Network Security

Brief networking overview

Computer networking is a vast subject. Here we will cover a few of the basic concepts needed for security. People often break down the various networking functions into 7 layers in something called the OSI model. Here are the layers and their names.

7. Application layer
6. Presentation layer
5. Session layer
4. Transport layer
3. Network layer
2. Data link layer
1. Physical layer

Layer 7 contains the networked applications we use all the time, like HTTP, email, and SSH. Layers 6 and 5 are not important. Layer 4 contains the *Transmission Control Protocol* (TCP) and the *User Datagram Protocol* (UDP). Layer 3 contains the *Internet Protocol* (IP). Layer 2 contains protocols like Ethernet and Wifi. Layer 1 won't feature much in this class.

Each layer features several *protocols*, which are standardized means of exchanging information. It's important that messages follow the same structure so that there's no chance for one side of the communication to misunderstand what the other is saying. Each message sent in a particular protocol contains data and a *header* that contains information to help the other side process the message. Think of headers as playing a similar role to what envelopes play in postal mail.

IP addresses At layer 3, all machines connected to the internet have an IP address. These come in two flavors: IPv4 and IPv6. IPv4 addresses are written in dotted decimal notation consisting of four blocks of digits from 0 to 255. Some examples include 127.0.0.1 and 192.168.1.2. There are about 4 billion possible IPv4 addresses, though because of the way they have been divvied up, there are considerably fewer that can actually be used. To address the lack of IPv4 addresses, IPv6 was developed. These addresses are 128 bits in size, which gives a huge number of potential addresses. IPv6 addresses are usually written in 8 groups of 4 hex digits. An example is fe80::d0c9:1638:64a0:39ae. The double colon indicates several blocks of 0000 that have been omitted to keep the notation shorter. The majority of the internet still uses IPv4 over IPv6, and IPv4 will be our focus here.

In network security, a few IPv4 address ranges are good to know. The ranges 10.0.0.0 – 10.255.255.255, 192.168.0.0 – 192.168.255.255, and 172.16.0.0 – 172.31.255.255 are private addresses. These ranges are often used on local networks. Millions of different businesses and households simultaneously use these ranges for their own private networks. Those addresses only have meanings on those local networks and packets with those addresses are never forwarded out of the network onto the main internet.

Another useful address range is 127.0.0.0 – 127.255.255.255. Any address in this range is called a *loopback address*. These are used for testing and running network applications on your own computer. These addresses always "loop back" to the computer itself and never go out to any network. The name *localhost* usually translates into a loopback address.

TCP The Internet Protocol's main purpose is routing packets across the internet, making sure things can get from a source to a destination, possibly one halfway across the world. Packets sometimes don't make it to their destinations due to network congestion or equipment malfunctions. One of the TCP's main functions is to add some reliability. It does this by breaking up the data to be sent into pieces called *segments* and attaching a *sequence number* to each segment. That sequence number is used by the sender and receiver to arrange for lost segments to be resent.

One of the most important parts of TCP is its *three-way handshake* that is used to establish a connection. The shorthand for it is SYN, SYN-ACK, ACK. What does this mean? The side that wants to start the connection sends a TCP message with a the SYN flag in the TCP header set to 1. SYN is short for “synchronize sequence numbers.” Along with the SYN flag, the side starting the connection also sends along what value they are using as their starting sequence number. For reasons we will soon see, we don’t want to use an initial sequence number of 0. In response to this message, the other side of the connection sends a SYN-ACK reply. In this reply, this side shares their own initial sequence number. Here ACK is short for “acknowledgement.” The SYN-ACK reply confirms that the first message was received and includes the other side’s starting sequence number. The last part involves the first side sending an ACK, which acknowledges that they got the SYN-ACK reply. A three-way handshake of this sort is necessary in order for both sides to be confident they are connected.

In addition to the SYN and ACK flags, there are a few more flags in the TCP header. For our purposes, the most important is the RST (reset) flag. It can be used to immediately end a connection. Also, whenever a machine running TCP gets a TCP message they are not expecting (such as a SYN-ACK when they didn’t initiate a connection with a SYN), then they will send a message with the RST flag sent to tell the other side to stop the connection.

UDP Besides TCP, the other important protocol at the transport layer is UDP. UDP is basically what you use if you don’t care about any of the reliability features of TCP. It does the bare minimum of what a transport layer protocol needs to do. A good way to remember this is to think of the U in UDP is as standing for “unreliable.” The reliability of TCP comes with a cost. TCP is slower, and if a packet is lost, TCP can hold up everything until the missing packet is resent and arrives. UDP, on the other hand, is sometimes called “fire and forget.” It’s what you use if you care about speed and can handle an occasional lost packet. A key application of UDP is streaming video.

Port numbers One feature provided by both TCP and UDP is *port numbers*. These are numbers in the range from 0 to 65,535. When traffic arrives at its destination, the networking software needs to know what application to send it to. Does it go to the web server, the email server, or SSH? Layer 4 protocols use port numbers for this purpose. Certain port numbers are usually reserved for specific services. Here is a list of a few of the more important ones to know.

22	SSH
53	DNS
80	HTTP
443	HTTPs

There are many more port numbers reserved for various services. These reserved port numbers are more suggestions than requirements. You can run a server that listens for SSH connections on port 80 (the HTTP port), though that would generally be confusing.

When you download a web page, your web browser will usually make a request to a web server running at port 80 or 443. That value will be the destination port. The source port will usually be a random port number greater than 1023. This source port is part of how the networking software on your computer will identify the response that comes back from the web server.

Attacks using TCP

Now that we know a little about how layers 3 and 4 work, let’s look at some common attacks that take advantage of the protocols at these layers.

SYN floods A SYN flood is a simple type of denial of service attack that involves an attacker sending a large amount of SYNs to a machine in an effort to either crash it or overwhelm it with so much traffic that legitimate users can't access it. Remember that a TCP SYN message is used when one side wants to initiate a TCP connection. Once a machine receives a SYN, it will send a SYN-ACK reply and set aside some resources for the connection. It will wait some time, possibly up to a few minutes, for the final ACK reply to come back. In a SYN flood, an attacker never sends back those replies. If enough SYNs are sent, all of the server's resources for accepting connections will be used up.

One important part of a SYN flood is for the attacker to *spoof* their IP address. That is, instead of using their own IP address as the source address, they make up a bogus IP address. There is free software out there that allows you to set all the individual parts of the packet to whatever values you want, including the source IP address. It's important that these bogus IP addresses not be addresses of actual machines on the internet that are listening for TCP traffic. Because if they are listening, then when they get the SYN-ACK send from the attacker's target, they will send an RST. Those machines did not initiate the connection and were not expecting a SYN-ACK, so the TCP specification says they should send back RSTs. The RSTs would tell the target server to stop the connection, which would free up the resources the attacker is trying to use up. Spoofing the IP address also makes it so that the target will not know who is attacking them.

On defense against SYN floods is to shorten the time the server waits for a SYN-ACK before dropping the connection. But if you shorten it too much, then real users on slow connections may not be able to access the server. Another defense is something called *SYN cookies*. Basically, when a SYN comes in, instead of reserving resources for a new connection, the machine will encode the connection information in the sequence number and then forget about the connection. When the final ACK comes back, they can use the information in the returning sequence number to reconstruct the initial connection information.

TCP reset attack The *TCP reset attack* is a simple attack by which a TCP connection can be terminated. An attacker observes a legitimate TCP connection and spoofs some packets with the RST flag set and makes them look like they are coming from one side of the connection. Remember that it's possible to craft a packet with whatever details you want it to have. One trick to this is that the sequence number of the packet being sent needs to look like it belongs in the connection. If the current sequence number is 1,000,000,000 and the attacker sends a packet with sequence number 123,456, the target will discard it as not relevant to the connection. The attacker doesn't need to exactly guess the sequence number; they just need to get it within a certain window. They could do this by randomly trying sequence numbers until something works, though this could take a lot of tries. If they are sniffing traffic on a local network, then they can just look at the actual sequence numbers and their job is much easier. The Linux utility *tcpkill* can be used to perform the TCP reset attack. Like many things, it has legitimate and nefarious uses. One legitimate use would be if you want to stop a TCP connection that is chewing up a lot of bandwidth on your machine and you don't know what program is actually using it.

TCP session hijacking The TCP reset attack involves an attacker injecting bogus TCP messages into a TCP session. It's also possible to inject data into a session. Just like with the TCP reset attack, the attacker needs to get the sequence number right in order for their data to be accepted as part of the connection. On a local network, this can be done by sniffing traffic, but it's considerably harder on a remote network.

One particularly famous instance of this is the Mitnick-Shimomura attack from the mid 1990s by hacker Kevin Mitnick on security researcher Tsutomu Shimomura. Mitnick wanted access to Shimomura's computer. Mitnick had earlier hacked into a site called toad.com that Shimomura was also using. Using the old Unix finger command on that site, he found that Shimomura's computer had a "trusted relationship" with a particular server. By "trusted relationship", we mean Shimomura's computer accepted terminal commands coming from the IP address of the server.

So Mitnick initiated a TCP connection with Shimomura's computer by sending a SYN message with the IP address spoofed to be that of the server. The first wrinkle with this plan is that Shimomura's computer would send a SYN-ACK response and that response would go to the server. The server would not be expecting that reply and would send back an RST, ending the connection attempt. So before sending the SYN, Mitnick performed a SYN flood on the server to make it unresponsive so that it wouldn't be able to send the RST.

Next, Mitnick would have to craft the third part of the three-way handshake, the final ACK reply. In order for that reply to be accepted, Mitnick would have to know the sequence number of the SYN-ACK that Shimomura's computer replied with. That reply went to the server, so Mitnick could not see it. He was not on the same network as Shimomura, so sniffing traffic was not an option, and trying to guess it randomly would take forever. However, back in the mid 1990s, most TCP systems used predictable initial sequence numbers. So before the attack, Mitnick had sent a few SYNs to directly to Shimomura's computer and observed the sequence numbers in the replies were always increasing by 12800. With this information, he was able to create an ACK that was accepted. The connection was now established.

At this point, Mitnick could send commands to Shimomura's computer. However the results of those commands would go the server, not to Mitnick, so he couldn't see what he was doing. But this was good enough. He sent commands that opened up a remote shell that could be accessed from anywhere. Then he used that to log on and have a full interactive session where he was able to find the things on Shimomura's computer that he was looking for.

As a result of this and other problems, people decided that TCP initial sequence numbers should be randomized. That would make the attack described above much more difficult. However, it's not impossible since some TCP implementations use random number generators that are weak in that observing a few values can allow you to predict future values.

Port scanning

Port scanning is a technique where an attacker tries to make connections on a variety of different port numbers on a target to see which ports are "open." By open, we mean that the machine is listening for traffic on those ports. Once the attacker knows what ports are open, they can try to see what services are running on those ports. If those services have vulnerabilities, either because they are old or just insecure, then the attacker can craft data to send to those services to exploit those vulnerabilities. In the simplest case, the user can use a tool like Metasploit. That tool has a database of services, what their vulnerabilities are, and prewritten scripts that exploit those vulnerabilities. More sophisticated attackers can write their own code to exploit vulnerabilities. The goal is often to get a remote shell on the system. Once you have that, you can execute commands on the system and try to find other exploits that could elevate you to a root user with all administrative privileges on the system.

We will focus on how the port scan works. The attacker sends messages to the target and uses the replies to determine if the port is open or closed. The most popular tool for this is called nmap.

One type of scan is the TCP connect scan. With this command you attempt to make a full TCP connection with the target. If the connection succeeds, then you know the port is open, and if it doesn't, then the port is likely closed. This is a little slow, and your IP address will likely be logged by the target. But it might be the only option available if you don't have administrative privileges on the system you are running it from.

A better scan is the SYN scan. With this, instead of making a full connection, you just send a SYN. If you get back a SYN-ACK, then you know the port is open, and you send an RST to stop the three-way handshake and to prevent accidentally hitting the target with a SYN flood since you will likely be scanning many ports. If the target port is closed, the target will send back an RST. However, it's also possible that you don't get any reply at all. This could be due to a firewall blocking connection attempts at the port number. You might also get back an ICMP (Internet Control Message Protocol) error message saying the port is unreachable. In both of these situations, nmap will mark the port as "filtered."

UDP scanning is more tricky because UDP has no flags to work with. If you send a message to a UDP port and get back a reply from the service at the port, then it's clearly an open port. But if you get back no reply, it could be that the port is closed, but it could be that it's open and you just didn't send data in a format that it understood. Or it could be that the data you sent was accepted, but the service doesn't send replies. Sometimes if a UDP port is closed, a system will send back an ICMP error message indicating as much, but network administrators often configure their systems to not send these messages back.

Protecting servers If you're in charge of a server on the internet, then you will find your system being scanned all the time by random attackers. One of the first things to do is a scan of yourself to see if there are any ports open that you aren't expecting. Sometimes things you install may open up ports without you realizing it, or people that use your system may open up ports and forget to close them. Next, limit the ports that are open to only the services you absolutely need. The less ports are open, the less chances there are for an attacker to find a vulnerability, the better.

People sometimes move services like HTTP or SSH to unusual port numbers. This can cut down on the number of attacks they get. A lot of attackers run scripts that just check the standard ports for these services. So moving things to weird-numbered ports will cut down on these types of attacks, but it won't hide it from attackers doing full port scans. This is sometimes called "security by obscurity." By itself, it's not secure, but used in concert with other techniques, it does aid security. It's a little like removing the front door to your house and putting the only entry somewhere in the back hidden by some bushes. It will keep some ordinary intruders out, but not determined ones.