EECS 388

Introduction to Computer Security

Lecture 13:

Network Defense

October 8, 2024
Prof. Chen

Web and Network Security

Last three weeks:

- The Web Platform
- Web Attacks and Defenses
- HTTPS and the Web PKI
- HTTPS Attacks and Defenses
- Networking 101
- Networking 102

Today:

- **Network Defense**

Later:

- Privacy and Anonymity
- Censorship and Circumvention

Denial of Service Attacks

Denial of service (DoS) attacks overwhelm a host or network with traffic, such that it can't process legitimate requests

DoS stems from *asymmetry*: especially vulnerable if attacker's cost to make requests is low, victim's cost to process them is high

DoS Bug: Design flaw that allows a malicious client to easily disrupt service (e.g., if a malformed request can crash the server)

DoS Flood: Attacker sends a large number of messages to exhaust finite resources (e.g., bandwidth, CPU, memory)

DoS opportunities exist at every layer:

- **Physical:** e.g., **radio jamming**
  interfere with transmission medium
- **Link/Network:** e.g., **packet flooding**
  exceed capacity of on-path switches/routers
- **Transport:** e.g., **TCP SYN flooding**
  open many simultaneous connections, each requiring the server to maintain state
- **Application:** e.g., **HTTP request flooding**
  cause servers to perform expensive queries or cryptographic operations

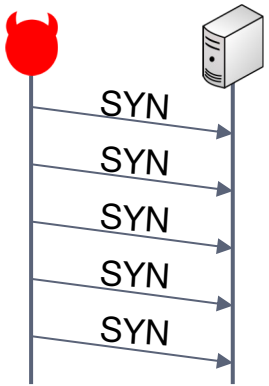
Preventing/responding to DoS attacks requires careful engineering and monitoring at each layer.

DoS: TCP SYN Flooding

SYN flood attack: attacker sends SYN packets from many spoofed source IPs

TCP implementations commit resources before receiving a valid ACK (which confirms source IP)

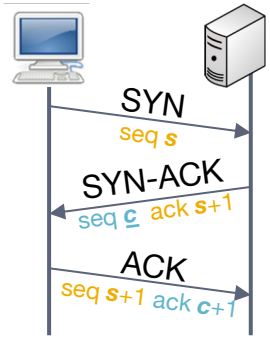
Spoofed SYNs fill up the server's connection backlog queue. When full, no service is possible



Poor solution: Increase backlog queue size
Poor solution: Decrease connect timeout

Good solution: Avoid committing state (i.e., memory) until 3-way handshake completes

SYN cookies is a defense against SYN flooding that *encodes* the initial connection state in the SYN-ACK packet itself. Invented by D. J. Benstein.



Server must remember session parameters to later validate the ACK.

For secret ***k***, server generates "cookie" $c = MAC_k(cli\_IP \| cli\_port \| s)$. Sets its initial seq. num. to first 32-bit bits of ***c***.

Server validates ACK by recomputing ***c*** and comparing to the ack number.

If connection is legitimate, client's ACK packet will use parameters ***s***+1 and ***c***+1, and MAC will validate.

Off-path attacker doesn't see **c**. Hard to guess.

Server has to rotate ***k*** (to timeout old values), but this is constant state in number of received SYNs

DoS: Amplification

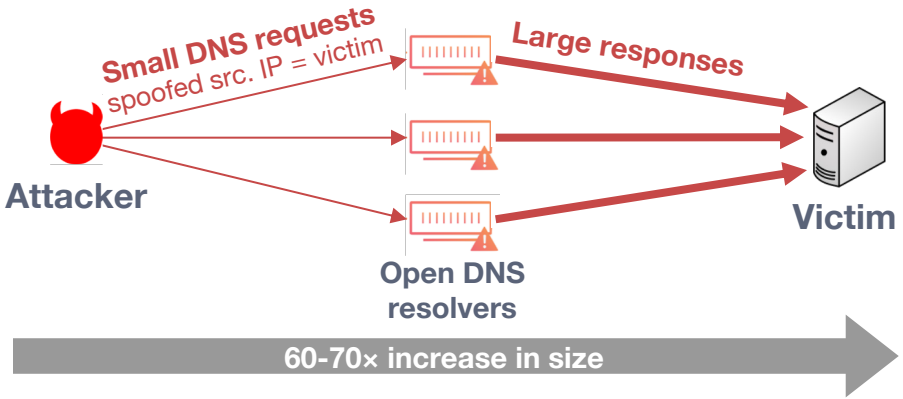
Services that respond to a single small **UDP packet** with a large UDP packet can be used to **amplify** DoS attacks

Attacker sets source IP to victim's IP address. When the service responds, it sends an *even larger* amount of data to the victim

Amplification in common protocols:

- **DNS:** (public recursive resolvers) `ANY` query returns all records server has about domain
- **NTP:** (public time servers) `MONLIST` returns IPs of last 600 clients who asked for the time

Allowing these requests on public servers is considered a misconfiguration, but there are 100Ks of misconfigured hosts!



Victim can easily filter out a few source IPs, so attacker needs to find *many* servers that amplify. Simultaneous DoS from many dispersed IPs is **Distributed Denial of Service (DDoS)**

Historically powerful DDoS attacks:

2013: DDoS attack generated **300 Gbps** using 31,000 misconfigured open DNS resolves

2014: **400 Gbps** DDoS using 4,500 NTP servers

DDoS Threats and Defenses

If an attacker can compromise enough hosts, can perform DDoS *without* amplification.

Botnets are collections of compromised machines (**bots**) under the unified control of an attacker (**botmaster**). Used for DDoS

Example: The **Mirai Botnet** (2016) infected 600K IoT devices, like home routers and IP webcams. Caused nearly 1 Tbps of DNS and HTTP requests!

THE WALL STREET JOURNAL.

Cyberattack Knocks Out Access to Websites

Popular sites such as Twitter, Netflix and PayPal were unreachable for part of the day

DDoS Defenses:

Egress filtering: To prevent IP source address spoofing, ISPs should drop packets when source IP is outside the local CIDR block

- *Disadvantage*: Requires global coordination. All ISPs need to filter for this to be effective, but little incentive for an ISP to implement.

Content Delivery Networks (CDNs): Large CDNs have enough bandwidth to absorb very large DDoS attacks. Security teams work with ISPs to filter DoS traffic upstream

CLOUDFLARE

Example: **Google Project Shield** (provides free DDoS protection via Google CDN to public interest sites, e.g., news, human rights orgs.)

DDoS Defense: Client Puzzles

Idea: What if we force every client to do moderate amount of work for every connection they make?

Client puzzles require clients to provide a **proof-of-work** before receiving service

1. Server sends challenge: **n**, **k**
2. Client solves challenge: finds **X** s.t. MAC_k(**X**) ends in **n** 0 bits.

Observe:

- On average, client must perform 2^{n-1} MAC computations to solve challenge.
  For **n**=20, takes about 0.1 s on a modern CPU
- Server requires only one MAC computation to validate client's response
- Maybe activate only when under attack (like SYN cookies).
  Server can dynamically increase **n** when it is under heavy load

Network Defenses: Adding Encryption

Legacy network protocols are plaintext. **Must add encryption at *one or more* other layers**

Several approaches:

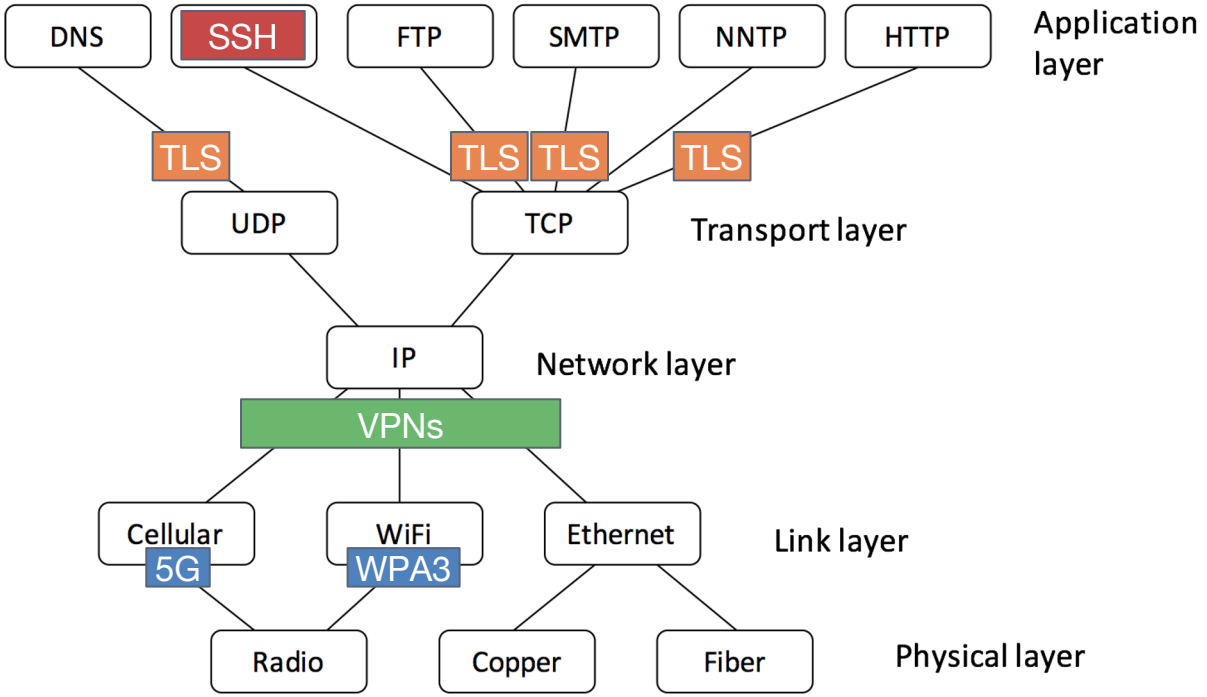
1. **Integrate into apps** (e.g., SSH)
2. **At transport layer** (e.g., TLS)
3. **Below network** (e.g., VPNs)
4. **In the link layer** (e.g. 5G, WPA3)

Different tradeoffs

Modern thinking prefers **end-to-end encryption** (i.e., option 1 or 2) since:

- Entire path is encrypted
- Authentication is simplified
- Upgrading tends to be easier

Remember, the network is evil and wants to kill you!



Defense: Building Protocols Securely

Implementing your own service?

Don't build a network proto from scratch!

- Never "roll your own" crypto
- Many opportunities to mess up crypto design or parsing network packets

Modern frameworks do the hard work for you:

gRPC: http2 + TLS 1.3 RPC framework

- Safe parsing in 11 languages
- Exceptionally efficient
- Streaming/Sync/Async
- TLS-based authentication

Or, use REST on top of HTTP/2 + TLS 1.3.

gRPC example:

```
syntax = "proto3";

package calc;

message AddRequest {
  int32 n1 = 1;
  int32 n2 = 2;
}

message AddReply{
  int64 res = 1;
}

service Calculator {
  rpc Add(AddRequest)       returns (AddReply) {}
  rpc Substract(SubRequest) returns (SubReply) {}
  rpc Multiply(MultRequest) returns (MultReply) {}
  rpc Divide(DivideRequest) returns (DivideReply) {}
}
```

Defense: Passive Network Monitoring

Intrusion Detection Systems (IDSes) are software/devices to monitor network traffic for attacks or policy violations

Violations reported to central security information and event management system where analysts can later investigate

Two types:

Signature Detection: Maintains list of traffic patterns (rules) associated with attacks

Anomaly Detection: Attempts to learn normal behavior and report deviations

Example: zeek

Network Telescopes passively monitor traffic coming into unallocated address space

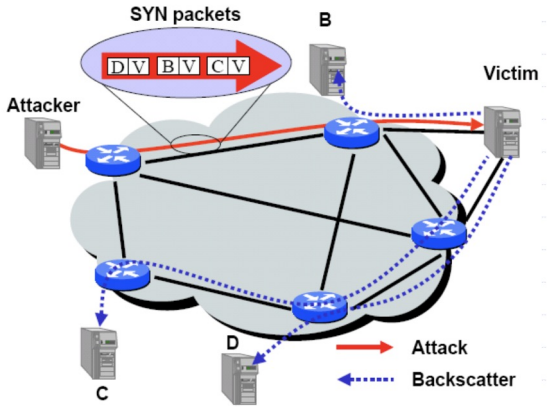
Used to monitor for Internet-wide scanning and for certain kinds of DDoS attacks

DoS attacks that forge random source IPs produce **backscatter** of SYN-ACKs to IPs in the telescope's address space

Example:

Merit Network Telescope

Records all inbound Packets to range of ~500k otherwise-unused IP addresses



Defense: Active Port Scanning

Host/network's **attack surface** is set of services running there that are reachable from outside

- May use weak or default passwords
- May be old versions/lack security patches
- May be forgotten or unmanaged

We assess network attack surfaces using active **port scanning**: Send a SYN (or app-specific UDP packet) to a port to see if any service is listening

Frequently used for **network penetration testing**.

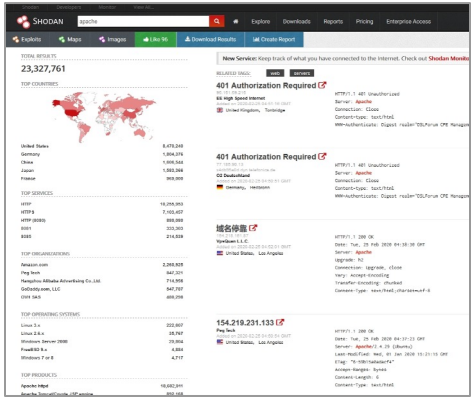
Vertical scanning: Try large number of ports on a single host or small network. Typically **Nmap**:

```
$ nmap www.eecs.umich.edu
Starting Nmap 7.60 ( https://nmap.org )
Not shown: 992 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
25/tcp   open  smtp
80/tcp   open  http
443/tcp  open  https
3306/tcp open  mysql
```

Horizontal scanning: Try a single port on a large number of hosts (e.g., *all* of IPv4!) Typically **ZMap**:

```
$ sudo ./zmap -r0 -p80 -o out.csv
Feb 19 21:03:13.960 [INFO] zmap: output module: csv
Feb 19 21:03:13.999 [INFO] recv: Data link layer Ethernet
0:01 0%; send: 1319818 1.32 Mp/s; recv: 17843 17.8 Kp/s; hitrate: 1.35%
0:02 0%; send: 2784336 1.46 Mp/s; recv: 41997 24.1 Kp/s; hitrate: 1.51%
0:03 0%; send: 4248172 1.46 Mp/s; recv: 65987 24.0 Kp/s; hitrate: 1.55%
0:04 0%; send: 5712161 1.46 Mp/s; recv: 89754 23.8 Kp/s; hitrate: 1.57%
0:05 0% (43m left); send: 7176055 1.46 Mp/s; recv: 113607 23.8 Kp/s; hitrate: 1.58%
```

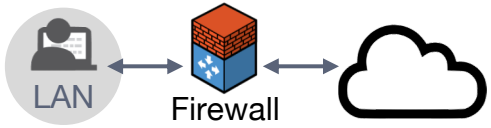
Service such as **Shodan.io** scan thousands of ports continually, make results publicly searchable:



Both attackers and defenders can use port scanning

Defense: Firewalls

Firewalls reduce a network attack surface by only allowing some traffic to pass:



Firewall inspects traffic and applies policy

Can be part of networking devices (e.g., routers), virtualized environments, or individual host's OS. Often use more than one. Examples:
Linux kernel **iptables**, Amazon EC2 **security groups**

Basic packet filtering policies use only:

- source and destination address
- protocol (TCP, UDP, ICMP, etc.)
- TCP flags, TCP/UDP source and destination ports

Example: You have lots of services, but only want outsiders to be able to access HTTPS?

DROP ALL INBOUND PACKETS IF DST PORT != 443

***Issue*: All outbound connections also have a source port. Don't want their responses blocked**

Firewalls with **stateful filtering** track outgoing connections and allow associated inbound packets back through (Caution: State carries DoS risk!)

Application layer firewalls enforces protocol-specific policies (e.g.: scanning SMTP email for viruses, or inspecting HTTP POSTs for XSS exploits)

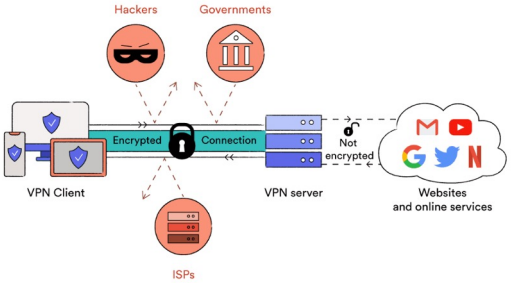
Many orgs. inspect **outbound traffic** too (e.g., to prevent **data exfiltration**, block services like BitTorrent, or detect malicious sites).

TLS interception firewalls: TLS complicates application-layer firewalls. Must either run on the host or force clients to install custom root CA cert. Buggy implementations risk degrading security

Defense: Virtual Private Networks

Issue: How to provide security for, e.g., non-encrypted protocols across the public Internet?

A **virtual private network (VPN)** creates an encrypted channel that "tunnels" IP packets to a distant network location.



Provides confidentiality and integrity of packets *inside the tunnel*, authentication of endpoints

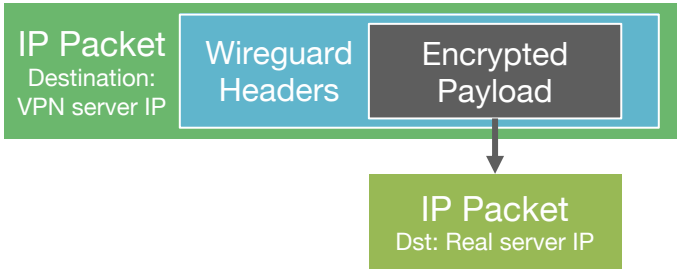
But…VPN can't protect packets traveling beyond endpoints (i.e., from VPN server to destination)

Broad applications of VPNs:

- Allow a remote device (e.g., a traveling employee) to access a corporate network
- Bridge two private networks via the Internet
- Provide Internet access from a distant ISP (to bypass local censorship or surveillance)

Common VPN protocols:

IPsec: complicated legacy protocol
OpenVPN: open-source, TLS-based
AnyConnect: proprietary, TLS-based
Wireguard: modern, high-performance



Zero Trust Security

Network Perimeter Security Model

VPNs and firewalls support the idea of a **secure internal network** that's isolated from the **untrusted public Internet**.

Disadvantage: If the network perimeter is breached, attacker gets privileged access

- User devices are compromised frequently
- Internal network services tend to be weak

Zero Trust Security Model

Modern best practice **assumes the internal network is evil too**. Removes privileged intranet and puts all services on the Internet

Goal: Protect *applications*, not the network

Access control decisions depend solely on device and user credentials, regardless of user's network location

Seamlessly supports remote work

Zero trust is a major philosophical shift in corporate network security

But…most universities have practiced zero trust for decades, so it may already feel familiar

Coming Up

Reminders:

Lab Assignment 3 due Thursday at 6 PM

Midterm Exam is Friday, Oct 18, 7–8:30 PM

Thursday

Authentication and passwords

Passwords, password cracking

Next week:

Midterm review