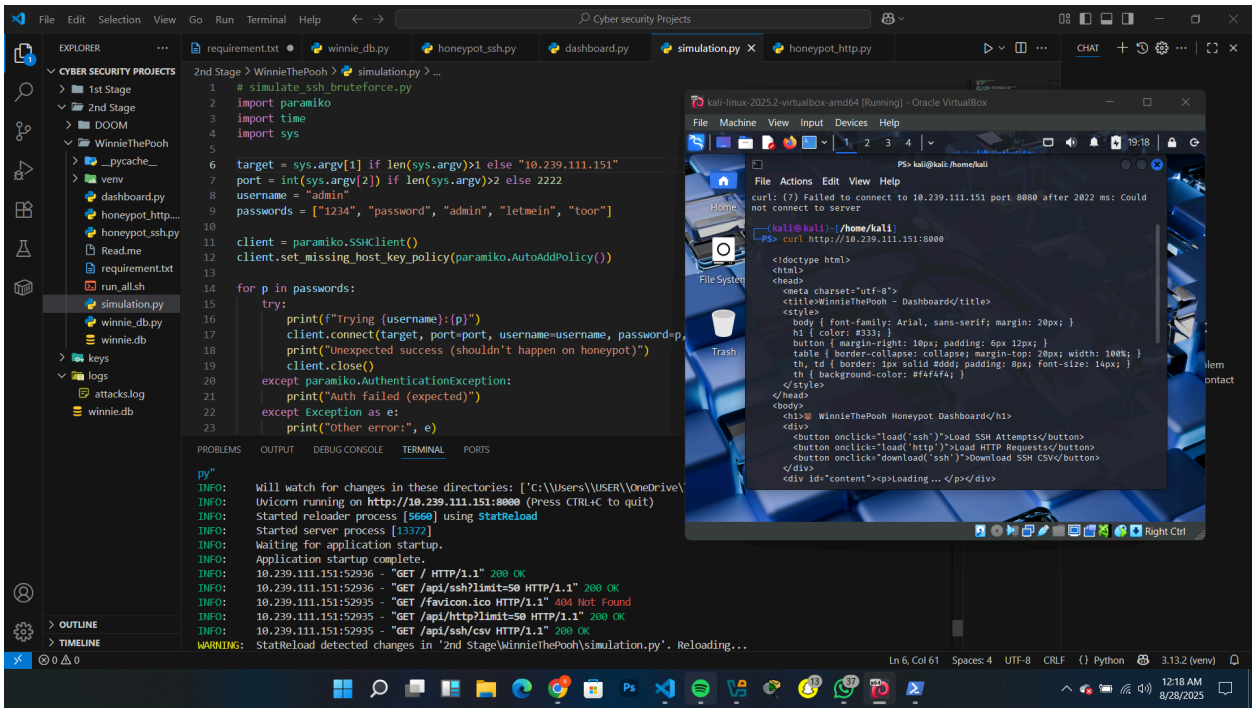


WinnieThePooh Honeypot Dashboard

Load SSH Attempts | Load HTTP Requests | Download SSH CSV

id	timestamp	src_ip	method	path	headers	body
3	1756335608	10.239.111.151	GET	/somepath	{'Host': '10.239.111.151:8080', 'User-Agent': 'curl/8.13.0', 'Accept': '*/*'}	
2	1756335502	10.239.111.151	GET	/favicon.ico	{'Host': '10.239.111.151:8080', 'Connection': 'keep-alive', 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36', 'Accept': 'image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8', 'Referer': 'http://10.239.111.151:8080/', 'Accept-Encoding': 'gzip, deflate', 'Accept-Language': 'en-US,en;q=0.9'}	
1	1756335501	10.239.111.151	GET	/	{'Host': '10.239.111.151:8080', 'Connection': 'keep-alive', 'Upgrade-Insecure-Requests': '1', 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36', 'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7', 'Accept-Encoding': 'gzip, deflate', 'Accept-Language': 'en-US,en;q=0.9'}	





5. Results & Observations

- **Successfully captured SSH brute-force attempts.**
- **Logged attacker keystrokes during SSH sessions.**
- **Detected HTTP scanning attempts from Kali Linux.**
- **Dashboard displayed real-time logs with attacker IPs.**

6. Conclusion

The WinnieThePooh Honeypot was successfully deployed and tested. It serves as a valuable tool for:

- **Studying attacker behavior.**
- **Demonstrating real-world cybersecurity threats.**
- **Educating others on password security and intrusion tactics.**

Future Improvements

- **Deploy on a public server/VPS to attract real attackers.**
- **Add email/SMS alerts for intrusion attempts.**
- **Expand honeypot services (e.g., FTP, RDP).**
- **Store data in Elasticsearch/Kibana for advanced visualization.**