9/15: Security problems with TCP/IP

Scribe: Derek Leung

The TCP and IP protocols have been a classic area of focus for network security owing to their ubiquity, age, and importance. Designed at a time when the Internet was small and friendly, attackers have taken advantage of many of the protocols' shortcomings unanticipated by their designers as the Internet rapidly grew in scale and complexity. Although many modern systems rely only on end-to-end encryption as problems of economics hinder the adoption of security measures, it is still worth taking a look at the security implications of these protocols' design.

1 Threat Model

Since a network is comprised of nodes and links, the threat model focuses on the compromise of either of these. An adversary may attack a node, compromising or impersonating a server, or attack a link, eavesdropping or intercepting communications on a connection between two parties. In fact, due to problems in these protocols' design, two common issues have been denial-of-service attacks on servers and connection hijacking attacks mounted by malicious users. Because implementation details are easy to change but design details very difficult in a distributed setting, security researchers have focused primarily on the design of these protocols.

2 TCP

TCP is a protocol designed to ensure reliable global transmission of packets over the Internet.

A TCP session begins with a three-way handshake between the client and the server (Figure 1). The client opens by announcing to the server that it wants to initiate a connection by sending a SYN message. It passes a sequence number SN_C along with this request. The server responds with a SYN-ACK message where it increments this sequence number $SN_C + 1$ and sends its own SN_S. Finally, the client returns to the server an ACK message with its incremented sequence number $SN_s + 1$.

The choice of a proper initial sequence number (ISN) is crucial.

- **Fix** $ISN = 1$

  If the initial sequence number is fixed to a value of 1, the protocol is incorrect. If at any point either the client or the server restarts the connection, neither party will be able to tell whether packets sent and received over the network are fresh or stale packets lingering from the previous connection.

It is thus necessary to advance the ISN at every new connection to prevent old connections from interfering with new ones. In particular, the TCP specification mandated that parties increment the ISN at 250,000 values per second.

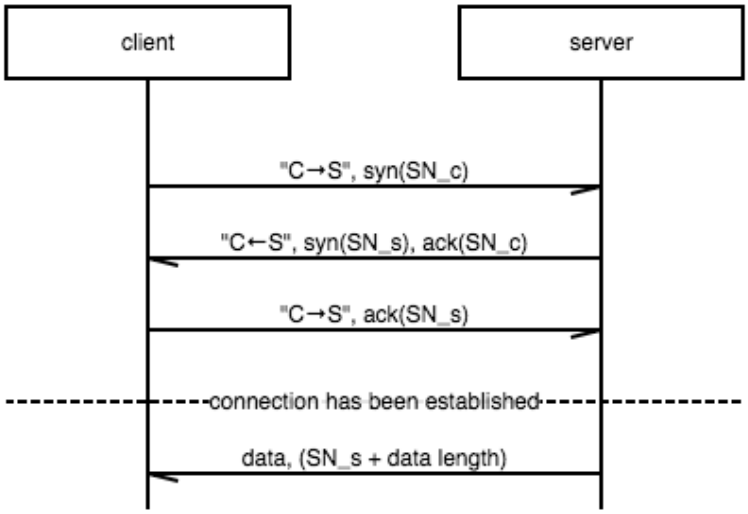

Figure 1: TCP three-way handshake

- **Insufficient ISN increment rate**

  If the rate at which the ISN is incremented is too low, the correctness issue persists. For example, if the rate is 100 values per second, and a server sends a client more than 100 bytes of data, then if another connection is established one second later the new connection will interfere with the old one.

- **Excessive ISN increment rate**

  On the other hand, if the rate at which the ISN is incremented is too high, wrap-around issues due to the fixed size of integers on nodes will cause the same correctness issues. At 250,000 values per second, the ISN values wrap around every 4 hours. A rate too high reduces this wrap-around time and may increase interference between connections.

2.1 Attack 1: Sequence Number Prediction

The problem with a fixed ISN incrementation rate is that an attacker can predict an ISN of a host and thus impersonate it. In a TCP sequence number attack, the attacker first begins a legitimate connection with the target host to get an initial sequence number SN'_s. Some time (t seconds) later, when the host tries to communicate with a server, the attacker floods the network with forged requests to reset the connection. Then because the incrementation rate of sequence numbers is fixed, the attacker can then derive the new initial sequence number $SN_S = SN'_S + t * 250000$ and impersonate the host.

2.1.1 Consequences

1. Applications which relied only on IP addresses for authentication were vulnerable to this attack. For instance, the `rlogin`, `rsh`, and `rcp` programs which authenticated using hostnames alone were vulnerable to this sequence number attack.

2. Using connection reset requests, adversaries can mount denial-of-service attacks on the network. A common instance of a denial-of-service attack would be one in which an attacker resets a client's valid connection by forging reset requests from the client.

3. By predicting sequence numbers in a connection, an attacker can hijack existing TCP connections.

2.2 Countermeasure 1: Sequence Number Randomization

The solution to the problem of predictable sequence numbers is to simply use unpredictable sequence numbers. However, two constraints exist:

- The entropy of the sequence number must fit in the size of a machine integer.
- The sequence numbers must continue to approximate an incrementation rate of 250,000 values per second (or at least, it must change at a rate so as to avoid correctness issues from overlapping connections).

To satisfy these constraints, a server can set a new ISN using a randomness function:

$$ISN = ISN_{\text{old}} + \text{RandomFn}(srcip, srcport, dstip, dstport, key) \tag{1}$$

where ISN_{old} is a global value holding the last ISN, and key is a global secret used to make the randomness unpredictable.

2.3 Attack 2: SYN Denial of Service

The design of the TCP handshake requires the server to maintain connection state for each incoming SYN request, potentially consuming many server resources. Specifically, the server must maintain a table of $(connectionid, srcip, srcport, dstip, dstport, ISN_s)$ rows: one for each client attempting to connect. If an attacker initiates (or spoofs) SYN requests to connect from many different sources, the server will need to maintain many such rows. Eventually it will run out of memory and fail to serve subsequent requests.

2.4 Countermeasure 2: Deterministic Randomness

To mitigate the memory depletion caused by these attacks, the server can simply handle handshake requests statelessly. Instead of storing rows in the table immediately after the initial SYN request, the server creates table entries only after the handshake is completed. Essentially, the server does not need to remember the ISN_s value it sent to a client if it can instead deterministically compute this value. In fact, the server can use a message authentication code to do this. Equation (1) thus becomes

$$ISN = ISN_{\text{old}} + \text{MAC}_{key}(srcip, srcport, dstip, dstport) \tag{2}$$

thus obviating the need for storing connection state before a handshake is established.

3 DNS

DNS is a service allowing clients to look up the IP address associated with a hostname.

3.1 Attack: DNS spoofing

By forging DNS responses, an attacker can redirect a client to a malicious IP address (Figure 2). If the attacker responds to a client's DNS request before the DNS server does so, the attacker can give the client a false IP address IP' and the client will accept it instead of the real address IP.

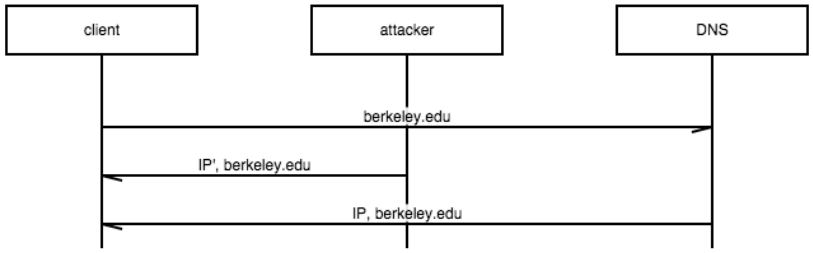

Figure 2: DNS spoofing attack

3.2 Countermeasure: DNS signing

To guard against DNS spoofing attacks, a client must be able to check the validity of the DNS response. This can be done with a public-key signature scheme: instead of sending just $[IP, \text{berkeley.edu}]$, the DNS responds with $[IP, \text{berkeley.edu}]_{S_K}$ where s_k is some signature key. Designing the signature of responses must solve two problems:

- The party signing the mapping of domain name to IP address should be authorized to do it.
- There should be some way to authenticate responses which involve invalid domain names.

To solve both of these problems, it is sufficient for the owner to sign only a whitelist of valid domains. However, more problems arise:

- The list of all valid domains is very long, and sending this list is prohibitively expensive.
- Domain name owners must be able to invalidate entries.

To compress this list, a DNS server can send lexicographically-ordered pairs which cover the desired range of IP addresses a domain-name owner wishes to sign. Moreover, the server adds an expiration date to this range. Thus, the final entry signed by a server is of the form

$$[h_1, IP_1, h_2, IP_2, TTL]_{s_k}$$

where TTL represents the time at which the entry expires.

One problem remains: with these pairs, an attacker can easily enumerate addresses owned by a domain. This countermeasure is thus a compromise between the ease of enumerating addresses and authenticating DNS responses.

4 Other Protocols

4.1 ARP

ARP is a protocol for converting IP address to MAC addresses. In ARP, a client sends a broadcast request for a MAC address when necessary. As in the DNS attack, a malicious attacker on the network can respond with a false MAC address. As a countermeasure, the network can treat switches and routers as "trusted servers" for translating IP address to MAC addresses.

4.2 DHCP

DHCP is the protocol by which a client queries for the IP address of a DNS server. Again, an attacker an respond with a false address. Like ARP, a solution is for the network to trust switches for DHCP.

4.3 BGP

In BGP, a host announces the route to a given address. However, nothing prevents a malicious host from announcing a false route. Thus, in a secure protocol (SBGP), route announcements are cryptographically signed.