Network Security, Part III

Network Address Translation

Network Address Translation (NAT) is a technology that allows multiple devices on a local network to share the same global IP address. Usually the network's gateway router will do the network address translation. That router has both an internal network address and a global address that the local network shares. When a machine on the local network needs to access a site on the internet, it sends a request that passes through the router. The router will rewrite the IP address on that packet with its own global address. It may also change the port number on the packet. It records all this information in a table, particularly the port number. When a response comes back from the remote site, the router looks up the incoming packet's destination port in its table to know which machine on the internal network to forward the response to.

NAT makes it difficult for an external site to start a connection with someone inside the network. Entries in the NAT router's table are made when someone inside the network tries to contact someone outside it. When someone outside tries to contact somebody inside the network, that packet will be dropped by the NAT router because there won't be a table entry there for the packet, and the NAT router won't know whom to forward the packet to. This makes certain networking things more difficult, like running a web server at home or playing networked games. It also adds a certain degree of security since attackers can't easily send unsolicited packets to devices inside the network. It also makes it hard for an attacker to get a map of what the internal network looks like. NAT does not provide perfect security, however, as information can still get through if the attacker picks a port that does correspond to an entry in the table.

Virtual Private Networks

A *virtual private network* (VPN) is a way to securely allow remote users to access a local network. For instance, in our Cyber lab, we have some servers that are only accessible on campus. They don't have global IP addresses. However, a VPN allows access to them from off campus.

In order for this to work, the network needs a machine that runs VPN software. Remote users who want to connect to the network will send their traffic to that VPN machine, and that machine will then send the traffic onto the network. Suppose a remote user wants to access a database on the private network that is at IP 192.168.4.47. It creates a packet with destination 192.168.4.47 and it puts that packet into another packet whose destination is the VPN machine. This is sent over an encrypted connection to the VPN machine, which pulls out the packet destined for the database and sends it through the local network. This is an example of tunneling, where we put one piece of network traffic inside another. The encrypted connection to the VPN machine can be done with a variety of technologies. One possibility is TLS. Another is IPsec, which is a technology that encrypts data at the network layer. A more recent technology is WireGuard.

With a VPN connection, it's like your computer is on the local network, even though you're remote. VPNs are used for this purpose and also for privacy on the internet, specifically to hide your IP. There are various VPN services that you can use for this. The VPN service's IP address is what the sites your accessing will see, not your IP address. This keeps the sites from seeing your IP, though the VPN service will still see it, so you have to trust the VPN service. This can also be used to get around sites that restrict access based on IP address. For instance, if you want to access a site that requires you to be in Australia, you can use a VPN service in Australia.

Firewalls

In the real world, a firewall is a device meant to keep fire out. For instance, cars have firewalls to separate the hot engine from the rest of the car. In a network, a *firewall* is used to keep certain traffic from entering a network. It can also be used to prevent things from leaving the network. Firewalls can be programs running on a computer or they could be a physical device.

Firewalls often operate by *blacklists* and *whitelists*. A blacklist is a list of "bad" things to be stopped. Anything not on the list is allowed through. A whitelist is a list of "good" things to allow. Anything not on the list is blocked. On the one hand, whitelists are more secure, as it's often hard to think of all possible "bad" things to put on a blacklist, and things often change rapidly. On the other hand, sometimes whitelists are so restrictive that they make it hard for legitimate users to get their work done. If the whitelist is too restrictive, those users will look for ways around the whitelist, which might end up making things more insecure.

We will look at three types of firewalls: *stateless packet filters*, *stateful packet filters*, and *application layer firewalls*. Packet filters look at packets and make decisions based on information in the packet headers. This includes things like IP addresses, port numbers, TCP flags, and the type of traffic (TCP, UDP, ICMP, etc.). For instance, you could set a packet filter to reject all traffic except to ports 80 and 443 or to allow all outgoing traffic except ICMP. A nice packet filter for Linux is iptables.

A stateless packet filter looks at each packet in isolation, not considering it in relation to anything else. A stateful packet filter goes further by remembering connection information. For instance, suppose we want to prevent anyone from starting a connection with computers on a network, but we want to allow the computers on the network to make connections with the outside world, say to download web pages. We can set the firewall to reject all incoming packets with the SYN flag set. But how do we allow stuff to come into the network that is associated with a connection started by someone inside? With a stateless packet filter, we would have to do this by allowing in any packet that has the ACK flag set. This has the drawback that we will allow in ACKs even if they are not associated with a real connection. A stateful packet filter, on the other hand, would be able to handle this. When someone on the inside starts a connection with a remote website, the packet filter will see the outgoing SYN and record that as a connection in its table. Now it will allow in ACKs that are associated with that connection, but it can block other ACKs.

Packet filters mostly work at layers 4 and below, inspecting info in packet headers. An application layer firewall works at layer 7, actually inspecting the data of a packet. Sometimes this is referred as *deep packet inspection*. An application layer firewall can do more sophisticated things than a packet filter, like blocking all outgoing FTP connections or blocking all packets that contain "stackoverflow.com". HTTPs can make an application layer firewall's job harder since the data is encrypted. Sometimes the firewall will man-in-the-middle (MITM) the connection. Instead of an encrypted connection directly with the remote website, a user will have an encrypted connection with the firewall and the firewall will have an encrypted connection with the website.

Besides firewalls, there are *intrusion detection systems* (IDS) and *intrusion prevention systems* (IPS). An IDS detects potential threats but doesn't do anything to stop them, while an IPS will try to stop them. The IDS will usually log suspicious events and it may send a notification to network administrators. Both types usually work via either *signature-based* or *anomaly-based* detection. With signature-based detection, we're looking for signatures of certain types of attacks, some special characteristic that identifies them. For instance, if we see packets from the same IP address with port 1, 2, 3, 4, etc. all in a short time, that is a signature for a port scan. In anomaly-based detection, we observe ordinary traffic for a while to build up a baseline of what normal traffic looks like. Then we look for anomalies, things that are out of the ordinary.

These systems are usually tunable, where we can specify how aggressive we want them to be in looking for malicious traffic. However, the more aggressive we make them, the more false positives we will get, where innocent traffic is flagged as being malicious. For instance, let's say a network has 100,000 things pass through it in an hour, and 5 of them are malicious. If we tune our system to be able to catch all 5, maybe it is at the expense of a 1% false positive rate. This would mean 1% of 99,995 pieces of innocent traffic would be flagged, amounting to about 1000 false positives. In an IDS, network administrators would have to weed through 1005 potential threats to find the 5 real threats. In an IPS, 1000 pieces of innocent traffic would be blocked, leading to lots of complaints to the networking department.

Wi-Fi security

Wi-Fi is the trade name for a wireless communication technology that is defined by the IEEE 802.11 standard. In Wi-Fi, data is sent over the airwaves in a similar way to how radio works. Wi-Fi typically works in the 2.4 GHz and 5 GHz bands. For contrast, FM radio is in roughly the 88 to 108 MHz range and AM radio is roughly in the 500 to 1700 KHz range. Signals in the Wi-Fi range can typically travel up to a few hundred feet at ordinary power levels. You can use higher power levels, but that might invite a visit from the FCC. Radio communication of all kinds, including Wi-Fi, is highly regulated.

Since Wi-Fi is broadcast over the air, anything you send can be picked up by anyone sniffing network traffic. You can use Wireshark to pick up Wi-Fi traffic if you put your network card into *monitor mode*. This is easier said than done, especially since many network cards can't do that on Windows.

The 2.4 GHz band is broken up into 13 *channels*. Again, these are a little like radio or even TV channels. The channels are numbered 1 to 13. Because of interference, a Wi-Fi network will not use channels that are too close to each other. It's typical for a network to use just channels 1, 6, and 11. The 5 GHz band is also broken into channels. There are more of them, specifically: 36, 40, 44, . . . , 64; 100, 104, 108, . . . , 144; and 149, 153, 157, 161, 165.

There are various Wi-Fi standards, including 802.11b, 802.11g, 802.11n, and 802.11ac. Each new standard generally offers better speeds. The most common right now are 802.11n and 802.11ac.

People connect to a Wi-Fi network by connecting to an *access point* (AP). That AP is usually connected to some wired device. In a home network this might be a cable or DSL modem. Wi-Fi networks are given a name (like MSMWireless or MSMGuess). This is its *service set identifier* (SSID). It's possible to turn off this feature. This will hide the network from some operating system displays, but anyone with the right software can see all the networks around. The access point will also have a BSSID, where the B is for "base". This is the AP's MAC address.

At the Wi-Fi level, instead of referring to chunks of traffic as packets, the term *frames* is used. Around every second or so, or possibly less, an AP sends out a *beacon frame*. This contains info about the AP and network such as the SSID, MAC address, security, and other things. When a device wants to connect to the network, it sends an *association frame* to the network. If an AP wants to boot a device from a network, it can send a *deauthentication frame* to the device. These are easy to spoof, making it easy for people to kick others off a network.

Recall from earlier in the class that people can create an *evil twin AP* that looks just like the real one. They get all the info to look like the real one from beacon frames. After they program their own AP with that info, they spoof deauthentication frames to kick users off the real network. They make sure their AP has a stronger signal than the real one, and the deauthenticated users will reconnect to the evil twin.

Wi-Fi encryption

There are four types of security: WEP, WPA, WPA2, and WPA3. WEP (wired equivalent privacy) is the oldest and weakest type. It was designed without input from cryptographers, which led to some easily exploited critical vulnerabilities. The WEP key can usually be figured out by observing traffic for just a few minutes and using freely available software. Though it's old, there are still many routers that use WEP. These are often home network routers where people plugged them in years ago and haven't bothered changing. Below we will look at three different problems with WEP.

Problem 1 WEP uses the RC4 stream cipher. RC4 is not a terrible cipher if used right, but the problem is that WEP didn't use it right. Recall that one of the worst things you can do with a stream cipher is to reuse the same keystream with different plaintexts. To avoid this, WEP uses a 24-bit initialization vector (IV) combined with the WEP key to seed the RC4 pseudorandom number generator. A 24-bit IV gives $2^{24} \approx 16$ million possible IVs, which seems like it would be enough.

However, there are a couple of issues. One is the birthday problem. The IV is random, and we would expect repeated IVs to start occurring after only about $\sqrt{2^{24}} = 2^{12} = 4096$ frames, which is can happen in minutes or less on a typical network. Worse, RC4 has biases in its keystream. That is, certain values are more likely to appear than others. It's a slight bias, but that bias combined with the fact that certain parts of the plaintext, namely the headers, are very predictable, leads to complete destruction. Using some statistics, a few minutes worth of traffic gives enough data to take advantage of the bias to break the key.

Problem 2 Another problem is the *chop-chop attack*. It takes advantage of the WEP checksum. A checksum is a calculation that both the sender and receiver of a frame do to catch errors in transmission. The sender computes the checksum of the frame they are sending. They send the frame and the checksum to the receiver. The receiver then computes the checksum themselves and compares it to what the sender sent. If the values don't match, then there has been an error in transmission. When sending something to an AP, if there is an error in transmission, the AP will drop the frame, but if the frame is okay, the AP will send an ACK. WEP uses the CRC-32 checksum, which is very nice and good at catching all sorts of transmission errors, but it is not cryptographically secure.

The way chop-chop works is an attacker first intercepts a Wi-Fi frame. Their goal is to get the WEP key. They chop off the last byte of the frame and send it along with the checksum to the AP. Most likely, chopping that last byte off will cause the checksum to not match anymore, and the frame will be dropped. So the attacker then resends it with a new guess at the checksum. This also likely won't match, but the attacker can keep making new guesses. There are only 256 possible values of the checksum, so eventually one will match. They will know they get a match because the AP sends back an ACK. Then using a little math, they can use the value of the correct checksum to deduce what the chopped off byte must have been. This is the weakness of CRC-32; ideally, knowing the checksum should not be enough to figure out the missing byte, but it's pretty easy with CRC-32.

Once the attacker knows the last byte, they move on to the second-to-last byte, by chopping it off and repeating the process. This is why it is called "chop-chop"; they continue this chopping process until eventually they know all the bytes of the plaintext. Once they have the plaintext and ciphertext, they can XOR them to get the key.

Problem 3 A third vulnerability of WEP is *shared key authentication*. This is where a device needs to prove to an AP that it knows the key. One way to prove that you know a key is to send it to the AP. But that could be observed by someone sniffing traffic, so that would be a bad idea. Instead, a challenge protocol is used, where the AP sends you some plaintext to encrypt. You encrypt it with the WEP key and send the ciphertext to the AP. The AP then does the encryption itself and makes sure it gets the same thing you got. The weakness of this is that someone observing traffic will see both the plaintext and ciphertext. Since WEP uses a stream cipher, they can XOR the plaintext and ciphertext to get the key.

WPA and WPA2 Within a few years of its introduction, WEP was clearly broken. Better security was developed, but it was not compatible with the old hardware, so a stopgap measure, WPA (Wi-Fi Protected Access) was developed. It was compatible and improved on a lot of the weaknesses of WEP. It's not perfect, but it's significantly better. Eventually, WPA-2 was introduced. It is considered the best security to use with Wi-Fi. WPA-3 was introduced in 2018, but it's still not in wide use. There are not many feasible attacks on WPA-2. The one big attack, called Krack, was found in 2017. WPA-2 uses AES in CTR mode. WPA-2 uses a 4-step handshake at the start of a connection. At the third step of the handshake the CTR nonce is sent. Recall from our discussion of CTR mode that the nonce must never be reused. People discovered that if you could stop the last step of the handshake from completing, then the third step could be repeated. Due to an oversight, this had the effect of setting the nonce back to 0, which leads to keystream reuse and then decryption. Patches have been available for it. The other attack involves observing the handshake and using an offline brute-force attack to try to recover the password. This works best if a weak password was used.

WPS Wi-Fi Protected Setup (WPS) is a technology people can use to avoid having to enter in a long WPA-2 password. Sometimes it works by pressing a button on the router, but it can also work via an 8-digit PIN. Unfortunately, WPS uses a really insecure way of verifying the PIN. First, note that an 8-digit pin should provide $10^8 = 100{,}000{,}000$ possible PINs. Now for some reason, when an AP checks if a PIN is correct, it checks it in two separate four-digit chunks. If the first half is incorrect, it will send a message indicating that. So an attacker can use this to figure out the first four digits before moving on to the last four digits. There are $10^4 == 10,000$ possible four-digit codes, so instead of 10^8 possible PINs to check, an attacker just has to check $10^4 + 10^4$ possible PINs. But actually, the last digit is a check digit, so it's actually $10^4 + 10^3 = 11{,}000$ possible PINs. This is considerably less than 100,000,000. Routers are supposed to rate-limit people on so that you can only try maybe three PINs a minute. With rate-limiting, it would still take a long time to try to brute force 11,000 PINs, but unfortunately, some routers did not implement it. Some routers at least provide an option to turn off WPS. Others don't, and some that do actually don't turn it off when tell it to.