EE122 Section 12 – Security

Question 1 – Crackdown on P2P

In 2007, a large broadband provider in the US disrupted end users' BitTorrent connections with a layer-4 Denial-of-Service (DoS) attack. The provider identifies BitTorrent connections by checking each packet's source/destination port pairs against a blacklist.

(a) More specifically, how might this have worked? Assume TCP is used.

The ISP can inject a single TCP RST packet, which tears down the connection. It can do this because it owns all the routers in the network, and thus knows the previous sequence number in the connection. The ISP is effectively a man-in-the-middle.

(b) How might the end host defend against this attack

a. On the application layer (TLS/SSL)?

Nothing. Encryption does not work because the transport header is still in the clear.

b. On the transport layer?

Nothing. TCP sequence number defends against blind spoofing, but not man-in-the-middle attacks.

c. On the IP layer?

The end hosts can encrypt the entire IP packet payload, which includes the TCP header. An example is IPsec.

(c) How can a different attacker in the same ISP (but is not the ISP itself) launch this attack?

S/he would have to resort to blind spoofing (i.e. inferring/guessing the correct TCP sequence number). This is much harder.

Question 2 – Defending against Spoofing

You are the network administrator for a large company. Your company will be held liable for any spoofing attacks that originate from within your network.

(a) What can you do to prevent spoofing attacks

i. by your own employees?

If an outgoing packet has a source address that doesn't match your subnet, drop. (Note: this means an employee can still spoof another employee within the network)

ii. by outsiders?

If an outgoing packet has a source address that matches your subnet, drop. (Note: This means an outsider can still spoof another outsider)

(b) Your company runs FailChat, an application that runs on top of UDP. Why is this more vulnerable to spoofing than ProChat, an application that runs on top of TCP?

TCP offers 3-way-handshake, which sends the SYNACK back to the source address of the SYN packet. The attacker does not get this SYNACK, and so cannot respond to it efficiently. Instead, s/he must resort to guessing sequence numbers (blind spoofing). UDP offers no such mechanism.

(c) Unfortunately, your uninformed CEO is adamant that the company must continue to use FailChat, because it is a product developed by the company itself. How can you modify FailChat to defend against spoofing?

Whenever a message is received, send a confirmation request to ask the sender whether the packet did originate from the source address it claims. Accept this message only if we receive the sender's confirmation response. This is essentially implementing handshaking in the application layer.

To guard against the attacker blindly spoofing the confirmation response, attach a nonce to the confirmation request, and require the sender to attach the same nonce in his/her confirmation response.

(d) What implications does the solution you proposed in (c) have?

It is costly to verify every single message you receive, as it includes an additional RTT to verify the sender. This can be mitigated by caching the result of the verification.

Additionally, it opens up an opportunity for DoS attacks, as each spoofed packet makes the receiver compute a nonce and send an extra packet.

Question 3 – DoS

You are an evil mastermind and you want to launch a DoS attack on web server X. However, it is still early in your hacking career and you don't have many resources.

(a) What mechanism in TCP makes your attack less effective?

The 3-way-handshake. Without it, TCP allocates a receiver buffer whenever it encounters a new connection. Instead, the 3-way-handshake guarantees that the receiver buffer is only allocated after it receives an ACK from the sender.

(b) You naively use your laptop to launch a TCP SYN flooding attack. How can web server X guard against this if

i. You use your true address?

A simple adaptive firewall can learn to block all packets from your IP.

ii. You use spoofed addresses?

There is nothing web server X can do. However, note that this is not a very efficient DoS attack, since your throughput is constrained by the bandwidth of your local network.

(c) Assume web server X provides for an e-commerce service that uses HTTPS. Is this in your favor?

The fact that web server X must perform cryptographic computations on demand is certainly a DoS opportunity. However, in TLS/SSL, this happens only after the TCP connection is fully set up. This implies that spoofing is not an option.

Thus, this is to your advantage only if you use your true IP address, an approach that is ineffective unless you have many machines.

(d) Realizing the follies of your previous attempt, you decide instead to launch a DDoS attack. Unfortunately, you don't have many evil friends to help you out. What can you do?

Send spoofed TCP SYN packets to random destinations using the address of X as the source address. This causes X to suddenly receive many SYNACK packets.

(Further, you can take advantage of the services that amplify your packets, such as DNS. DNS response is typically much larger than the DNS request. Sending many spoofed DNS requests is even more effective.)

(e) Twenty years later, you finally assembled an army of compromised hosts. How can web server X mitigate the consequences of your DDoS attacks?

When the time finally comes, there is nothing web server X can do to prevent large-scale DDoS attacks. To mitigate the consequences, however, web server can resort to over-provisioning or distributing its services to multiple web servers. Both approaches are proactive, rather than reactive.

Question 4 – Keys

Andrew wants to send message M to Steve. Andrew knows about Steve's public key, but Steve knows nothing about Andrew.

(a) Suppose Andrew sends an encrypted M to Steve using Steve's public key. Is this safe if

i. There is an eavesdropper?

Sure. The eavesdropper cannot decrypt the message, since s/he does not have Steve's private key.

ii. There is a man-in-the-middle?

No. The man-in-the-middle can replace the encrypted M with an encrypted M'.

(b) Normally, to verify that M is indeed sent by Andrew, Steve can use the hash (MAC) that Andrew sends along with the message. Why can't this happen here?

The MAC that comes with M is signed with Andrew's private key. Since Steve does not have Andrew's public key, he cannot verify whether the validity of M with the MAC.

(c) As a remedy, Andrew sends his public key K in addition to M and the hash to Steve, all encrypted with Steve's public key. Is this safe if

i. There is an eavesdropper?

Yes. As before, the eavesdropper cannot even read what Andrew sent.

ii. There is a man-in-the-middle?

No. The man-in-the-middle can not only replace the encrypted M with encrypted M', but also replace the encrypted K with an encrypted K' of his/her choosing.