Networking and Computing

Extending Your Network

* Port Forwarding
  + Port forwarding is an essential component in connecting applications and services to the Internet. Without port forwarding, applications and services such as web servers are only available to devices within the same direct network.
  + Take the network below as an example. Within this network, the server with an IP address of "192.168.1.10" runs a webserver on port 80. Only the two other computers on this network will be able to access it (this is known as an intranet).
  + If the administrator wanted the website to be accessible to the public (using the Internet), they would have to implement port forwarding
  + It is easy to confuse port forwarding with the behaviours of a firewall (a technology we'll come on to discuss in a later task). However, at this stage, just understand that port forwarding opens specific ports (recall how packets work). In comparison, firewalls determine if traffic can travel across these ports (even if these ports are open by port forwarding).
  + Port forwarding is configured at the router of a network.
* Firewalls 101
  + A firewall is a device within a network responsible for determining what traffic is allowed to enter and exit. Think of a firewall as border security for a network. An administrator can configure a firewall to permit or deny traffic from entering or exiting a network based on numerous factors such as:
  + Where the traffic is coming from? (has the firewall been told to accept/deny traffic from a specific network?)
  + Where is the traffic going to? (has the firewall been told to accept/deny traffic destined for a specific network?)
  + What port is the traffic for? (has the firewall been told to accept/deny traffic destined for port 80 only?)
  + What protocol is the traffic using? (has the firewall been told to accept/deny traffic that is UDP, TCP or both?)
  + Firewalls perform packet inspection to determine the answers to these questions.
  + Firewalls come in all shapes and sizes. From dedicated pieces of hardware (often found in large networks like businesses) that can handle a magnitude of data to residential routers (like at your home!) or software such as Snort, firewalls can be categorised into 2 to 5 categories.
  + We'll cover the two primary categories of firewalls in the table below:

Graphical user interface, text, application, email

Description automatically generated

* VPN Basics
  + A Virtual Private Network (or VPN for short) is a technology that allows devices on separate networks to communicate securely by creating a dedicated path between each other over the Internet (known as a tunnel). Devices connected within this tunnel form their own private network.
  + For example, only devices within the same network (such as within a business) can directly communicate. However, a VPN allows two offices to be connected.
  + The devices connected on Network #3 are still a part of Network #1 and Network #2 but also form together to create a private network (Network #3) that only devices that are connected via this VPN can communicate over.
  + Let's cover some of the other benefits offered by a VPN in the table below:

Graphical user interface, text, application

Description automatically generated

* + TryHackMe uses a VPN to connect you to our vulnerable machines without making them directly accessible on the Internet! This means that:
  + You can securely interact with our machines
  + Service providers such as ISPs don't think you are attacking another machine on the Internet (which could be against the terms of service)
  + The VPN provides security to TryHackMe as vulnerable machines are not accessible using the Internet.
  + VPN technology has improved over the years. Let's explore some existing VPN technologies below

Graphical user interface, text, application, email

Description automatically generated

* LAN Networking Devices
  + Routers
    - It's a router's job to connect networks and pass data between them. It does this by using routing (hence the name router!).
    - Routing is the label given to the process of data travelling across networks. Routing involves creating a path between networks so that this data can be successfully delivered. Routers operate at Layer 3 of the OSI model. They often feature an interactive interface (such as a website or a console) that allows an administrator to configure various rules such as port forwarding or firewalling.
    - Routing is useful when devices are connected by many paths, such as in the example diagram below, where the most optimal path is taken:
    - Routers are dedicated devices and do not perform the same functions as switches.
    - We can see that Computer A's network is connected to the network of Computer B by two routers in the middle. The question is: what path will be taken? Different protocols will decide what path should be taken, but factors include:
    - - What path is the shortest?
    - - What path is the most reliable?
    - - Which path has the faster medium (e.g. copper or fibre)?
  + What is a Switch?
    - A switch is a dedicated networking device responsible for providing a means of connecting to multiple devices. Switches can facilitate many devices (from 3 to 63) using Ethernet cables.
    - Switches can operate at both layer 2 and layer 3 of the OSI model. However, these are exclusive in the sense that Layer 2 switches cannot operate at layer 3.
    - Take, for example, a layer 2 switch in the diagram below. These switches will forward frames (remember these are no longer packets as the IP protocol has been stripped) onto the connected devices using their MAC address.
    - These switches are solely responsible for sending frames to the correct device.
    - Now, let's move onto layer 3 switches. These switches are more sophisticated than layer 2, as they can perform some of the responsibilities of a router. Namely, these switches will send frames to devices (as layer 2 does) and route packets to other devices using the IP protocol.
    - Let's take a look at the diagram below of a layer 3 switch in action. We can see that there are two IP addresses:
    - 192.168.1.1
    - 192.168.2.1
    - A technology called VLAN (Virtual Local Area Network) allows specific devices within a network to be virtually split up. This split means they can all benefit from things such as an Internet connection but are treated separately. This network separation provides security because it means that rules in place determine how specific devices communicate with each other. This segregation is illustrated in the diagram below:

Intro to LAN

* LAN Topologies
  + Star Topology
    - The main premise of a star topology is that devices are individually connected via a central networking device such as a switch or hub. This topology is the most commonly found today because of its reliability and scalability - despite the cost.
    - Any information sent to a device in this topology is sent via the central device to which it connects. Let's explore some of these advantages and disadvantages of this topology below:
    - Because more cabling & the purchase of dedicated networking equipment is required for this topology, it is more expensive than any of the other topologies. However, despite the added cost, this does provide some significant advantages. For example, this topology is much more scalable in nature, which means that it is very easy to add more devices as the demand for the network increases.
    - Unfortunately, the more the network scales, the more maintenance is required to keep the network functional. This increased dependence on maintenance can also make troubleshooting faults much harder. Furthermore, the star topology is still prone to failure - albeit reduced. For example, if the centralised hardware that connects devices fails, these devices will no longer be able to send or receive data. Thankfully, these centralised hardware devices are often robust.
  + Bus Topology
    - This type of connection relies upon a single connection which is known as a backbone cable. This type of topology is similar to the leaf off of a tree in the sense that devices (leaves) stem from where the branches are on this cable.
    - Because all data destined for each device travels along the same cable, it is very quickly prone to becoming slow and bottlenecked if devices within the topology are simultaneously requesting data. This bottleneck also results in very difficult troubleshooting because it quickly becomes difficult to identify which device is experiencing issues with data all travelling along the same route.
    - However, with this said, bus topologies are one of the easier and more cost-efficient topologies to set up because of their expenses, such as cabling or dedicated networking equipment used to connect these devices.
    - Lastly, another disadvantage of the bus topology is that there is little redundancy in place in case of failures. This disadvantage is because there is a single point of failure along the backbone cable. If this cable were to break, devices can no longer receive or transmit data along the bus.
  + Ring Topology
    - The ring topology (also known as token topology) boasts some similarities. Devices such as computers are connected directly to each other to form a loop, meaning that there is little cabling required and less dependence on dedicated hardware such as within a star topology.
    - A ring topology works by sending data across the loop until it reaches the destined device, using other devices along the loop to forward the data. Interestingly, a device will only send received data from another device in this topology if it does not have any to send itself. If the device happens to have data to send, it will send its own data first before sending data from another device.
    - Because there is only one direction for data to travel across this topology, it is fairly easy to troubleshoot any faults that arise. However, this is a double-edged sword because it isn't an efficient way of data travelling across a network, as it may have to visit many multiple devices first before reaching the intended device.
    - Lastly, ring topologies are less prone to bottlenecks, such as within a bus topology, as large amounts of traffic are not travelling across the network at any one time. The design of this topology does, however, mean that a fault such as cut cable, or broken device will result in the entire networking breaking.
  + What is a Router?
    - It's a router's job to connect networks and pass data between them. It does this by using routing (hence the name router!).
    - Routing is the label given to the process of data travelling across networks. Routing involves creating a path between networks so that this data can be successfully delivered.
    - Routing is useful when devices are connected by many paths, such as in the example diagram below.
  + What is a Switch?
    - Switches are dedicated devices within a network that are designed to aggregate multiple other devices such as computers, printers, or any other networking-capable device using ethernet. These various devices plug into a switch's port. Switches are usually found in larger networks such as businesses, schools, or similar-sized networks, where there are many devices to connect to the network. Switches can connect a large number of devices by having ports of 4, 8, 16, 24, 32, and 64 for devices to plug into.
    - Unlike Routers, these devices do not perform routing in the sense of directing paths along a certain route using the IP protocol. Instead, Switches use a technology called "packet switching" to break down pieces of data into smaller, more manageable chunks of data called packets. This technology allows for the efficiency of a network because large pieces of data take up more resources -- slowing down a busy network.
  + Both Switches and Routers can be connected to one another. The ability to do this increases the redundancy (the reliability) of a network by adding multiple paths for data to take. If one path goes down, another can be used. Whilst this may reduce the overall performance of a network because packets have to take longer to travel, there is no downtime -- a small price to pay considering the alternative.
* Subnetting
  + Subnetting is achieved by splitting up the number of hosts that can fit within the network, represented by a number called a subnet mask. Let's refer back to our diagram from the first room in this module:
  + As we can recall, an IP address is made up of four sections called octets. The same goes for a subnet mask which is also represented as a number of 8 bytes (32 bits), ranging from 0 to 255 (0-255).
  + Subnets use IP addresses in three different ways:
    - Identify the network address
    - Identify the host address
    - Identify the default gateway
  + Let's split these three up to understand their purposes into the table below:

Table

Description automatically generated

* Subnetting
  + Now, in small networks such as at home, you will be on one subnet as there is an unlikely chance that you need more than 254 devices connected at one time.
  + However, places such as businesses and offices will have much more of these devices (PCs, printers, cameras and sensors), where subnetting takes place.
  + Subnetting provides a range of benefits, including:
    - Efficiency
    - Security
    - Full control
  + We'll come on to explore exactly how subnetting provides these benefits at a later date; however, for now, all we need to understand is the security element to it. Let's take the typical café on the street. This cafe will have two networks:
    - One for employees, cash registers, and other devices for the facility
    - One for the general public to use as a hotspot
  + Subnetting allows you to separate these two use cases from each other whilst having the benefits of a connection to larger networks such as the Internet.
* ARP Protocol
  + - Recalling from our previous tasks that devices can have two identifiers: A MAC address and an IP address, the ARP protocol or Address Resolution Protocol for short, is the technology that is responsible for allowing devices to identify themselves on a network.
    - Simply, the ARP protocol allows a device to associate its MAC address with an IP address on the network. Each device on a network will keep a log of the MAC addresses associated with other devices.
    - When devices wish to communicate with another, they will send a broadcast to the entire network searching for the specific device. Devices can use the ARP protocol to find the MAC address (and therefore the physical identifier) of a device for communication
  + How does ARP Work?
    - Each device within a network has a ledger to store information on, which is called a cache. In the context of the ARP protocol, this cache stores the identifiers of other devices on the network.
    - In order to map these two identifiers together (IP address and MAC address), the ARP protocol sents two types of messages:
      * ARP Request
      * ARP Reply
    - When an ARP request is sent, a message is broadcasted to every other device found on a network by the device, asking whether or not the device's MAC address matches the requested IP address. If the device does have the requested IP address, an ARP reply is returned to the initial device to acknowledge this. The initial device will now remember this and store it within its cache (an ARP entry).
* DHCP Protocol
  + IP addresses can be assigned either manually, by entering them physically into a device, or automatically and most commonly by using a DHCP (Dynamic Host Configuration Protocol) server. When a device connects to a network, if it has not already been manually assigned an IP address, it sends out a request (DHCP Discover) to see if any DHCP servers are on the network. The DHCP server then replies back with an IP address the device could use (DHCP Offer). The device then sends a reply confirming it wants the offered IP Address (DHCP Request), and then lastly, the DHCP server sends a reply acknowledging this has been completed, and the device can start using the IP Address (DHCP ACK).

Packets & Frames

* Packets and Frames
  + Packets and frames are small pieces of data that, when forming together, make a larger piece of information or message. However, they are two different things in the OSI model. A frame is at layer 2 - the data link layer, meaning there is no such information as IP addresses. Think of this as putting an envelope within an envelope and sending it away. The first envelope will be the packet that you mail, but once it is opened, the envelope within still exists and contains data (this is a frame).
  + This process is called encapsulation which we discussed in room 3: the OSI model. At this stage, it's safe to assume that when we are talking about anything IP addresses, we are talking about packets. When the encapsulating information is stripped away, we're talking about the frame itself.
  + Packets are an efficient way of communicating data across networked devices such as those explained in Task 1. Because this data is exchanged in small pieces, there is less chance of bottlenecking occurring across a network than large messages being sent at once.
  + For example, when loading an image from a website, this image is not sent to your computer as a whole, but rather small pieces where it is reconstructed on your computer. Take the image below as an illustration of this process. The dog's picture is divided into three packets, where it is reconstructed when it reaches the computer to form the final image.
  + Packets have different structures that are dependant upon the type of packet that is being sent. As we'll come on to discuss, networking is full of standards and protocols that act as a set of rules for how the packet is handled on a device. With the Internet predicted to have approximately 50 billion devices connected by the end of 2020, things quickly get out of hand if there is no standardisation.
  + Let's continue with our example of the Internet Protocol. A packet using this protocol will have a set of headers that contain additional pieces of information to the data that is being sent across a network.

Text

Description automatically generated

* TCP/IP
  + This protocol is very similar to the OSI model that we have previously discussed in room three of this module so far. The TCP/IP protocol consists of four layers and is arguably just a summarised version of the OSI model. These layers are:
    - Application
    - Transport
    - Internet
    - Network Interface
  + Very similar to how the OSI model works, information is added to each layer of the TCP model as the piece of data (or packet) traverses it. As you may recall, this process is known as encapsulation - where the reverse of this process is decapsulation.
  + One defining feature of TCP is that it is connection-based, which means that TCP must establish a connection between both a client and a device acting as a server before data is sent.
  + Because of this, TCP guarantees that any data sent will be received on the other end. This process is named the Three-way handshake, which is something we'll come on to discuss shortly. A table comparing the advantages and disadvantages of TCP is located below:

A picture containing table

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated with medium confidence

* + Any sent data is given a random number sequence and is reconstructed using this number sequence and incrementing by 1. Both computers must agree on the same number sequence for data to be sent in the correct order. This order is agreed upon during three steps:
    - SYN - Client: Here's my Initial Number Sequence (ISN) to SYNchronise with (0)
    - SYN/ACK - Server: Here's my Initial Number Sequence (ISN) to SYNchronise with (5,000), and I ACKnowledge your initial number sequence (0)
    - ACK - Client: I ACKnowledge your Initial Number Sequence (ISN) of (5,000), here is some data that is my ISN+1 (5,000 + 1)
  + TCP Closing a Connection
    - Let's quickly explain the process behind TCP closing a connection. First, TCP will close a connection once a device has determined that the other device has successfully received all of the data.
    - Because TCP reserves system resources on a device, it is best practice to close TCP connections as soon as possible.
    - To initiate the closure of a TCP connection, the device will send a "FIN" packet to the other device. Of course, with TCP, the other device will also have to acknowledge this packet.
* UDP/IP
  + The User Datagram Protocol (UDP) is another protocol that is used to communicate data between devices.
  + Unlike its brother TCP, UDP is a stateless protocol that doesn't require a constant connection between the two devices for data to be sent. For example, the Three-way handshake does not occur, nor is there any synchronisation between the two devices.
  + Recall some of the comparisons made about these two protocols in Room 3: "OSI Model". Namely, UDP is used in situations where applications can tolerate data being lost (such as video streaming or voice chat) or in scenarios where an unstable connection is not the end-all. A table comparing the advantages and disadvantages of UDP is located below:

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

* Ports
  + Perhaps aptly titled by their name, ports are an essential point in which data can be exchanged. Think of a harbour and port. Ships wishing to dock at the harbour will have to go to a port compatible with the dimensions and the facilities located on the ship. When the ship lines up, it will connect to a port at the harbour. Take, for instance, that a cruise liner cannot dock at a port made for a fishing vessel and vice versa.
  + These ports enforce what can park and where — if it isn't compatible, it cannot park here. Networking devices also use ports to enforce strict rules when communicating with one another. When a connection has been established (recalling from the OSI model's room), any data sent or received by a device will be sent through these ports. In computing, ports are a numerical value between 0 and 65535 (65,535).
  + Because ports can range from anywhere between 0-65535, there quickly runs the risk of losing track of what application is using what port. A busy harbour is chaos! Thankfully, we associate applications, software and behaviours with a standard set of rules. For example, by enforcing that any web browser data is sent over port 80, software developers can design a web browser such as Google Chrome or Firefox to interpret the data the same way as one another.
  + This means that all web browsers now share one common rule: data is sent over port 80. How the browsers look, feel and easy to use is up to the designer or the user's decision.
  + While the standard rule for web data is port 80, a few other protocols have been allocated a standard rule. Any port that is within 0 and 1024 (1,024) is known as a common port. Let's explore some of these other protocols below:

Table

Description automatically generated

* + We have only briefly covered the more common protocols in cybersecurity. You can find a table of the 1024 common ports listed for more information.
  + What is worth noting here is that these protocols only follow the standards. I.e. you can administer applications that interact with these protocols on a different port other than what is the standard (running a web server on 8080 instead of the 80 standard port). Note, however, applications will presume that the standard is being followed, so you will have to provide a colon (:) along with the port number.

NIS – Cloud Essentials

* Introduction
  + Pre-Cloud
    - colocation centers – datacenter solution
      * host virtual or physical servers
        + Lumen
        + Equinix
      * self-management
  + Cloud Services
    - * GCP
      * AWS
      * Azure
    - pricing models
    - management
* Networking Basics
  + OSI and TCP/IP
* Cloud Firewalls
  + Transport Layer
    - Next-Generation (L4-7)
    - Load Balancers
      * L4 + L7
  + Importance of Network Redundancy
    - * balanced with budget
  + Types
    - Packet Filtering
    - Stateful Inspection
    - Next-generation
  + Host vs Network Based
    - Software vs Hardware
    - Cloud Offerings
* Firewalling
  + pros and cons to solutions
    - Cisco ASA/Firepower
    - Meraki MX
    - FortiGate
    - Palo Alto
    - Cisco
    - pfSense
* Cloud Services
  + - IaaS
    - SaaS
    - PaaS
    - Traditional IT
  + separation of what organization and provider are expected to manage in given environment
    - what can and cannot be tested in a pentesting environment
      * RoE
* Redundancy
  + previously colocation
    - power, cooling, physical security
    - cloud computing, disaster recovery
  + Reputation
    - indicate necessity of compensating controls
      * load balancers, CDNs, cloud gateways, additional servers
      * IaC
  + Load balancers
    - Nginx
    - Hardware or Software
      * Redundancy still important
        + Active/Active
        + Active/Passive
  + CDNs
    - cache content from a website and deliver it to clients from cache
    - speeds up delivery process
* Cloud Types
  + Public
    - AWS
    - Azure
    - GCP
  + Hybrid
  + Private
  + Community – similar risks and compliances
    - reliability
    - security
    - compliance
    - reliability
    - improved services
* Proxy Services
  + Forward Proxy
    - anonymize traffic
    - filter traffic
    - block traffic
  + Reverse Proxy
    - hide server identity
    - reduce load
  + Squid Web Proxy + Privoxy
* Conclusion

Jupyter 101

* Preface
  + Data science is a very broad, enormous topic that isn't (in my opinion) easy to approach and much harder to master. It has a huge variety of uses across all types of industries! To name a few real-world examples of where data science can be found day-to-day:
    - Recommendations of content on Netflix and YouTube based upon your previous viewing history
    - Fraud detection in Banking
    - Intrusion Detection for Cyber Security
    - Weather Forecasting / Prediction
    - Metrics of a Business' sales performance
    - Route planning in Google Maps.
  + It is also goes hand-in-hand with Machine Learning / Artificial Intelligence.
* What is Jupyter
  + Jupyter is a web-based platform often used for data analytics / plotting, machine learning, where code is stored in "Notebooks" whom can imported/exported and shared in many formats such as LaTex, HTML, PDF and many more!
    - <https://oldblog.cmnatic.co.uk/posts/thm-room-jupyter101-support-material/>
    - <https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks>
    - <https://oldblog.cmnatic.co.uk/posts/thm-room-jupyter101-support-material/>
* How Notebooks Run
  + “Cells” act like code interpreters
    - run for the first time 🡪 assigned In[#]
      * Shift+Enter to run code
    - changes In[#]
* Interacting with File System
  + Jupyter directly interacts with the Operating System's filesystem. For example, making files, folders and/or Notebooks.
  + These files are reflected on the Operating System that Jupyter is running on. The directory that you are shown after logging into Jupyter is for all intents and purposes, the "root" directory of Jupyter.
  + However, just because it is the "root" directory, it does not mean it is the Operating Systems /root/ directory. It is simply the directory of wherever Jupyter was launched upon / or told to set to during configuration.
  + Jupyter directly uses Linux' User permissions. So you will only be able to read/write/modify the files that the you (the user the Jupyter Server is running as) within the directory (that has either been specified in configuration, or from where it is launched)
* Handling Data with Pandas
  + fantastic library for data wrangling. It allows us to read data from a wide variety of formats such as CSV files, JSONs, Databases and more!
    - Series and Dataframes
      * read a CSV file with read\_csv
      * display initial rows with .head command
        + .tail command for rear rows
      * .shape command for numerical count of columns and rows in dataset
* Visualizing Data with Matplotlib
  + displayed with .plot()
    - xlabel
    - ylabel
    - zlabel
  + title()
  + color()

Windows x64 Assembly

* Introduction
  + Reverse Engineering on Windows
    - Tools
      * x64dbg, Ghidra, SysInternals
    - Prerequisites
      * CS Concepts
      * C/C++ Exp
      * Assembly
* Number Systems
  + Decimal 🡪 Base10
  + Base7
  + Binary 🡪 Base2
  + Hexadecimal
    - 0-9
    - A-F
  + Prefixes and Suffixes
    - Decimal is represented with the suffix "d" or with nothing. Examples: 12d or 12.
    - Hexadecimal is represented with the prefix "0x" or suffix "h". Examples: 0x12 or 12h. Another way hexadecimal is represented is with the prefix of "\x". However, this is typically used per byte. Two hexadecimal digits make one byte. Examples: \x12 or \x12\x45\x21. If bits and bytes seem a little weird we'll get into them soon so don't worry.
    - Binary is represented with a suffix "b" or with padding of zeros at the start. Examples: 100101b or 00100101. The padding at the start is often used because a decimal number can't start with a zero.
* Bits and Bytes
  + - Bit is one binary digit. Can be 0 or 1.
    - Nibble is 4 bits.
    - Byte is 8 bits.
    - Word is 2 bytes.
    - Double Word (DWORD) is 4 bytes. Twice the size of a word.
    - Quad Word (QWORD) is 8 bytes. Four times the size of a word.
  + Data Type Sizes
    - Char - 1 byte (8 bits).
    - Int - There are 16-bit, 32-bit, and 64-bit integers. When talking about integers, it's usually 32-bit. For signed integers, one bit is used to specify whether the integer is positive or negative.
    - Signed Int
    - 16 bit is -32,768 to 32,767.
    - 32 bit is -2,147,483,648 to 2,147,483,647.
    - 64-bit is -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
    - Unsigned Int - Minimum is zero, maximum is twice that of a signed int (of the same size). For example: unsigned 32-bit int goes from 0 to 4,294,967,295. That is twice the signed int maximum of 2,147,483,647, however, its minimum value is 0. This is due to signed integers using the sign bit, making it unavailable to represent a value.
    - Bool - 1 byte. Interestingly, a bool only needs 1 bit because it's either 1 or 0 but it still takes up a full byte. This is because computers don't tend to work with individual bits due to alignment (talked about later). So instead, they work in chunks such as 1 byte, 2 bytes, 4 bytes, 8 bytes, and so on.
  + <https://www.tutorialspoint.com/cprogramming/c_data_types.htm>
* Binary Operations
  + Four Fundamental Operations
    - NOT, AND, OR, XOR
  + NOT as !
    - NOT 1 = 0
    - NOT 0 =1
  + AND as &
    - 1 AND 1 = 1
    - 1 AND 0 = 0
    - 0 AND 0 = 0
  + OR as |
    - 1 OR 1 = 1
    - 1 OR 0 = 1
    - 0 OR 0 = 0
  + XOR as ^
    - 1 XOR 1 = 0
    - 1 XOR 0 = 1
    - 0 XOR 0 = 0
  + Inverses
    - NAND
    - NOR
* Registers
  + The Registers
    - General Purpose Registers
      * Eight Main GPRs
        + RAX - Known as the accumulator register. Often used to store the return value of a function.
        + RBX - Sometimes known as the base register, not to be confused with the base pointer. Sometimes used as a base pointer for memory access.
        + RDX - Sometimes known as the data register.
        + RCX - Sometimes known as the counter register. Used as a loop counter.
        + RSI - Known as the source index. Used as the source pointer in string operations.
        + RDI - Known as the destination index. Used as the destination pointer in string operations.
        + RSP - The stack pointer. Holds the address of the top of the stack.
        + RBP - The base pointer. Holds the address of the base (bottom) of the stack.
  + The Instruction Pointer
    - RIP is probably the most important register. RIP is the "Instruction Pointer". It is the address of the next line of code to be executed. You cannot directly write into this register, only certain instructions such as ret can influence the instruction pointer.
  + Register Break Downs
    - Each register can be broken down into smaller segments which can be referenced with other register names. RAX is 64 bits, the lower 32 bits can be referenced with EAX, and the lower 16 bits can be referenced with AX. AX is broken down into two 8 bit portions. The high/upper 8 bits of AX can be referenced with AH. The lower 8 bits can be referenced with
* Different Data Types
  + Floating Point Values - Floats and Doubles.
  + Integer Values - Integers, Booleans, Chars, Pointers, etc.
* Extra Registers
  + There are registers r8 to r15 which are designed to be used by integer type values (not floats or doubles). The lower 4 bytes (32 bits), 2 bytes (16 bits), and 8 bits (1 byte) can all be accessed. These can be accessed by appending the letter "d", "w", or "b".
* Instructions
  + - Intel and AT&T
      * Intel for Windows
      * AT&T for Linux
  + Instructions
    - * Before we get started there are three different terms you should know: immediate, register, and memory.
      * An immediate value (or just immediate, sometimes IM) is something like the number 12. An immediate value is not a memory address or register, instead, it's some sort of constant data.
      * A register is referring to something like RAX, RBX, R12, AL, etc.
      * Memory or a memory address refers to a location in memory (a memory address) such as 0x7FFF842B.
    - (Instruction/Opcode/Mnemonic) <Destination Operand>, <Source Operand>
  + Common Instructions
    - Data Movement
      * MOV
      * LEA
      * PUSH
      * POP
    - Arithmetic
      * INC
      * DEC
      * ADD
      * SUB
    - Multiplication and Division
      * MUL
      * DIV
    - Flow Control
      * RET
      * CMP
      * JCC
      * NOP
  + Pointers
    - * ptr is a pointer to num, which means ptr is holding the memory address of num.
      * Then return the sum of what's at the address inside ptr (num which is 10) and 5.
    - Two of the most important things to know when working with pointers and addresses in Assembly are LEA and square brackets.
      * Square Brackets - Square brackets dereference in assembly. For example, [var] is the address pointed to by var. In other words, when using [var] we want to access the memory address that var is holding.
      * LEA - Ignore everything about square brackets when working with LEA. LEA is short for Load Effective Address and it's used for calculating and loading addresses.
    - It's important to note that when working with the LEA instruction, square brackets do not dereference.
  + Zero Extension
  + JMP
    - For unsigned comparisons:
    - JB/JNAE (CF = 1) ; Jump if below/not above or equal
    - JAE/JNB (CF = 0) ; Jump if above or equal/not below
    - JBE/JNA (CF = 1 or ZF = 1) ; Jump if below or equal/not above
    - JA/JNBE (CF = 0 and ZF = 0); Jump if above/not below or equal
    - For signed comparisons:
    - JL/JNGE (SF <> OF) ; Jump if less/not greater or equal
    - JGE/JNL (SF = OF) ; Jump if greater or equal/not less
    - JLE/JNG (ZF = 1 or SF <> OF); Jump if less or equal/not greater
    - JG/JNLE (ZF = 0 and SF = OF); Jump if greater/not less or equal
* Flags
  + Status Flags
    - Zero Flag (ZF) - Set if the result of an operation is zero. Not set if the result of an operation is not zero.
    - Carry Flag (CF) - Set if the last unsigned arithmetic operation carried (addition) or borrowed (subtraction) a bit beyond the register. It's also set when an operation would be negative if it wasn't for the operation being unsigned.
    - Overflow Flag (OF) - Set if a signed arithmetic operation is too big for the register to contain.
    - Sign Flag (SF) - Set if the result of an operation is negative.
    - Adjust/Auxiliary Flag (AF) - Same as the carry flag but for Binary Coded Decimal (BCD) operations.
    - Parity Flag (PF) - Set to 1 if the number of bits set in the last 8 bits is even. (10110100, PF=1; 10110101, PF=0)
    - Trap Flag (TF) - Allows for single-stepping of programs.
  + <https://www.tech-recipes.com/rx/1239/assembly-flags/>
* Calling Conventions
  + Windows x64 Calling Convention
    - * Callee refers to the function being called, and the caller is the function making the call.
  + Fastcall
    - How does the x64 Windows calling convention work?
      * The first four parameters are passed in registers, LEFT to RIGHT. Parameters that are not floating-point values, such as integers, pointers, and chars, will be passed via RCX, RDX, R8, and R9 (in that order). Floating-point parameters will be passed via XMM0, XMM1, XMM2, and XMM3 (in that order).
      * If there is a mix of floating-point and integer values, they will still be passed via the register that corresponds to their position. For example, func(1, 3.14, 6, 6.28) will pass the first parameter through RCX, the second through XMM1, the third through R8, and the last through XMM3.
      * If the parameter being passed is too big to fit in a register then it is passed by reference (a pointer to the data in memory). Parameters can be passed via any sized corresponding register. For example, RCX, ECX, CX, CH, and CL can all be used for the first parameter. Any other parameters are pushed onto the stack, RIGHT to LEFT.
    - Rules
      * The base pointer (RBP) is saved when a function is called so it can be restored.
      * A function's return value is passed via RAX if it's an integer, bool, char, etc., or XMM0 if it's a float or double.
      * Member functions have an implicit first parameter for the "this" pointer. Because it's a pointer and it's the first parameter, it will be passed via RCX. This can be very useful to know.
      * The caller is responsible for allocating space for parameters for the callee. The caller must always allocate space for 4 parameters even if no parameters are passed.
      * The registers RAX, RCX, RDX, R8, R9, R10, R11, and XMM0-XMM5 are considered volatile and must be considered destroyed on function calls.
      * The registers RBX, RBP, RDI, RSI, RSP, R12, R13, R14, R15, and XMM6-XMM15 are considered nonvolatile and should be saved and restored by a function that uses them.
  + Stack Access
    - 1-4 Parameters:
    - Arguments will be pushed via their respective registers, left to right. The compiler will likely use RSP+0x0 to RSP+0x18 for other purposes.
    - More Than 4 Parameters:
    - The first four arguments are passed via registers, left to right, and the rest are pushed onto the stack starting at offset RSP+0x20, right to left. This makes RSP+0x20 the fifth argument and RSP+0x28.
  + Further Exploration
    - <https://docs.microsoft.com/en-us/cpp/build/x64-calling-convention?view=vs-2019>
    - <https://docs.microsoft.com/en-us/cpp/build/x64-software-conventions?view=vs-2019>
  + cdecl (C Declaration)
    - The parameters are passed on the stack backward (right to left).
    - The base pointer (RBP) is saved so it can be restored.
    - The return value is passed via EAX.
    - The caller cleans the stack. This is what makes cdecl cool. Because the caller cleans the stack, cdecl allows for a variable number of parameters.
  + Additional Info
    - <https://docs.microsoft.com/en-us/cpp/build/x64-software-conventions?view=vs-2019>
    - <https://docs.microsoft.com/en-us/cpp/build/x64-calling-convention?view=vs-2019>
    - <https://docs.microsoft.com/en-us/cpp/build/prolog-and-epilog?view=vs-2019>
    - <https://www.gamasutra.com/view/news/171088/x64_ABI_Intro_to_the_Windows_x64_calling_convention.php>
* Memory Layout
  + Memory Segments
    - Stack - Holds non-static local variables. Discussed more in-depth soon.
    - Heap - Contains dynamically allocated data that can be uninitialized at first.
    - .data - Contains global and static data initialized to a non-zero value.
    - .bss - Contains global and static data that is uninitialized or initialized to zero.
    - .text - Contains the code of the program (don't blame me for the name, I didn't make it).
  + Overview of Memory Sections
    - Stack - Area in memory that can be used quickly for static data allocation. Imagine the stack with low addresses at the top and high addresses at the bottom. This is identical to a normal numerical list. Data is read and written as "last-in-first-out" (LIFO). The LIFO structure of the stack is often represented with a stack of plates. You can't simply take out the third plate from the top, you have to take off one plate at a time to get to it. You can only access the piece of data that's on the top of the stack, so to access other data you need to move what's on top out of the way. When I said that the stack holds static data I'm referring to data that has a known length such as an integer. The size of an integer is defined at compile-time, the size is typically 4 bytes, so we can throw that on the stack. Unless a maximum length is specified, user input should be stored on the heap because the data has a variable size. However, the address/location of the input will probably be stored on the stack for future reference. When you put data on top of the stack you push it onto the stack. When data is pushed onto the stack, the stack grows up, towards lower memory addresses. When you remove a piece of data off the top of the stack you pop it off the stack. When data is popped off the stack, the stack shrinks down, towards higher addresses. That all may seem odd but remember, it's like a normal numerical list where 1, the lower number, is at the top. 10, the higher number, is at the bottom. Two registers are used to keep track of the stack. The stack pointer (RSP/ESP/SP) is used to keep track of the top of the stack and the base pointer (RBP/EBP/BP) is used to keep track of the base/bottom of the stack. This means that when data is pushed onto the stack, the stack pointer is increased since the stack grew towards higher addresses. The base pointer has no reason to change when we push or pop something to/from the stack. We'll talk about both the stack pointer and base pointer more as time goes on.
    - Heap - Similar to the stack but used for dynamic allocation and it's a little slower to access. The heap is typically used for data that is dynamic (changing or unpredictable). Things such as structures and user input might be stored on the heap. If the size of the data isn't known at compile-time, it's usually stored on the heap. When you add data to the heap it grows towards higher addresses.
    - Program Image - This is the program/executable loaded into memory. On Windows, this is typically a Portable Executable (PE).
    - TEB - The Thread Environment Block (TEB) stores information about the currently running thread(s).
    - PEB - The Process Environment Block (PEB) stores information about the process and the loaded modules. One piece of information the PEB contains is "BeingDebugged" which can be used to determine if the current process is being debugged.
      * <https://docs.microsoft.com/en-us/windows/win32/api/winternl/ns-winternl-peb>
  + Stack Frames
  + Endianness
    - Big Endian - The most significant byte (far left) is stored first. This would be 0xDEADBEEF from the example.
    - Little Endian - The least significant byte (far right) is stored first. This would be 0xEFBEADDE from the example.
    - <https://www.youtube.com/watch?v=NcaiHcBvDR4>
  + Data Storage
    - Data positions are referenced by how far away they are from the address of the first byte of data, known as the base address (or just the address), of the variable. The distance of a piece from its base address is considered the offset. For example, let's say we have some data, 12345678. Just to push the point, let's also say each number is 2 bytes. With this information, 1 is at offset 0x0, 2 is at offset 0x2, 3 is at offset 0x4, 4 is at offset 0x6, and so on.
    - Variables are allocated on the stack one on top of the other like a stack of trays. This means they're put on the stack starting from higher addresses and going to lower addresses.
    - Data is put into the variables from left to right, top to bottom. That is, from lower to higher addresses.
    - It's a simple concept, try not to over-complicate it just because I've given a long explanation. It's vital you understand it, which is why I've taken so much time to explain this concept. It's because of these concepts that there are so many depictions of memory out there that go in different directions.
  + RBP & RSP on x64
* Final Thoughts
  + <https://software.intel.com/content/www/us/en/develop/articles/intel-sdm.html>
  + <https://developer.amd.com/resources/developer-guides-manuals/>
  + <https://docs.microsoft.com/en-us/>
  + <https://github.com/ReversingID/Awesome-Reversing>

Principles of Security

* Introduction
  + Defense in Depth
* CIA Triad
  + Confidentiality
    - protection of data from unauthorized access and misuse
  + Integrity
    - information is kept accurate and consistent unless authorized changes are made
  + Availability
    - available and accessible by the user
    - 99.999% uptime
* Principles of Privileges
  + administrate and define various levels of access
    - by role/function
    - by sensitivity of information stored on the system
  + Privileged Identity Management
    - translate role into access role
  + Privileged Access Management
    - management of privileges an access role has
  + Principle of Least Privilege
* Security Models
  + Bell-La Padula Model
    - * used to achieve confidentiality
    - grants access to pieces of data (objects) on a strictly need to know basis
      * no write down, no read up
    - popular in govt and military
      * assume member of org have already gone through vetting process
  + Biba Model
    - * similar to Bell-La Padula for integrity
    - applies rules to objects (data) and subjects (users)
      * no write up, no read down
        + rule means that subject can create or write content to objects at or below their level but can only read the contents of objects above the subjects level
    - used when integrity is more important than confidentiality
* Threat Modeling and Incident Response
  + reviewing, improving, and testing security protocols in place in an organizations infotech infrastructure and services
    - identify likely threats that an application or system may face
  + principles
    - preparation
    - identification
    - mitigation
    - review
  + an effective threat model includes
    - threat intelligence
    - asset identification
    - mitigation capabilities
    - risk assessment
  + Frameworks
    - STRIDE
      * Spoofing, identity, Tampering with data, Repudiation threats, Information disclosure, Denial of Service and Elevation of privileges
    - PASTA
      * Process for Attack Simulation and Threat Analysis
  + CSIRT Teams

Protocols and Servers

* Introduction
  + HTTP
  + FTP
  + POP3
  + SMTP
  + IMAP
* Telnet
  + Port 23
* HTTP
  + telnet IP 80
    - GET /index.html HTTP/1.1
    - Host: telnet
  + Determine webserver and clients
  + Browser information
* FTP
  + telnet IP 21
    - USER <username>
    - PASS <password>
    - SYST
    - PASV
    - STAT
  + ftp IP
    - frank
    - password
    - ls
    - ascii
    - get
    - put
  + vsftpd
  + ProFTPD
  + uFTP
  + FileZilla
* SMTP
  + - Components
      * + MSA
        + MTA
        + MDA
        + MUA
      * A Mail User Agent (MUA), or simply an email client, has an email message to be sent. The MUA connects to a Mail Submission Agent (MSA) to send its message.
      * The MSA receives the message, checks for any errors before transferring it to the Mail Transfer Agent (MTA) server, commonly hosted on the same server.
      * The MTA will send the email message to the MTA of the recipient. The MTA can also function as a Mail Submission Agent (MSA).
      * A typical setup would have the MTA server also functioning as a Mail Delivery Agent (MDA).
      * The recipient will collect its email from the MDA using their email client.
    - SMTP
    - POP3/IMAP
  + telnet IP 25
    - mail from:
    - rcpt to:
    - data
* POP3
  + telnet IP 110
    - USER <user>
    - PASS <pass>
    - STAT
    - LIST
    - RETR <#>
    - QUIT
* IMAP
  + telnet IP 143
  + LOGIN <user> <pass>
  + LIST “” “\*”
  + EXAMINE INBOX
* Summary

| **Protocol** | **TCP Port** | **Application(s)** | **Data Security** |
| --- | --- | --- | --- |
| FTP | 21 | File Transfer | Cleartext |
| HTTP | 80 | Worldwide Web | Cleartext |
| IMAP | 143 | Email (MDA) | Cleartext |
| POP3 | 110 | Email (MDA) | Cleartext |
| SMTP | 25 | Email (MTA) | Cleartext |
| Telnet | 23 | Remote Access | Cleartext |

Protocols and Servers 2

* Introduction

Servers implementing these protocols are subject to different kinds of attacks. To name a few, consider:

1. Sniffing Attack (Network Packet Capture) – violates confidentiality
2. Man-in-the-Middle (MITM) Attack – violates integrity
3. Password Attack (Authentication Attack) – leads to disclosure
4. Vulnerabilities

* Sniffing Attack
  + Tcpdump
    - Sudo tcpdump port 110 -A
      * Use pop as well
      * Sudo tcpdump port 23
  + Wireshark
  + Tshark
* MitM Attack
  + Easy to carry out if the two parties do not confirm the authenticity and integrity of each message
    - No secure authentication or integrity checking
  + Ettercap and Bettercap
    - Can also effect other cleartext protocols
      * FTP, SMTP, POP3
  + Counter with PKI, trusted root, TLS
* TLS
  + Upgrade HTTP, FTP, SMTP, POP3, IMAP

| **Protocol** | **Default Port** | **Secured Protocol** | **Default Port with TLS** |
| --- | --- | --- | --- |
| HTTP | 80 | HTTPS | 443 |
| FTP | 21 | FTPS | 990 |
| SMTP | 25 | SMTPS | 465 |
| POP3 | 110 | POP3S | 995 |
| IMAP | 143 | IMAPS | 993 |

1. The client sends a ClientHello to the server to indicate its capabilities, such as supported algorithms.
2. The server responds with a ServerHello, indicating the selected connection parameters. The server provides its certificate if server authentication is required. The certificate is a digital file to identify itself; it is usually digitally signed by a third party. Moreover, it might send additional information necessary to generate the master key, in its ServerKeyExchange message, before sending the ServerHelloDone message to indicate that it is done with the negotiation.
3. The client responds with a ClientKeyExchange, which contains additional information required to generate the master key. Furthermore, it switches to use encryption and informs the server using the ChangeCipherSpec message.
4. The server switches to use encryption as well and informs the client in the ChangeCipherSpec message.

* SSH
  + Benefits
    - Can confirm server ID
    - Exchanged messages are encrypted
    - Both sides can detect modification
  + Manually check SSH server public key to avoid MitM
  + Can use SCP to transfer files
  + Secure FTP using SSH
    - SFTP port 22
* Password Attack
  + - Password guessing
    - Dictionary attack
    - Brute force attack
  + Hydra -l username -P wordlist.txt server service
    - -s PORT to specify a non-default port for the service in question.
    - -V or -vV, for verbose, makes Hydra show the username and password combinations that are being tried. This verbosity is very convenient to see the progress, especially if you are still not confident of your command-line syntax.
    - -t n where n is the number of parallel connections to the target. -t 16 will create 16 threads used to connect to the target.
    - -d, for debugging, to get more detailed information about what’s going on. The debugging output can save you much frustration; for instance, if Hydra tries to connect to a closed port and timing out, -d will reveal this right away.
  + Counter with
    - Password policy
    - Account lockout
    - Throttling authentication attempts
    - CAPTCHAs
    - Require use of public cert authentication
    - 2FA
* Summary

| **Protocol** | **TCP Port** | **Application(s)** | **Data Security** |
| --- | --- | --- | --- |
| FTP | 21 | File Transfer | Cleartext |
| FTPS | 990 | File Transfer | Encrypted |
| HTTP | 80 | Worldwide Web | Cleartext |
| HTTPS | 443 | Worldwide Web | Encrypted |
| IMAP | 143 | Email (MDA) | Cleartext |
| IMAPS | 993 | Email (MDA) | Encrypted |
| POP3 | 110 | Email (MDA) | Cleartext |
| POP3S | 995 | Email (MDA) | Encrypted |
| SFTP | 22 | File Transfer | Encrypted |
| SSH | 22 | Remote Access and File Transfer | Encrypted |
| SMTP | 25 | Email (MTA) | Cleartext |
| SMTPS | 465 | Email (MTA) | Encrypted |
| Telnet | 23 | Remote Access | Cleartext |

* Hydra

| **Option** | **Explanation** |
| --- | --- |
| -l username | Provide the login name |
| -P WordList.txt | Specify the password list to use |
| server service | Set the server address and service to attack |
| -s PORT | Use in case of non-default service port number |
| -V or -vV | Show the username and password combinations being tried |
| -d | Display debugging output if the verbose output is not helping |