Task 8 — Intrusion Detection Room

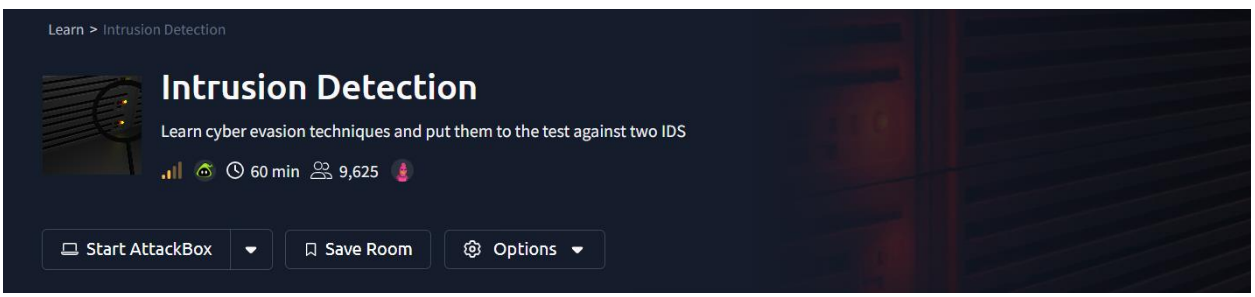
Room: https://tryhackme.com/room/idsevasion

Task: Prepare a report while doing the room, add screenshots and explain what is happening while you are doing the room.

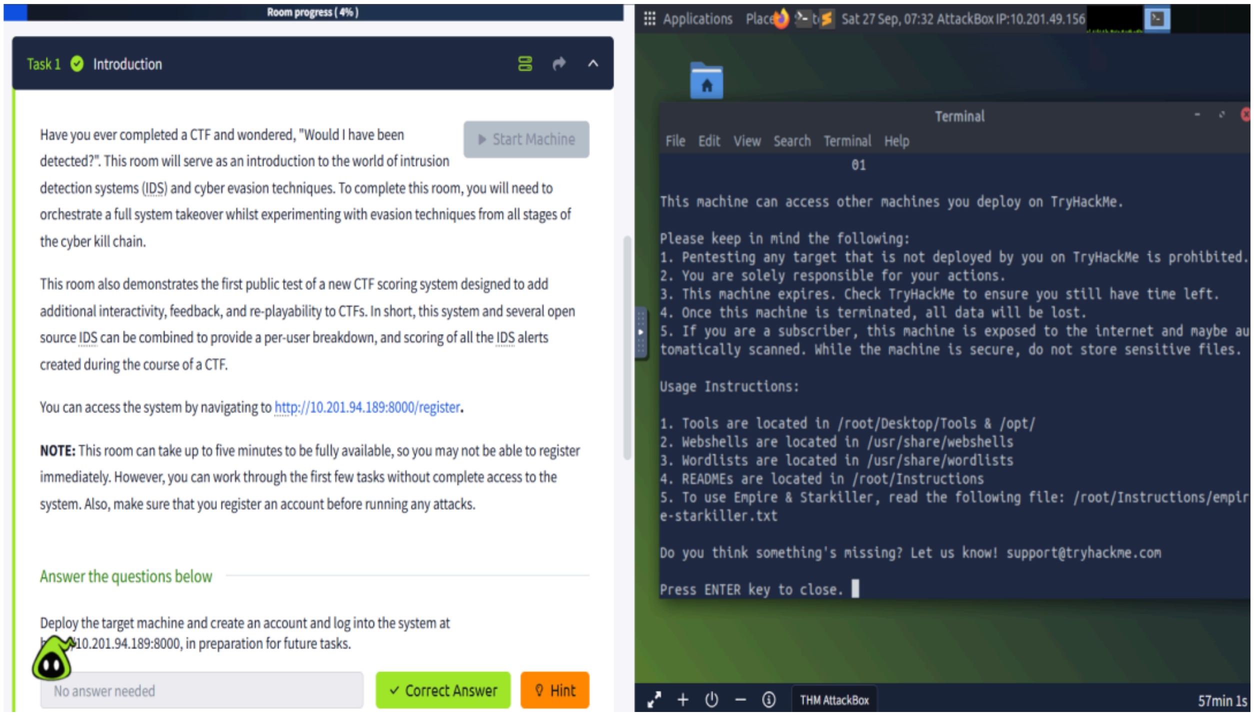
Cover Page

- **Title:** IDSEvasion — TryHackMe Room Report
- **Student / Tester:** *Yedhukrishna*
- **Course:** Cyber Security Bootcamp — MuLearn / OWASP Kerala
- **Room Link:** https://tryhackme.com/room/idsevasion
- **Profile :** https://tryhackme.com/p/yedhu.irl
- **Date:** 1/10/2025

Task 1

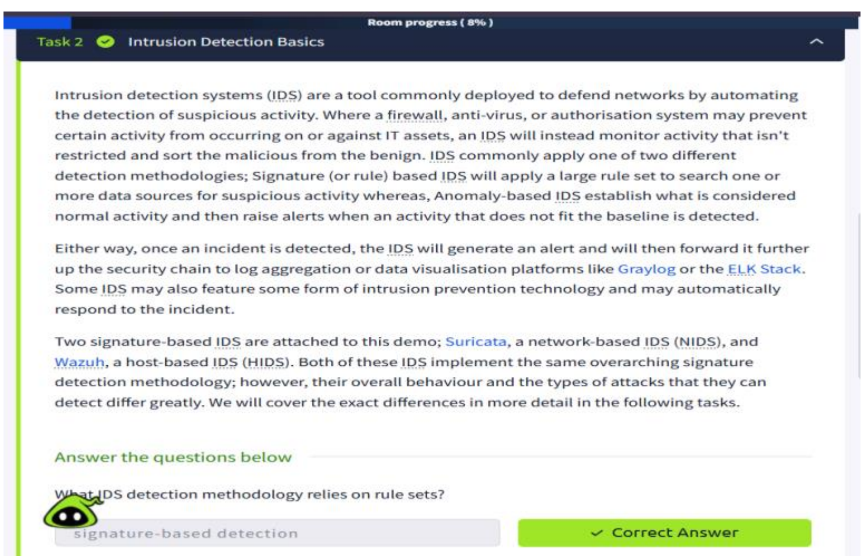


Introduction I deployed the TryHackMe machine and registered a user on the web interface to ensure activity and alerts would be linked to my session. I noted the target IP and web ports (for example port 3000) and confirmed the alerts page was reachable at http://MACHINE_IP:8000/alerts. This setup step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



Task 2 : Intrusion Detection Basics

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.

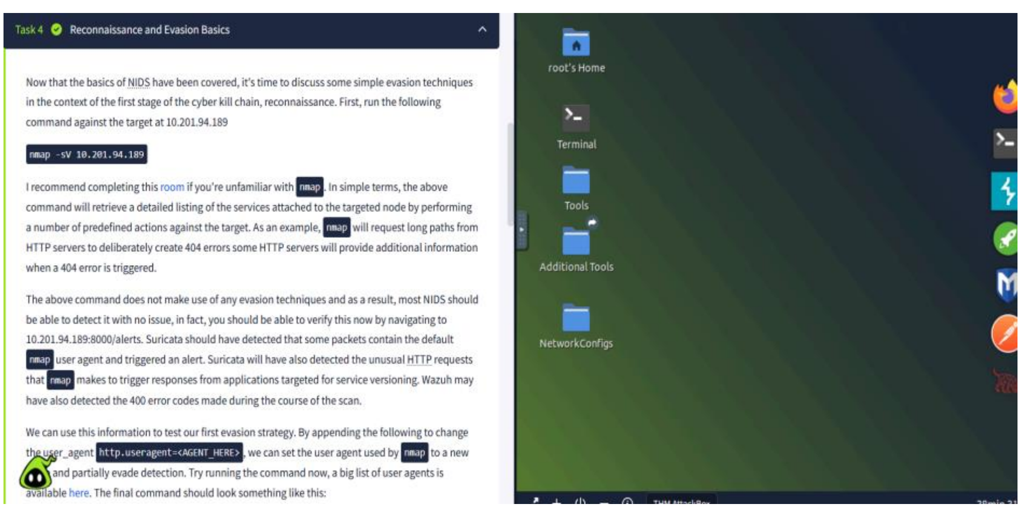


Task 3 : Network-based IDS (NIDS)

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

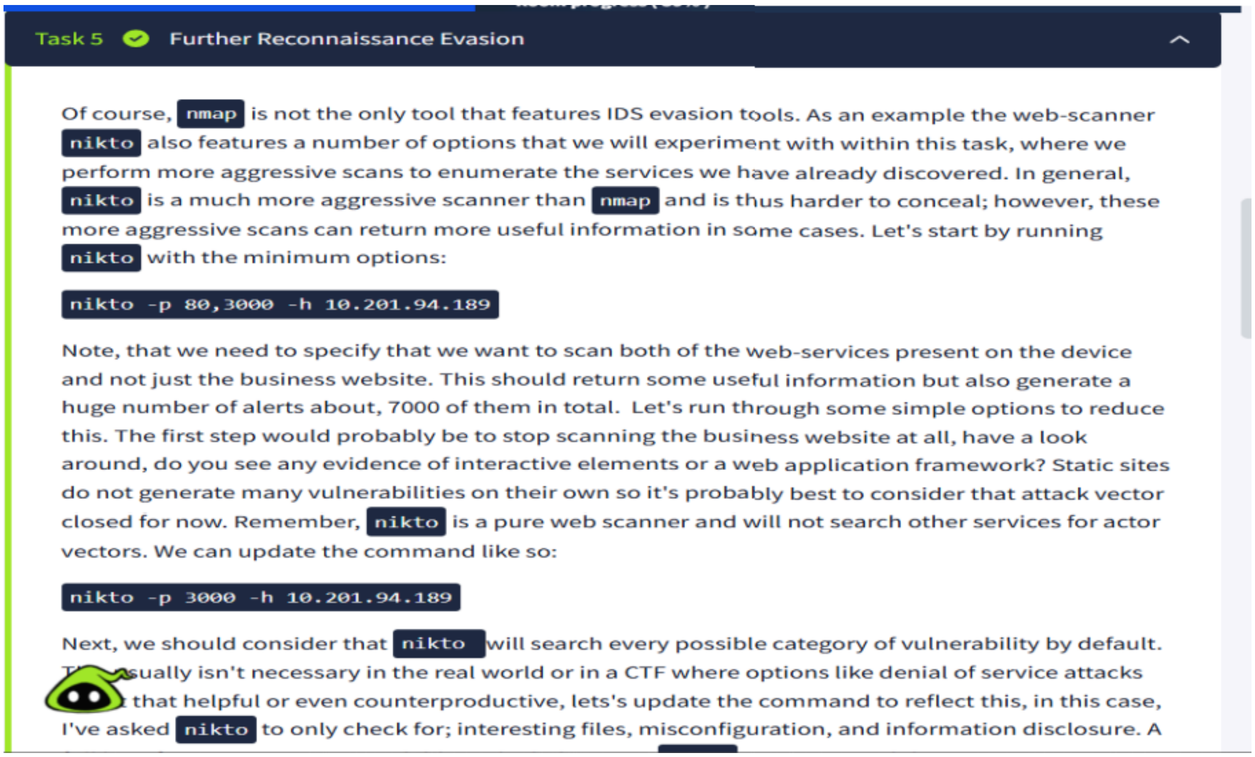
Task 4 : Reconnaissance and Evasion Basics

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`

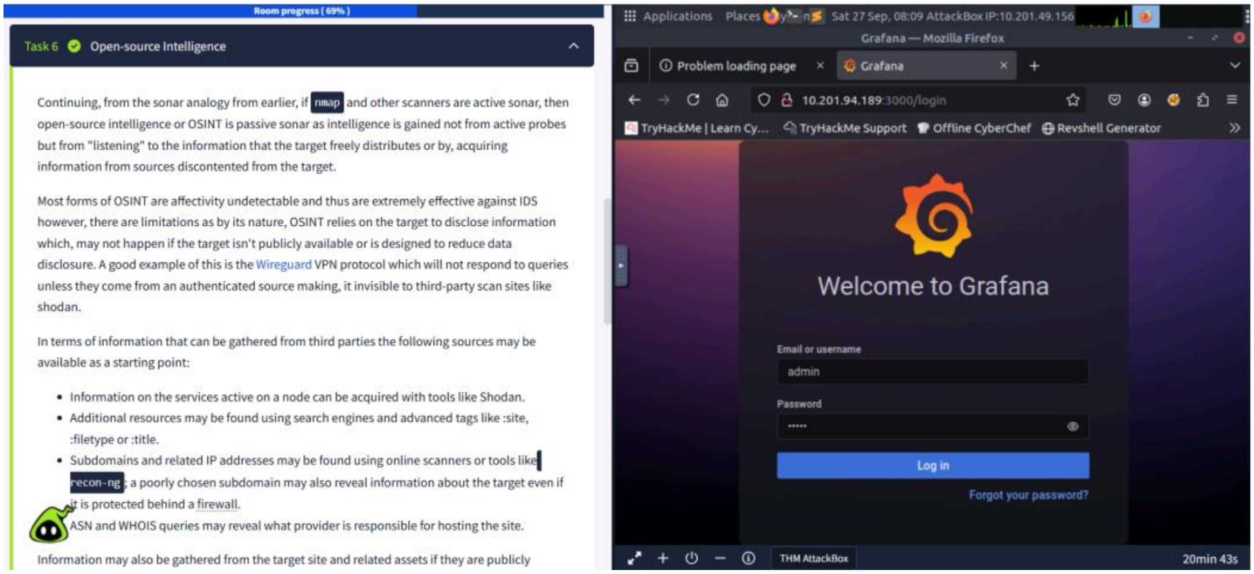
Note, that we need to specify that we want to scan both of the web-services present on the device and not just the business website. This should return some useful information but also generate a huge number of alerts about, 7000 of them in total. Let's run through some simple options to reduce this. 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. T... usually isn't necessary in the real world or in a CTF where options like denial of service attacks ... 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

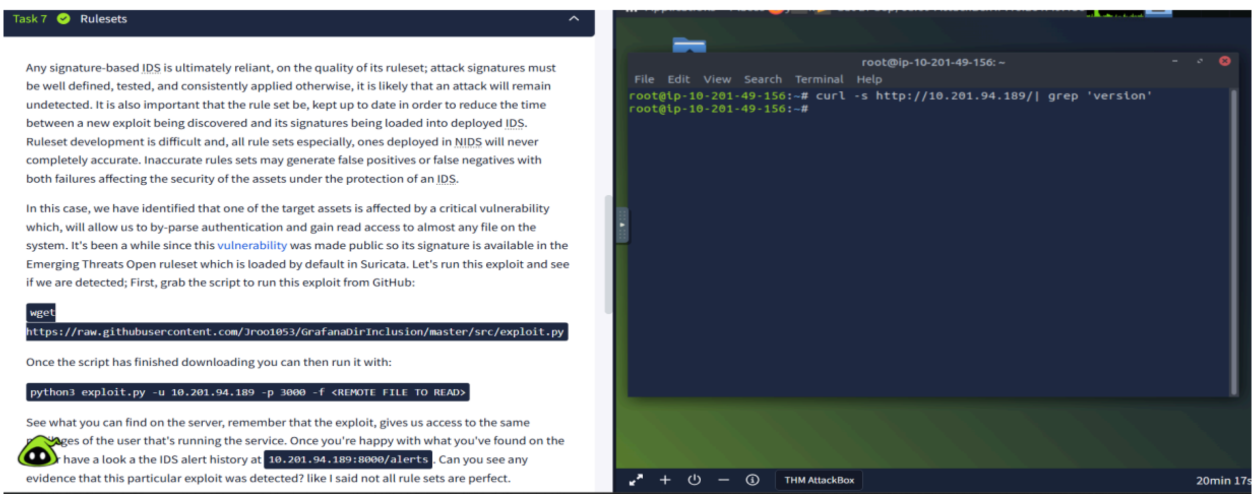
Task 6 : Open-source Intelligence

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 8 : Host Based IDS (HIDS)

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 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

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

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 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:

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.

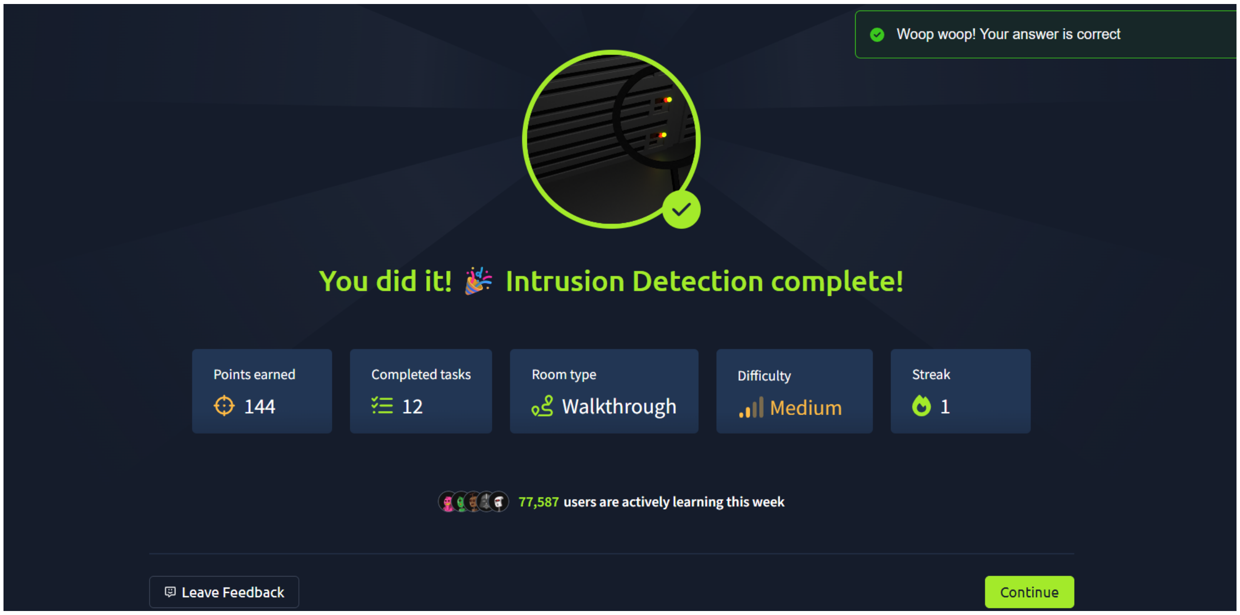
Try this now, a valid key pair can be generated for the attack box by running `ssh-keygen`. Once this key is added to the authorized_keys file in `/root/.ssh/` you should be able to gain remote access to root whenever it's needed, simple right? Well, unfortunately, this tactic has one big disadvantage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

Task 12 : Conclusion

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.


Woop woop! Your answer is correct
You did it! Intrusion Detection complete!
Points earned
144
Completed tasks
12
Room type
Walkthrough
Difficulty
Medium
Streak
1
77,587 users are actively learning this week
Leave Feedback
Continue