Scribe: Cryptography and Network Security (Week 4 - Class 1)

Rashil Gandhi

26-Sep-2020

1 Introduction

This class introduces some basic concepts of TCP communication and DNS resolving. It also tells us how rudimentary attacks on them are performed and what mitigation steps are taken.

2 TCP Communication

2.1 3-way handshake

Whenever a user (let's say Alice) wishes to contact a server, she sends a SYN packet to let the server know. The SYN packet has a random 32-bit sequence number (suppose A) associated with it to uniquely identify this connection request. In response, the server replies with a SYN-ACK packet. An acknowledgment number is sent with this, which is set to one more than the received sequence number i.e. A+1, and the sequence number that the server chooses for the packet is another random number, B. Finally, the client sends an ACK packet back to the server. The sequence number is set to the received acknowledgement value i.e. A+1, and the acknowledgement number is set to one more than the received sequence number i.e. B+1. This establishes a TCP connection between them with both of them confirming that they are ready to respond to each other.

2.2 Attacks on TCP

Two types of attacks can be performed on TCP.

2.2.1 Blind Spoofing

In this attack, an adversary (let's say Eve) spoofs Alice's IP address (i.e. fools the server that the request is coming from Alice's IP when actually it's not). Eve sends a SYN packet to the server. The server responds with an ACK and

another SYN packet (whose sequence number is Y, suppose) to Alice's IP. Now, Eve has no way of knowing Y (since Y was sent to Alice's original IP location). So, Eve tries to guess Y either by brute-forcing or by some other methods. If Eve guesses Y correctly, it sends an ACK to the server with number Y+1, therefore successfully creating a forged connection as Alice.

Originally, Y was based on timestamps. It is nowadays a pseudo-random number to make brute-forcing difficult.

2.2.2 RST Hijacking

In this, if Eve knows the port on which an already established TCP communication between Alice and the server is happening, she can send reset requests to the server to kill the TCP connection by guessing Y. This is done by sending a forged request with RST flag set to 1 in the header.

Generally, Eve would flood the network with reset requests, trying all the possible Y sequence numbers.

3 DNS Resolution

3.1 Resolution and Caching

DNS resolvers provide clients with the IP address that is associated with a domain name. In other words, they take human-readable website addresses like 'example.com' and translate them into machine-readable IP addresses. When a user attempts to navigate to a website, their operating system sends a request to a DNS resolver. The DNS resolver responds with the IP address, and the web browser takes this address and initiates loading the website.

A DNS resolver will save responses to IP address queries for a certain amount of time. In this way, the resolver can respond to future queries much more quickly, without needing to communicate with the many servers involved in the typical DNS resolution process. DNS resolvers save responses in their cache for as long as the designated time to live (TTL) associated with that IP address allows them to.

3.2 Attacks on DNS - Cache Poisoning

Attackers can poison DNS caches by impersonating DNS nameservers, making a request to a DNS resolver, and then forging the reply when the DNS resolver queries a nameserver. This is possible because DNS servers use UDP instead of TCP, and because currently there is no verification for DNS information. With UDP, there is no guarantee that a connection is open, that the recipient is ready to receive, or that the sender is who they say they are. If a DNS resolver receives a forged response, it accepts and caches the data uncritically because there is no way to verify if the information is accurate and comes from a legitimate source.

Despite this, DNS poisoning attacks are not easy. Because the DNS resolver does actually query the authoritative nameserver, attackers have only a few

milliseconds to send the fake reply before the real reply from the authoritative nameserver arrives.

3.3 Kaminsky Vulnerability (2008)

A vulnerability discovered in 2008 allowed attackers to bypass the TTL defense of DNS caches by targeting "sibling" names like "83.example.com" instead of "www.example.com" directly. Because the name was unique, it had no entry in the cache, and thus no TTL. But because the name was a sibling, the transaction-ID guessing spoofed response could not only include information for itself, but for the target as well, in the form of additional information sent to the DNS server. By using many "sibling" names in a row, they could induce a DNS server to make many requests at once. This provided enough opportunities to guess the transaction ID to successfully spoof a reply in a reasonable amount of time.

3.4 Mitigation

To fix this vulnerability, all major DNS servers implemented Source Port Randomization, which used randomized UDP ports along with the transaction ID's to make guessing the ID's 65536 times harder. This was not actually a repair, but rather a stopgap measure.

Other scientists have proposed using DNSSEC, an improved method of DNS resolution that uses public-key cryptography to verify the identity of authoritative nameservers. This however has slow and costly adaptation.

4 Conclusion

We saw how vulnerabilities exploit TCP and DNS internal processes and what measures are taken to prevent them.

References

www.cloudflare.com/learning/dns/dns-cache-poisoning

ktflash.gitbooks.io/ceh_v9/content/103_network_level_session_hijacking.html

en.wikipedia.org/wiki/Transmission_Control_Protocol

duo.com/blog/the-great-dns-vulnerability-of-2008-by-dan-kaminsky