Definitions

Attack code.
A program or other implementation of an exploit used to attack the vulnerability in a system.

Attacks.
The use of the attack code against a device.

Exploit.
A method to take advantage of the vulnerability in a device; the method has not been implemented.

Impact.
A measure of what would happen if the device or object was compromised as the result of a security breach.

Risk.
A measure of how critical something is based on several metrics.

Risk assessment.
A process or procedure to determine the level of risk associated with a device or object.

Threat.
A measure of how likely it is that a device or object will be attacked.

Vulnerability.
A weakness in a protocol, application, or other aspect of the network that can be used to attack the device.

Zero-day exploit.
Where the attack code is used to attack a system before the vulnerability or exploit is known outside the developers of the attack code.

4.2 The Taxonomy

Now that we have looked at possible attack points in the network it would be tempting to classify attacks based on the protocol, layer, or application they targeted. There have been many different types of taxonomies proposed over the years with different goals in mind [15–19]. Some of them have been designed to help study the evolution of attacks and to classify attack code. For the purpose of this book, the author proposes another taxonomy that focuses on network security.

This network security taxonomy consists of four categories of vulnerabilities that can be played out against any layer or protocol: header based, protocol based, authentication based, and traffic based. These categories are defined in the following sections with simple examples. Following the description of the taxonomy is a short discussion on how to apply it.

4.2.1 Header-Based Vulnerabilities and Attacks

Header-based vulnerabilities are when the protocol header is created in violation of the standard, such as using invalid values in a field in a header. As we saw in a previous chapter, each layer adds a header to the data it receives (the payload) from the upper layer. This header is used by the layer to carry out the function of the protocol and to communicate with its corresponding layer. For example, one attack would be setting all of the bits to zero in a control field when the standard calls for at least one bit being set. An attacker could also create invalid headers, where the header is too long or too short. This is often seen in freeform headers. Most protocol specifications do not cover the intentional corruption of packet headers, and therefore the consequences of these attacks often are implementation dependent. Different implementations of a protocol will handle these header violations differently.

One of the more famous header-based attacks was the ping of death [20]. Someone discovered that certain operating systems did not handle invalid values in the IP header. The problem was with the way the IP protocol handled segmentation and reassembly. In the IP header there is a length field that indicates the length of the IP packet and an offset field that indicates where the segmented packet is to be placed during reassembly. The operating system allocates a buffer that is 64K in length (the maximum length of an IP packet). As shown in Figure 4.6, the attack contained an invalid packet with an offset of 65528, which is the maximum value for the offset. If the length of the packet was greater than 7, then the packet would not fit into the reassembly buffer. All the attacker had to do was send one packet with the offset set to 65528 and the length greater than 7. When packets arrive out of order, the IP protocol handles the packets by placing them in a reassembly buffer based on the offset. In the case of this attack, the last packet arrives first and the IP layer places it in a reassembly buffer. In some implementations the segmented packet payload was copied into the reassembly buffer without checking to see if it would fit and the data was copied past the end of the buffer. This caused some computers to crash. The protocol specification stated that the maximum packet is 64K in length. The implementation did not consider that the offset and length

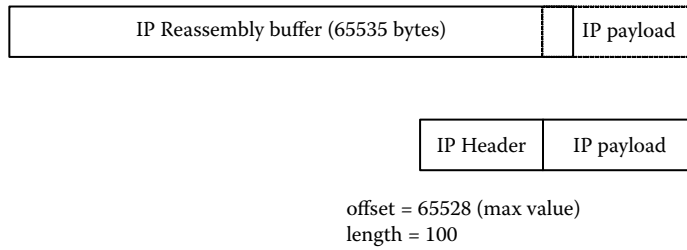

Figure 4.6: Ping of death example.

could be used to create a packet that was longer than allowed, and the programmers never checked to see if the end of the reassembly buffer had been reached.

Header-based attacks are often easy to fix once discovered, but are difficult to discover since they rely on finding mistakes in the implementation of the protocol.

4.2.2 Protocol-Based Vulnerabilities and Attacks

Protocol-based vulnerabilities are where all of the packets are valid, but they violate the procedural aspect of the protocol. As we saw earlier in this chapter, a protocol consists of a series of packets that are exchanged in a certain order to carry out a function. There are several ways for a protocol-based attack to be carried out, including:

Sending packets out of order

Sending packets too fast or too slow

Not sending packets

Sending valid packets to the wrong layer

Sending valid packets to the wrong multiplexed packet stream

Sending packets out of order can involve sending the wrong packet in response to a packet. An example would be sending an open connection packet in response to a closed connection packet. Another example of out-of-order packets is sending a packet that was not expected, like sending an open connection when the connection is already open. Most of these out-of-order packets are covered in either the protocol specification or during implementation. The most common solution is to just drop the out-of-order or unexpected packet.

The case where packets arrive too fast or too slow is often handled during the implementation and is typically treated as an out-of-order packet or unexpected packet. This type of attack is difficult to carry out on the Internet since the end systems have little control over the speed of the packets. A too-slow attack would be the most common and might be best used on shared applications where you keep the application busy waiting for a packet. These types of attacks are not common, and we do want to make a distinction between sending packets too fast and overwhelming a network with too many packets. Attacks that simply send too much network traffic are categorized in a separate classification in the taxonomy.

The missing packet protocol violation is the most difficult one to handle, since in some cases we do not know how long to wait for a response. Think about the telephone protocol, for example. When you call someone, you expect the called party to say something when they pick up the phone. There is no specified amount of time to wait for them to say something.

One classic protocol-based attack violates the TCP open connection protocol. When TCP opens a connection, the protocol uses what is called a three-way handshake. A simple description of the three-way handshake is when the client sends the first packet requesting a connection with the server, and the server responds with a packet indicating it can accept the connection. When the server does this, it must allocate enough memory to maintain the connection. When the client receives the open acknowledgment from the server, it will send back an acknowledgment and the connection is opened. This is shown in Figure 4.7.

The classic attack (called the SYN flood attack) against the three-way handshake is shown in Figure 4.8 [21]. In the SYN flood attack, the attacker sends the open request (called a synchronize [SYN] packet) and the server responds, but the attacker never finishes the three-way handshake by acknowledging the server. This leaves the server in a pending open state waiting for the client acknowledge packet. The attacker sends another open request and does not respond to the server acknowledgment. The attacker continues making requests until all of the server buffer space is allocated and the server cannot accept any additional connections. This is where the name of the attack comes from: the attacker floods the server with SYN packets. There is a timeout specified in the standard for how long to wait for the client to respond to the open connection acknowledgment, but the attacker sends enough requests before the timeout that all of the resources are allocated. This is much more complex to fix, since the time it takes the client to respond is not known and can vary. One fix is to limit the number of connection attempts between a single computer and the server. The attackers circumvented that defense by launching the attack from multiple clients in a coordinated

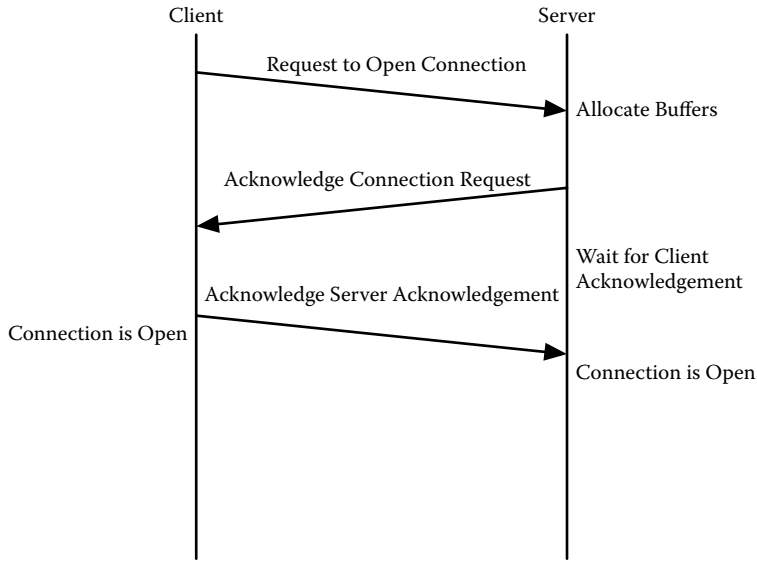

Figure 4.7: Three-way handshake.

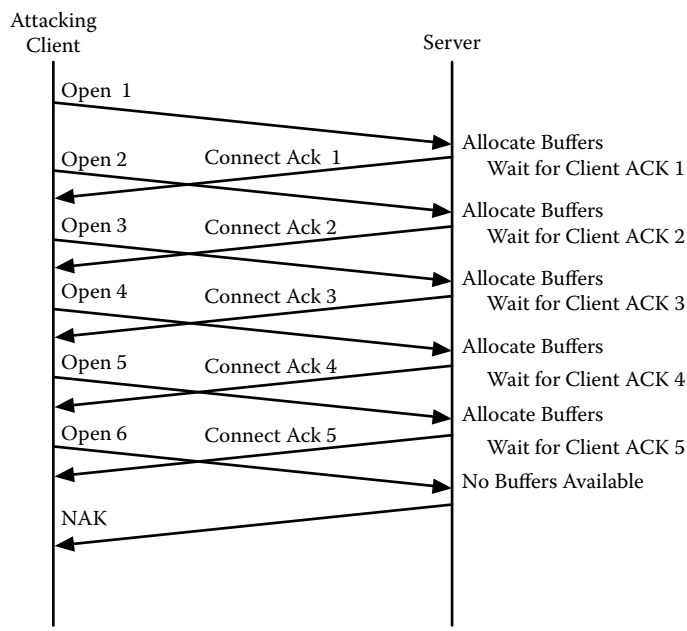

Figure 4.8: SYN flood attack.

attack. This brings up another issue—that attackers are able to adapt to mitigation methods.

4.2.3 Authentication-Based Vulnerabilities and Attacks

Authentication is the proof of one's identity to another. Authentication is often thought of as a username and password. In network security, authentication is where one layer relies on the identity of another layer to carry out its functions. We have already discussed spoofing, which really is an attack on the authentication of a layer. Before we look at categories of authentication we should look at the parts of the network protocol stack that can rely on authentication. Figure 4.9 shows a network protocol stack with the several possible places where authentication would be needed.

Starting with the user we can see that a user might want to prove who he or she is to another user, which is often called user-to-user authentication. User-to-user authentication is where two or more users prove their identity to each other. This is often done with encryption keys and certificates. This form of authentication is most often found in email and secure documents. This type of authentication can be used as a solution to some network-based attacks. The use of user-to-user authentication will be discussed as a solution where appropriate.

The user may also need to prove who he is to an application, host, or protocol layer before he can gain access, which is often called user-to-host authentication. User-to-host authentication is what everyone thinks about when they look at authentication. The most common form is a username and password that allows the user to prove his identity to the resource requesting authentication and gain access to a server, application, or data. This type of authentication is attacked all of time using many different methods, ranging from trying to break the passwords to guessing the passwords. For the purpose of this book, we will not look at user-to-host authentication except where it is part of the network. For example, in wireless security there is a password used to gain access to a secure wireless network.

In the two previous examples a person was responsible for providing the authentication information. In both cases the network is often used to carry the authentication information and is beyond the scope of this book. However, we will see cases where, because we are using a network to carry authentication information, we introduce a security risk. In those cases we will examine methods to mitigate the risk.

Two other types of authentication involve an application, host, or network as the object providing the authentication.

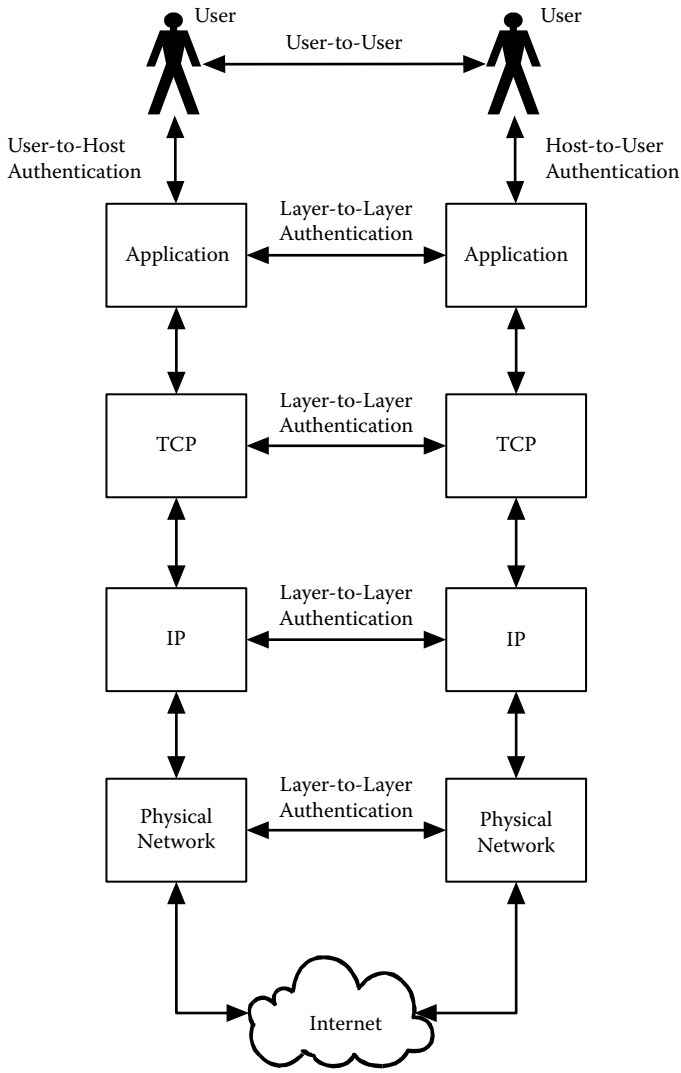

Figure 4.9: Network authentication.

As shown in Figure 4.9 and as we have discussed in the earlier chapters, two layers communicate with a protocol and implied in the communication is the fact that each layer knows the identity of the other. Authentication between two applications, hosts, or network layers is referred to as host-to-host authentication. Host-to-host authentication is where two hosts authenticate each other in order to

carry out a function. This is often done using the host or application addresses, like the IP address or hardware address. This form of authentication can be weak because, as we have seen, addresses can be changed.

The final type of authentication is where an application, host, or network layer provides proof to a user of its identity. This is called host-to-user authentication, which allows the user to prove the identity of the host he or she is connecting to. This is often used when a user connects to a secure web site. However, we will see that in many cases the user does not authenticate the host, or the authentication is done using the IP or hardware address, which can cause security problems. We will look at several attacks based on nonexistent or flawed host-to-user authentication.

4.2.4 Traffic-Based Vulnerabilities and Attacks

Traffic-based vulnerabilities and attacks focus on the traffic on the network, either having too much traffic on the network or an attacker being able to capture the traffic and steal the information.

Traffic-based vulnerabilities occur when too much data is sent to a layer or layers and they cannot keep up with the incoming data, which can cause the layer to drop packets or stop handling packets all together. These attacks can be the most devastating to a network and can be caused by a single attacker or by multiple attacking devices working together. We will look at traffic-based attacks throughout the remaining chapters since each layer responds differently to too much traffic. Also, depending on the type of traffic, sending a single packet can cause multiple packets to be sent in response, thus creating a flood of traffic. One example of a single-packet traffic-based vulnerability is where the attacker would send a directed broadcast packet into a remote network that required a response. A broadcast packet is one that is received by all devices on a network. As shown in Figure 4.10, the attack code sends a broadcast packet into a network, and every device on the network gets the request and responds back through the router. If the network is large, a single inbound packet could create hundreds of outbound packets. If the attacker floods the network with the inbound requests, then tens of thousands of outbound packets would be generated per second, which would cause the victim's network to become flooded and make it unusable.

Another type of traffic-based vulnerability is packet sniffing. Packet sniffing is where you capture all of the traffic on a network. Traffic sniffing can be carried

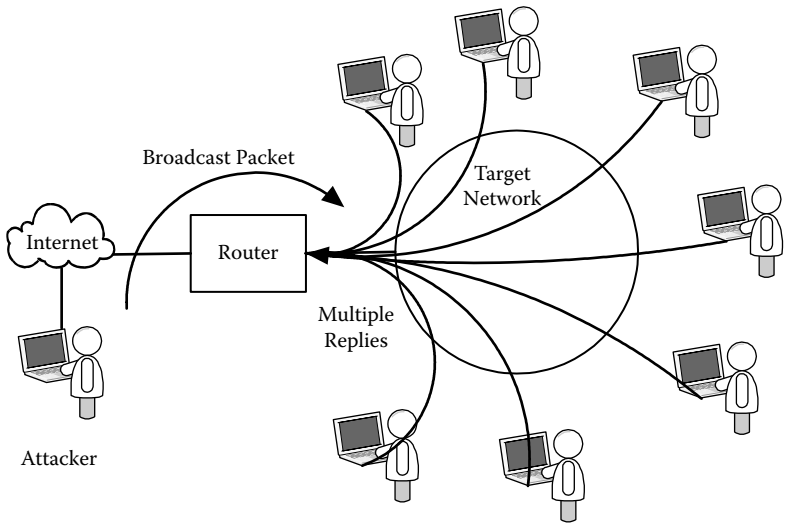

Figure 4.10: Broadcast flood attack.

out against almost every protocol used in the Internet. The vulnerabilities due to traffic sniffing depend on the protocol.

4.3 Applying the Taxonomy

There are a few comments that can be made about using the taxonomy to classify vulnerabilities and attacks. To start with, the four categories do appear to have some overlap, since authentication-based attacks deal with a goal as well as a method. Header-based, protocol-based, and traffic-based attacks are methods of attack. The best way to help tell the difference is if the goal of breaking authentication is accomplished using one of the other three methods, then it is not classified as an authentication-based attack. For example, if a header-based attack causes the attacker to gain access to a computer, it would be tempting to call that an authentication-based attack, but the method used was header based.

The more complex categorization is when the attacker uses the payload to attack the authentication. This would be classified as an authentication-based attack. Think of it as the method, which in this case would be authentication, which also happens to be the goal.