

# 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 <b>PHP</b> 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:

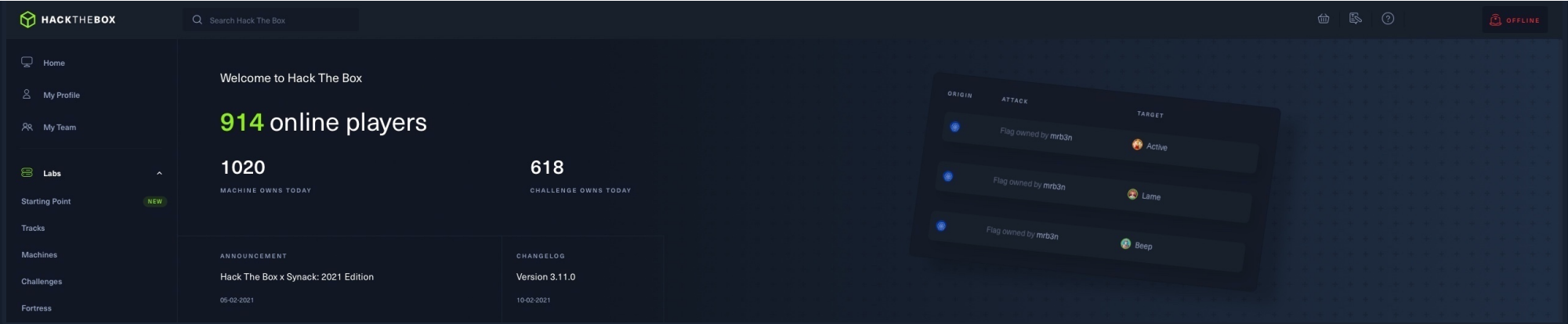
Port(s)	Protocol
<b>20/21</b> (TCP)	<b>FTP</b>
<b>22</b> (TCP)	<b>SSH</b>
<b>23</b> (TCP)	<b>TeLnet</b>
<b>25</b> (TCP)	<b>SMTP</b>
<b>80</b> (TCP)	<b>HTTP</b>
<b>161</b> (TCP/UDP)	<b>SNMP</b>
<b>389</b> (TCP/UDP)	<b>LDAP</b>
<b>443</b> (TCP)	<b>SSL/TLS (HTTPS)</b>
<b>445</b> (TCP)	<b>SMB</b>
<b>3389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

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.	<a href="#">Broken Access Control</a>	Restrictions are not appropriately implemented to prevent users from accessing other users accounts, viewing sensitive data, accessing unauthorized functionality, modifying data, etc.
2.	<a href="#">Cryptographic Failures</a>	Failures related to cryptography which often leads to sensitive data exposure or system compromise.
3.	<a href="#">Injection</a>	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.	<a href="#">Insecure Design</a>	These issues happen when the application is not designed with security in mind.
5.	<a href="#">Security Misconfiguration</a>	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.	<a href="#">Vulnerable and Outdated Components</a>	Using components (both client-side and server-side) that are vulnerable, unsupported, or out of date.
7.	<a href="#">Identification and Authentication Failures</a>	Authentication-related attacks that target user's identity, authentication, and session management.
8.	<a href="#">Software and Data Integrity Failures</a>	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.	<a href="#">Security Logging and Monitoring Failures</a>	This category is to help detect, escalate, and respond to active breaches. Without logging and monitoring, breaches cannot be detected..
10.	<a href="#">Server-Side Request Forgery</a>	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.


Table of Contents

Introduction

Infosec Overview

✓

Setup

 Getting Started with a Pentest Distro

✓

Staying Organized

✓


Connecting Using VPN


✓

Pentesting Basics

[Common Terms](#)

 Basic Tools

 Service Scanning

 Web Enumeration

 Public Exploits

Types of Shells

 Privilege Escalation


Transferring Files

Getting Started with Hack The Box (HTB)


Starting Out

Navigating HTB

Attacking Your First Box

 Nibbles - Enumeration

 Nibbles - Web Footprinting

 Nibbles - Initial Foothold

 Nibbles - Privilege Escalation

Nibbles - Alternate User Method - Metasploit

Problem Solving

Common Pitfalls

Getting Help

What's Next?

Next Steps

 Knowledge Check

My Workstation

▶ Start Instance

1 / 1 spawns left