Intrusion Detection Room

Tryhackme 💻

-Abhinav V R

Objective: To perform a full system takeover (initial access, privilege escalation, and persistence) on a target machine while actively monitoring and attempting to evade two primary Intrusion Detection Systems (IDS): Suricata (NIDS) and Wazuh (HIDS).

Reconnaissance and NIDS Evasion

This phase focuses on network scanning and information gathering, with an emphasis on evading Suricata.

Action/ Tool Used	Command Executed	IDS Activity & Evasion Outcome	Screenshots to Capture
Initial Nmap Scan (Detection)	`nmap -sV [Target_IP]`	Suricata: High-severity alert for "Potential network scan/probes" is immediately triggered.	1. Nmap output showing open ports/services. 2. Suricata console showing the alert.

Nmap Evasion Attempt	`nmap --script http-useragent-mod --script-args http.useragent="[New_Agent]" [Target_IP]`	Suricata: By modifying the User-Agent, the new scan may not trigger the same signature-based alert, resulting in lower severity/no alert.	1. The Nmap command with the evasion flag. 2. The Suricata console showing a reduction in alerts (or a different, lower-severity alert).
Web Enumeration (Nikto)	`nikto -h http://[Target_IP]:[Port]`	Suricata: Multiple high-severity alerts for "Web application scanning" and specific vulnerability checks. Identified key path (e.g., `/login`).	1. Nikto scan output. 2. Suricata console showing the barrage of Nikto alerts.
Nikto Evasion Attempt	`nikto -h http://[Target_IP] -T 6 -evasion 6,a,b`	Suricata: Using scan tuning (`-T 6` for DoS vectors) and evasion flags (`6,a,b` for request spacing) can reduce the scan's aggressiveness and potentially evade some high-severity rules.	Nikto command with evasion flags and the subsequent (quieter) Suricata activity.
Open Source Intelligence (OSINT)	Searching Shodan, Google Dorking for exposed services and versions (e.g., Grafana 8.2.5).	None: Passive intelligence gathering does not touch the target, resulting in zero IDS alerts.	Screenshot of the OSINT tool/search query and the resulting vulnerability (e.g., Grafana CVE-2021-43798).

Initial Access and Exploitation

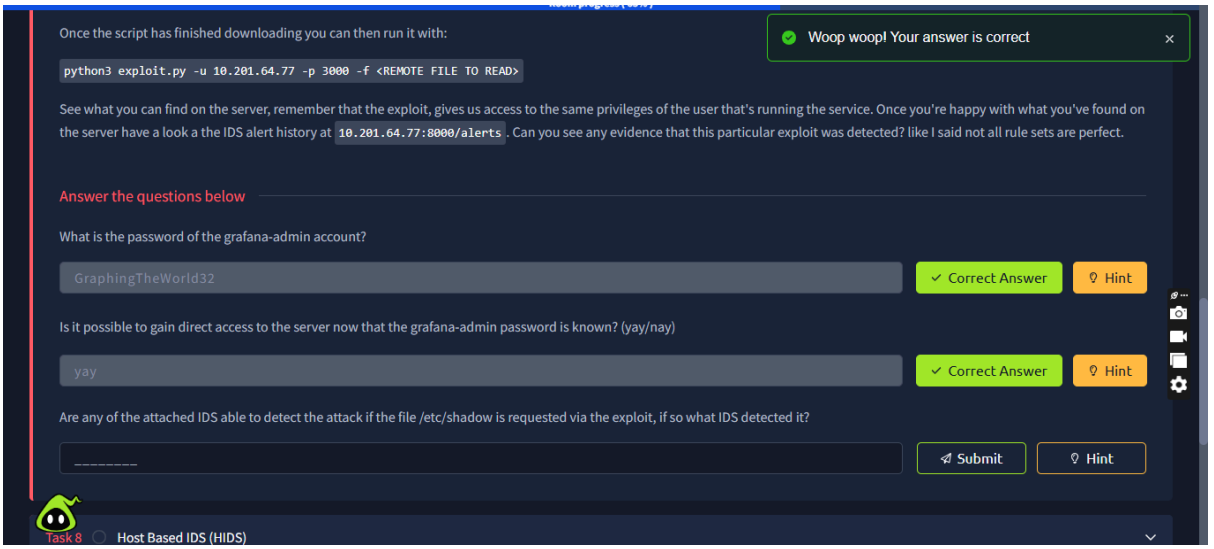
Action/Tool Used	Command Executed	IDS Activity & Evasion Outcome	Screenshots to Capture
Exploitation	Requesting a sensitive file via the LFI exploit (e.g., accessing `/etc/shadow` via the Grafana vulnerability).	Suricata: Detected as a "Malicious File Access" or "Traversal Attempt" due to the specific request pattern (the path traversal payload).	1. The exploit command/request. 2. The Suricata alert specifically flagging the attack.
Gaining a Shell	Setting up a listener (`nc -lvnp 4242`) and sending the reverse shell payload from the target.	Suricata: Triggers a high-severity alert for "Command and Control (C2) traffic" or "Reverse Shell Connection."	1. Netcat listener successfully catching the shell. 2. Suricata alert for the reverse shell.

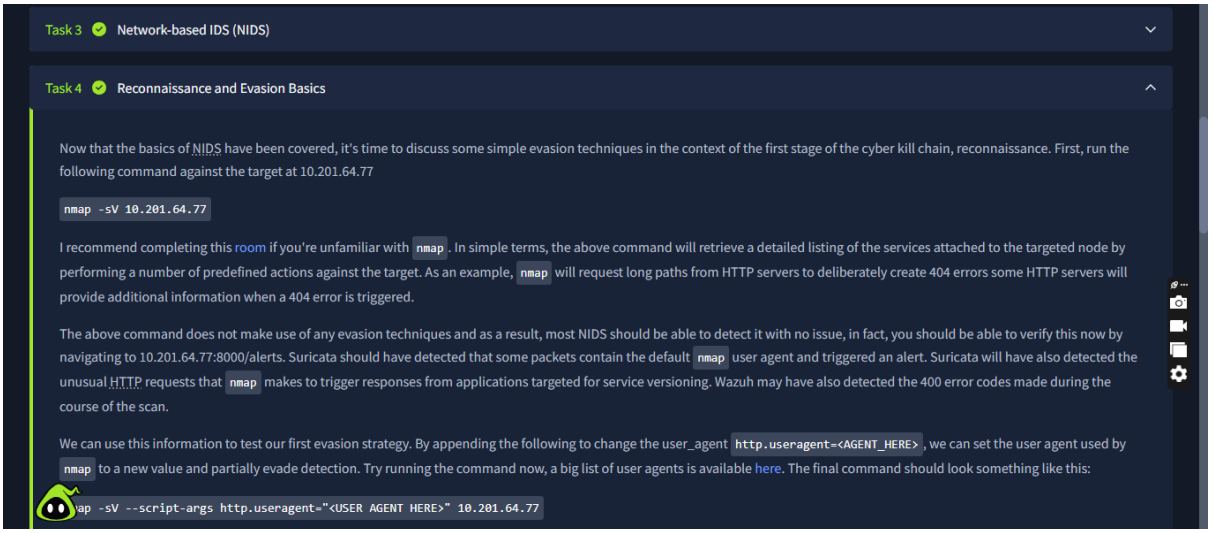
Privilege Escalation and HIDS Evasion

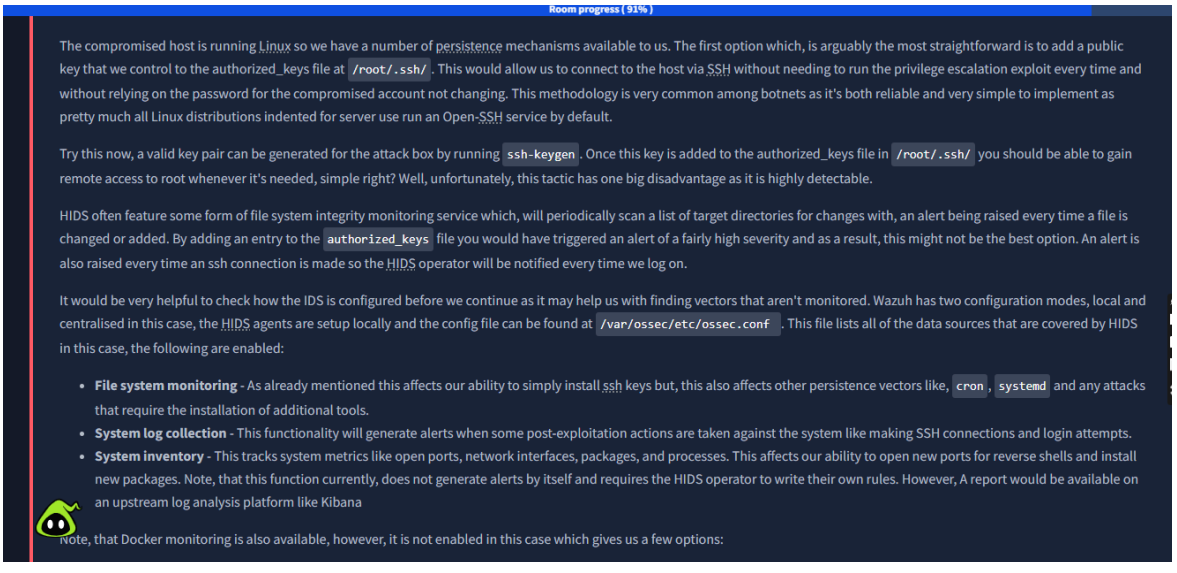
Action/Tool Used	Command Executed	IDS Activity & Evasion Outcome	Screenshots to Capture
Transferring Post-Exploit Tools	Transferring LinPEAS to the compromised host (e.g., using `wget` or `python -m SimpleHTTPServer`).	Wazuh: Triggers a Level 5 alert for "File Integrity/System Audit" when the script is added, as LinPEAS is a known enumeration tool.	1. Command to transfer LinPEAS. 2. Wazuh console showing the file addition alert.
Privilege Escalation Recon	Running LinPEAS to find a vector, which points to a misconfigured Docker installation.	Wazuh: Running the script itself may generate additional "System Auditing" or "Unusual Process Execution" alerts.	LinPEAS output highlighting the Docker misconfiguration.

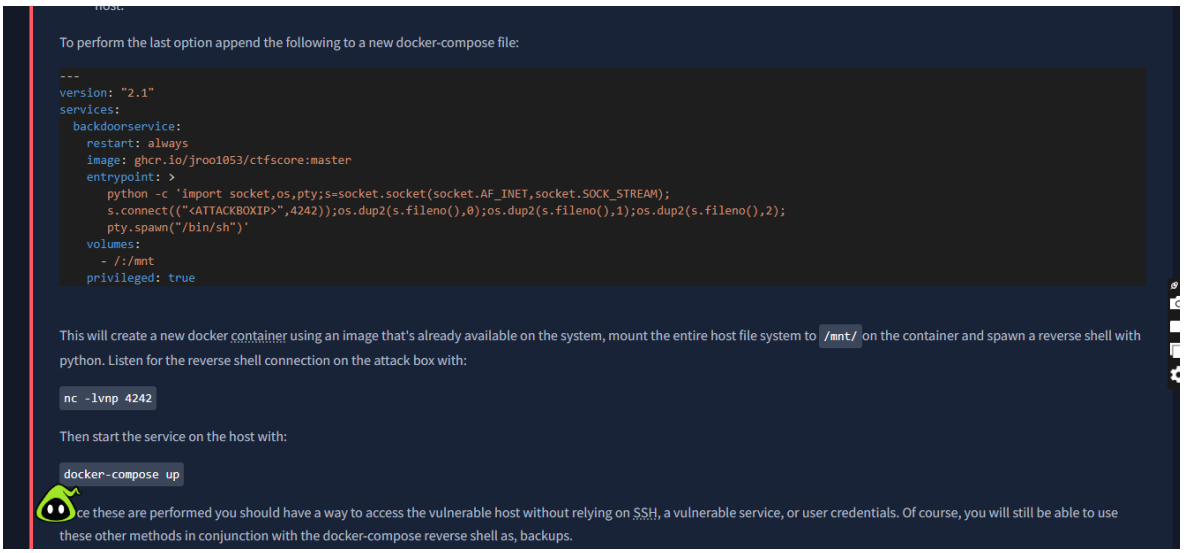
Attempted Persistence (Failed)	Attempting to establish persistence via a simple SSH key addition: `echo "key" >> /root/.ssh/authorized_keys`	Wazuh: CRITICAL Alert (High Severity) from File Integrity Monitoring (FIM) for modification of a critical system file (`/root/.ssh/authorized_keys`).	1. The SSH key modification command. 2. The Wazuh Critical FIM alert.
Evasive Persistence (Success)	Abusing Docker by creating a new `docker-compose.yml` file with a reverse shell entry point and mounting the root directory. This method avoids direct modification of sensitive files monitored by FIM.	Wazuh: This technique is designed to be evasive by manipulating a less-monitored configuration file (`docker-compose.yml`) and leveraging a legitimate service (Docker). This should result in low-severity or no alerts compared to the FIM critical alert.	1. Content of the evasive `docker-compose.yml` file (as seen in the room content). 2. The command to start the persistent service. 3. The final root shell connection.
Final Flag Retrieval	`cat /root/flag.txt`	None (System takeover complete and persistent access established).	Screenshot showing the final root flag: `{SNEAK_ATTACK_CRITICAL}`.

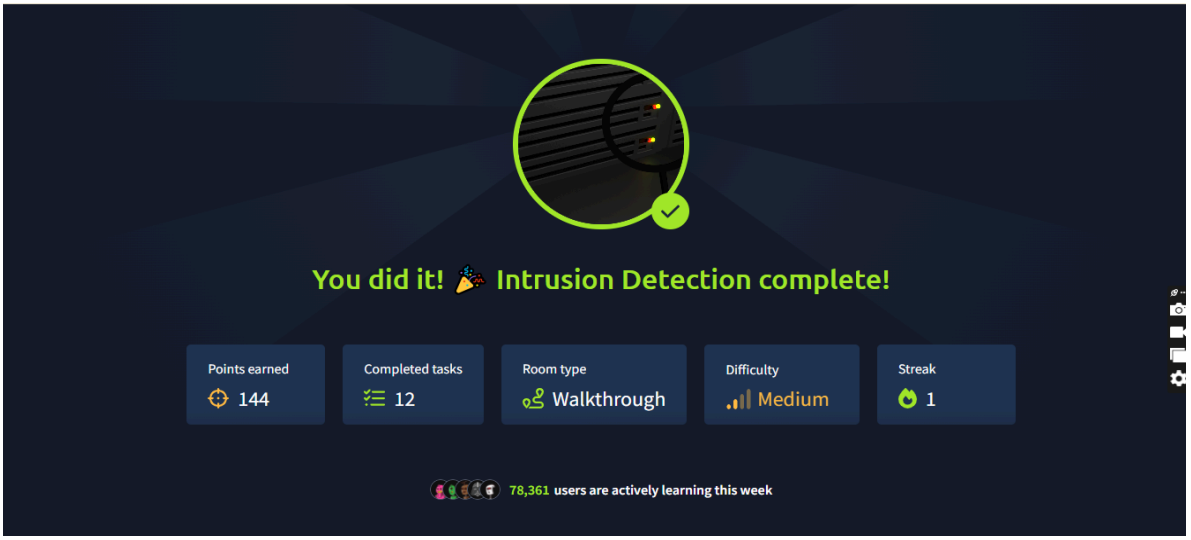
Screenshots:











Result: **The system takeover was successful, culminating in the retrieval of the root flag `{SNEAK_ATTACK_CRITICAL}`. Key stages of the attack, particularly initial scanning and simple persistence attempts, were detected. Evasion techniques were required to successfully complete the final stages of the cyber kill chain without detection.**