

Content Delivery Networks

- Replication is a huge benefit to availability, scalability, and performance
 - We saw this with DNS
 - Can spread the load
 - Places content closer to clients (less latency)
- Caching is a form of opportunistic replication
 - .. but what if a given organization doesn't have a forward proxy?
 - .. what if content provider and wants its content always replicated?
 - Idea: Caching and replication as a service — “CDNs 1.0”

CDNs “1.0”

- Large-scale distributed storage infrastructure
 - (Usually) administered by one entity
 - e.g., Akamai has 275,000+ servers in 136 countries
- Any server can host content for the many clients of the CDN (virtual hosting)
- How does content provider get its data onto Akamai’s servers?
- Two major ways
 - Pull
 - Push
 - .. we’ll come back to these in a moment

CDNs “1.0” - the basic idea

- Content provider buys service from a CDN, e.g., Akamai
- CDN creates new domain names for the customer content provider
 - e.g., e12596.dscj.akamaiedge.net for cnn.com
 - The CDN's DNS servers are authoritative for the new domains
- Content provider modifies its content so that embedded URLs reference the new domains
 - “Akamaize” content
 - e.g.: <http://www.cnn.com/some-photo.jpg> becomes <http://e12596.dscj.akamaiedge.net/some-photo.jpg>
- Initial request goes to CNN (e.g., for main <http://www.cnn.com> page)
 - .. but embedded links go to Akamai, which handles DNS resolution for URL
 - .. Akamai DNS servers pick one of their 275,000+ servers to serve it
 - (based on IP geolocation, server load, etc.)

CDNs “1.0” - the basic idea

- Content provider buys service from a CDN, e.g., Akamai
- CDN creates new domains
 - e.g., e12596.dscj.akamai.net
 - The CDN's DNS server
- Content provider embeds new domains
 - “Akamaize” content
 - e.g.: <http://www.cnn.com> → <http://e12596.dscj.akamai.net>

DNS Question:

What if a content provider doesn't want to embed a weird Akamai domain into its pages?

Answer:

Add a DNS CNAME record to CNN nameserver (CNAME, cdn.cnn.com, e12596.dscj.akamaiedge.net)

- Initial request goes to CNN (e.g., for main <http://www.cnn.com> page)
 - .. but embedded links go to Akamai, which handles DNS resolution for URL
 - .. Akamai DNS servers pick one of their 275,000+ servers to serve it
 - (based on IP geolocation, server load, etc.)

provider

reference the new

How do you get content onto the CDN servers?

- Pull
 - Akamai servers act like a cache
 - Content provider gives CDN “origin” URL
 - When a client requests from Akamai
 - .. if cached, serve it
 - .. if not cached, request (“pull”) from origin, cache it, serve it
- Push
 - Akamai servers just act like normal servers
 - Content provider uploads content to CDN (“pushes” their content)
 - When a client requests from Akamai, just serve like any web server
- Various tradeoffs
 - Short version: pull is less work for content provider but push gives more control

Web Security

This isn't a security class

- This isn't intended to be a lecture on all crypto
- I want you to appreciate the important principles, understand what's important for TLS (and other protocols like it)

The Internet is insecure

- Designed for simplicity in a naïve era
- Lots of insecure systems that can be compromised
- Attacks look like normal traffic
- Internet's federated operation obstructs cooperation for diagnosis/mitigation

Basic Requirements for Secure Communication

- **Availability:** Will the network deliver data?
 - Infrastructure compromise, DDoS
- **Authentication:** Who is this actor?
 - Spoofing, phishing
- **Integrity:** Do messages arrive in original form?
- **Confidentiality:** Can adversary read the data?
 - Sniffing, man-in-the-middle
- **Provenance:** Who is responsible for this data?
 - Forging responses, denying responsibility
 - Not who sent the data, but who created it

Other desirable security properties

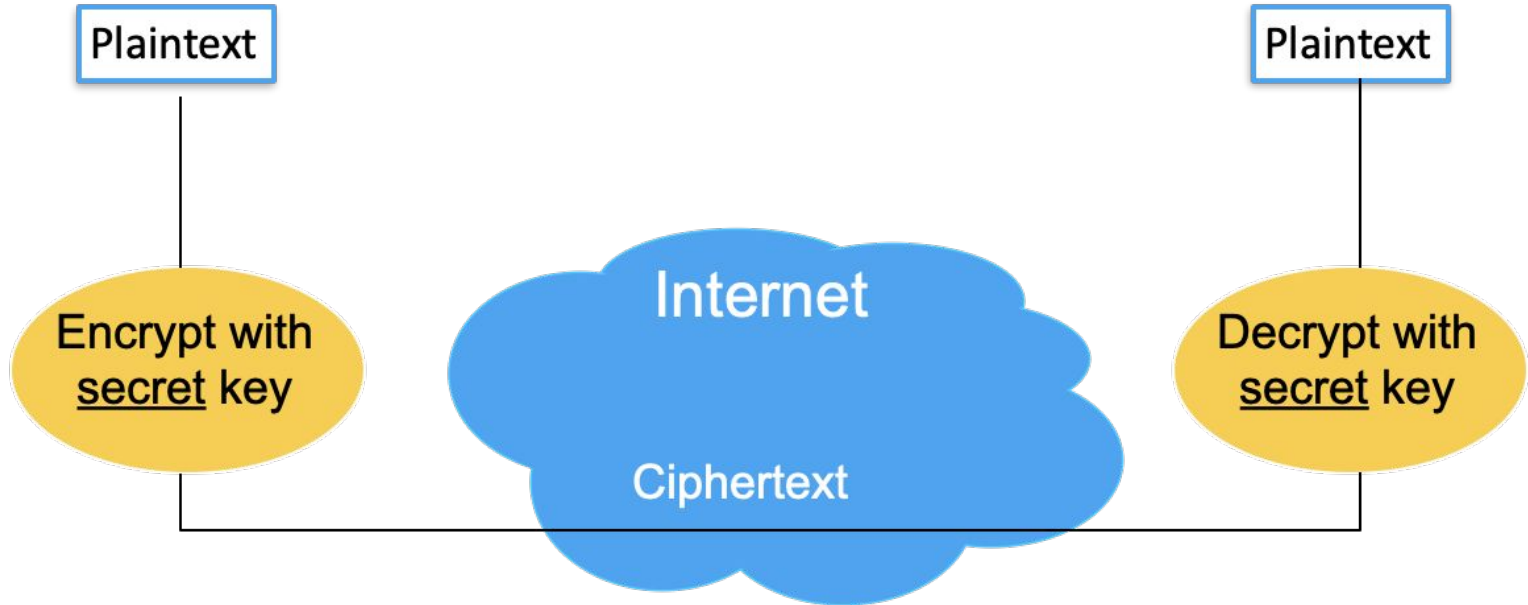
- **Authorization:** is actor allowed to do this action?
 - Access controls
- **Accountability/Attribution:** who did this activity?
- **Audit/Forensics:** what occurred in the past?
 - A broader notion of accountability/attribution
- **Appropriate use:** is action consistent with policy?
 - E.g., no spam; no games during business hours; etc.
- **Freedom from traffic analysis:** can someone tell when I am sending and to whom?
- **Anonymity:** can someone tell I sent this packet?

What is TLS?

- Security for the transport layer
- Bidirectional pipe between two parties (client and server), but can enable:
 - Confidentiality
 - Integrity
 - Authentication
- Is this all the security properties we might want? No

Fundamental crypto: symmetric keys

Both the sender and the receiver use the same secret keys

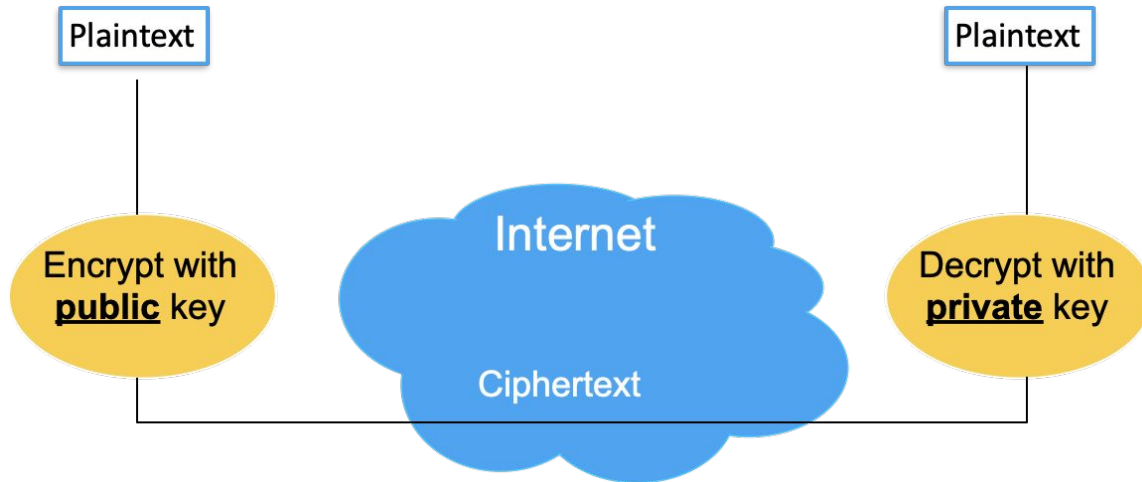


Fundamental crypto: asymmetric encryption (public key)

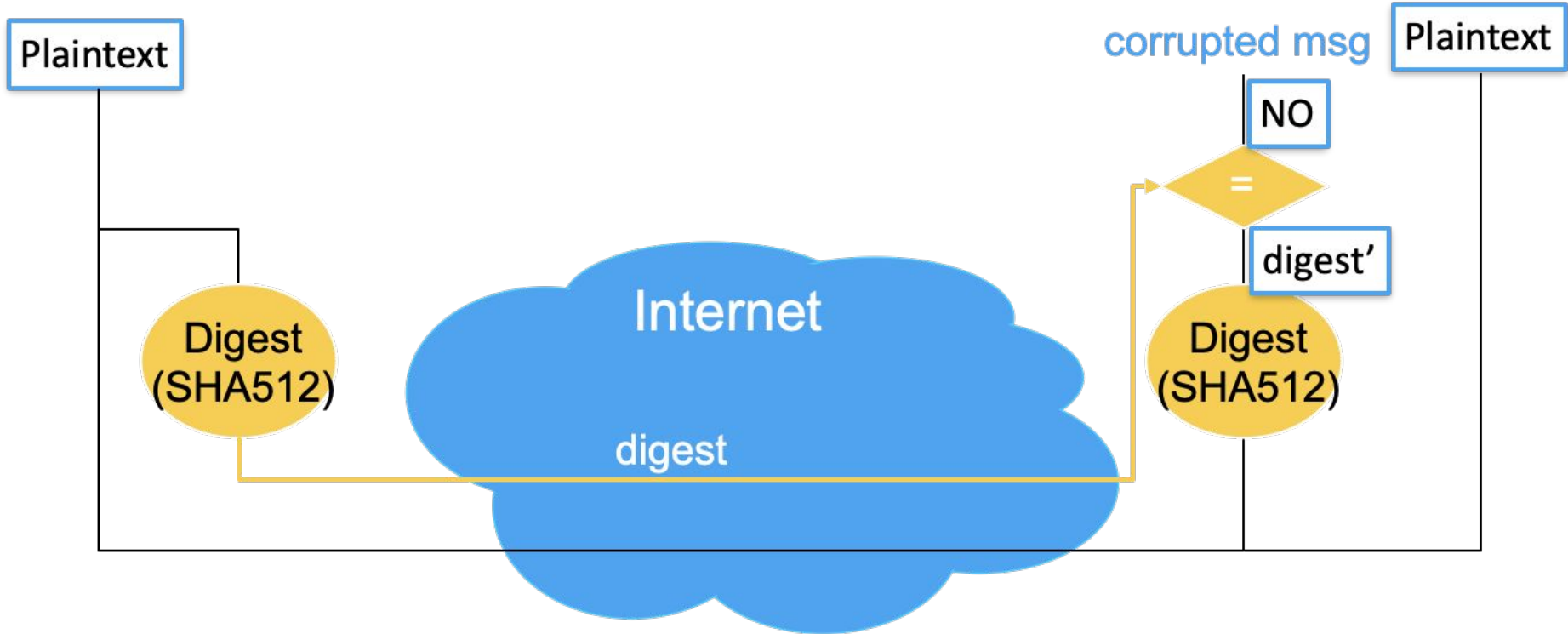
- Idea: use two different keys, one to encrypt (**e**) and one to decrypt (**d**)
 - A key pair
- Crucial property: knowing **e** does not give away **d**
- **e** can be public: everyone knows it
- If Alice wants to send to Bob, she fetches Bob's public key (say from Bob's home page) and encrypts with it
 - Alice can't decrypt what she's sending to Bob ...
 - ... but then, neither can anyone else (except Bob)

Public Key / Asymmetric Encryption

- Sender uses receiver's public key
 - Advertised to everyone
- Receiver uses complementary private key
 - Must be kept secret



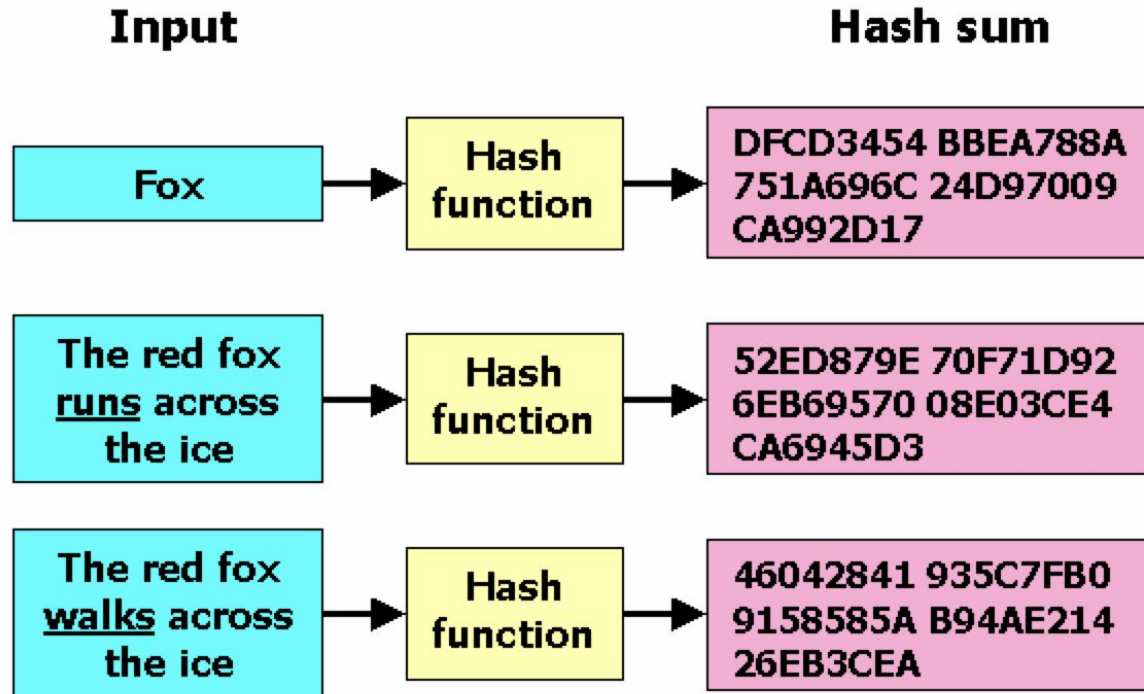
Hashing for Integrity



Cryptographically Strong Hashes

- Hard to find collisions
 - Adversary can't find two inputs that produce same hash
 - Someone cannot alter message without modifying digest
 - Can succinctly refer to large objects
- Hard to invert
 - Given hash, adversary can't find input that produces it
 - Can refer obliquely to private objects (e.g., passwords)
 - Send hash of object rather than object itself

Effects of cryptographic hashes



Public key authentication

Each side need only to know the other side's public key

No secret key need be shared

A encrypts a nonce (random number) x using B's public key

B proves it can recover x

A can authenticate itself to B in the same way

Basic crypto toolkit

- If we can securely distribute a key, then
 - Symmetric ciphers (e.g., AES) offer fast, presumably strong confidentiality
- Public key cryptography can make this easier (can share public keys anywhere)
 - But not as computationally efficient
 - Use public key crypto to exchange **session key**, which is used for symmetric encryption

Public Key Infrastructure (PKI)

- In reality, hierarchy of trust
- Root CAs sign certificates for Intermediate CAs
- Intermediate CAs sign certificates for general users/sites

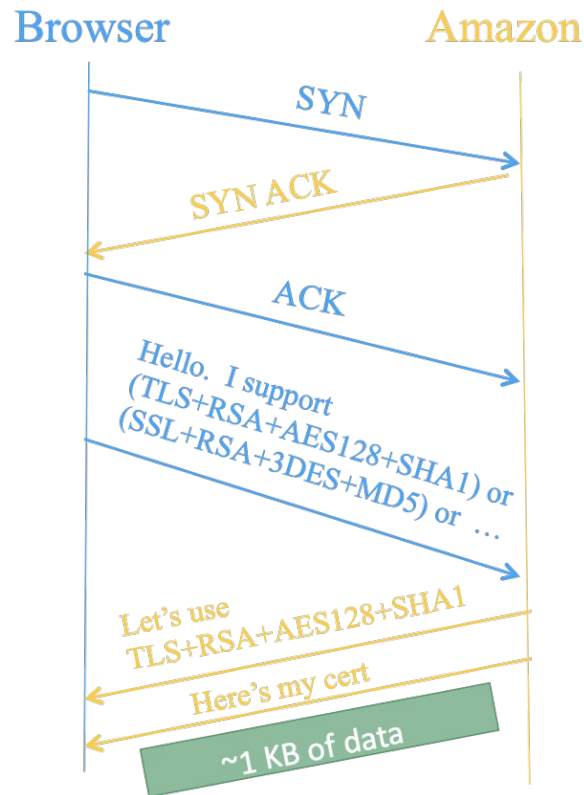
- The further up the hierarchy, the more protections it needs
 - CA's often use Hardware Security Modules (HSMs), other physical protections...
 - What happens if a CA is compromised?

Putting It All Together: HTTPS


- Steps after clicking on `https://www.amazon.com`
- `https` = “Use HTTP over TLS”
 - SSL = Secure Socket Layer (older version)
 - TLS = Transport Layer Security
 - Successor to SSL, and compatible with it
 - RFC 4346, and many others
- Provides security layer (authentication, encryption) on top of transport layer
 - Fairly transparent to the app (once set up)

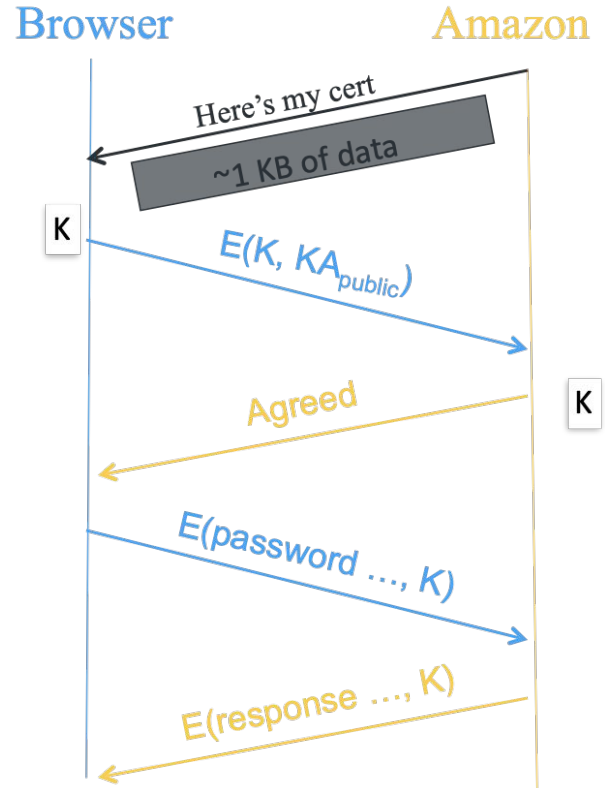
HTTPS Connection (SSL/TLS), con't

- Browser (client) connects via TCP to Amazon's HTTPS server
- Client sends over list of crypto protocols it supports
- Server picks protocols to use for this session
- Server sends over its certificate
- (all of this is in the clear)



HTTPS Connection (SSL/TLS), con't

- Browser constructs a random session key K
- Browser encrypts K using Amazon's public key
- Browser sends $E(K, KA_{\text{public}})$ to server
- Browser displays 
- All subsequent communication encrypted w/ symmetric cipher using key K
 - E.g., client can authenticate using a password



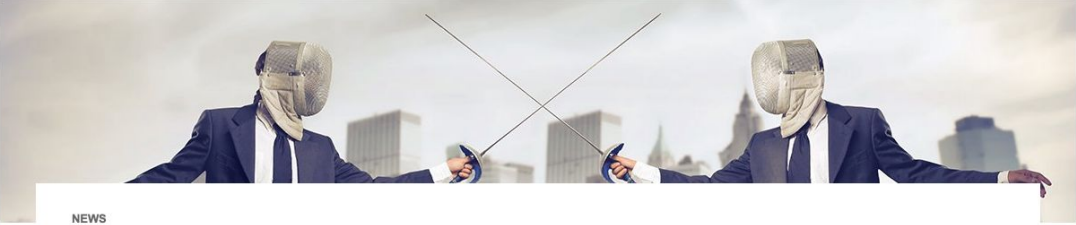
When does this break down?

- TLS is hard to implement
- Need to trust the CAs
- Users need to understand warnings

When does this break down?

- TLS is
- Need
- Users


Home > Data security and privacy OLLY - STOCKADBE.COM








NEWS

Mozilla, Microsoft drop Trustcor as root certificate authority

Mozilla and Microsoft removed support for TrustCor certificates after a Washington Post report revealed the company's ties to government contractors specializing in spyware.

By  Rob Wright, News Director Published: 01 Dec 2022

After weeks of discussions, Mozilla and Microsoft have removed trust for TrustCor Systems' certificates and removed the company from their respective root certificate stores.

The decisions follow an investigate [report](#) from The Washington Post earlier this month that showed TrustCor's apparent connections to spyware vendor Packet Forensics as well as other companies with ties to the U.S. intelligence community. Rachel McPherson, TrustCor's vice president of operations, responded angrily in an open letter, claiming the article was driven by biased security researchers and "filled with ridiculous, false claims and out-of-context statements."

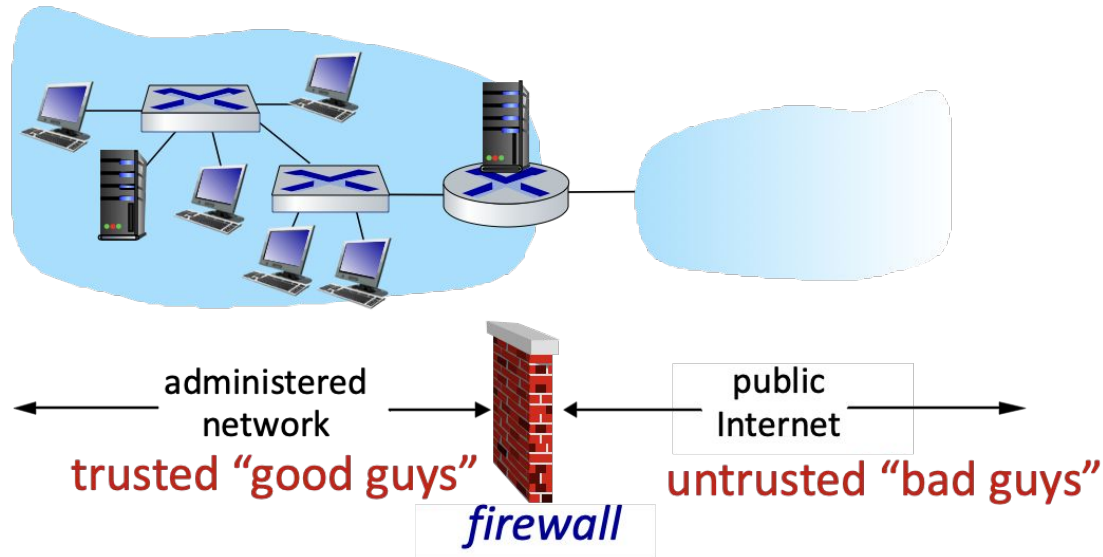
Sponsored News

Built for Business, Built for Now
Intel

Operational Security

Firewalls

- isolates organization's internal network from larger Internet, allowing some packets to pass, blocking others

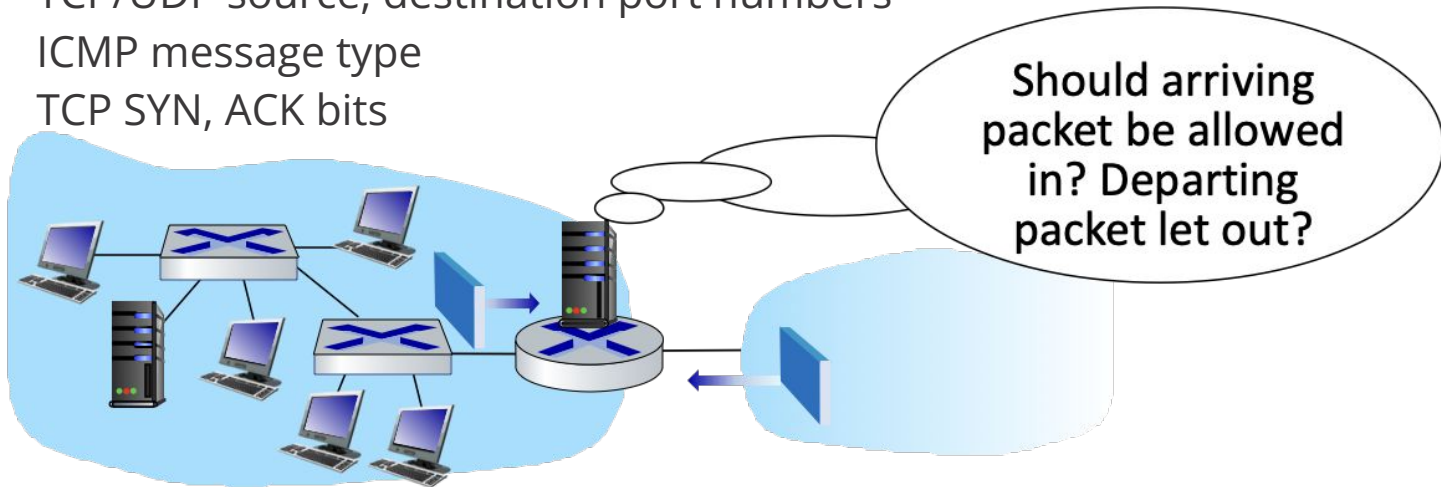


Firewalls

- prevent denial of service attacks:
 - SYN flooding: attacker establishes many bogus TCP connections, no resources left for “real” connections
- prevent illegal modification/access of internal data
 - e.g., attacker replaces CIA’s homepage with something else
- allow only authorized access to inside network
 - set of authenticated users/hosts
- three types of firewalls:
 - stateless packet filters
 - stateful packet filters
 - application gateways

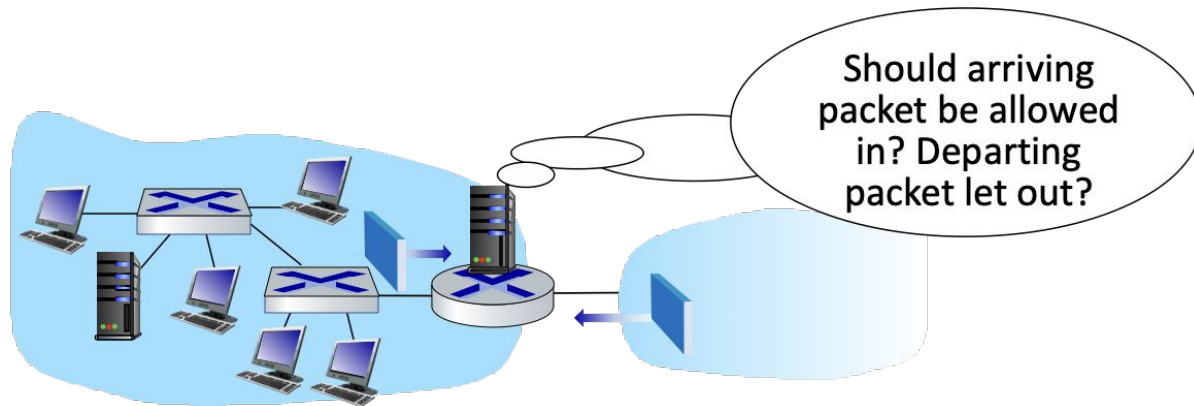
Stateless Packet Filtering

- internal network connected to Internet via router firewall
- filters **packet-by-packet**, decision to forward/drop packet based on:
 - source IP address, destination IP address
 - TCP/UDP source, destination port numbers
 - ICMP message type
 - TCP SYN, ACK bits



Stateless Packet Filtering

- example 1: block incoming and outgoing datagrams with IP protocol field = 17 and with either source or dest port = 23
 - result: all incoming, outgoing UDP flows and telnet connections are blocked
- example 2: block inbound TCP segments with ACK=0
 - result: prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside



Stateful Packet Filtering

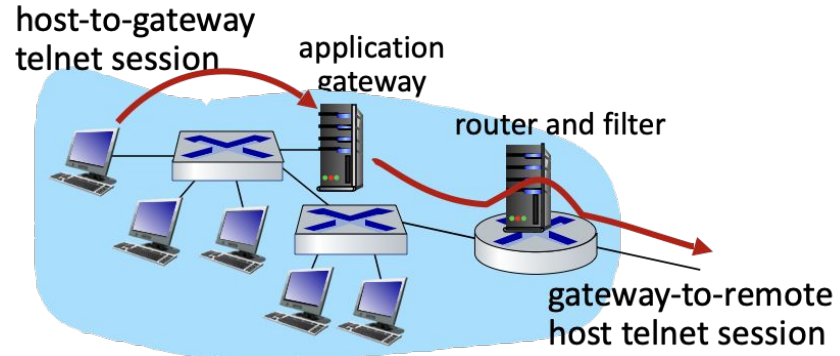
- stateless packet filter: heavy handed tool
 - admits packets that “make no sense,” e.g., dest port = 80, ACK bit set, even though no TCP connection established
- stateful packet filter: track status of every TCP connection
 - track connection setup (SYN), teardown (FIN): determine whether incoming, outgoing packets “makes sense”
 - timeout inactive connections at firewall: no longer admit packets

Application gateways

filter packets on application data as well as on IP/TCP/UDP fields.

example: allow select internal users to telnet outside

1. require all telnet users to telnet through gateway.
2. for authorized users, gateway sets up telnet connection to dest host
 - a. gateway relays data between 2 connections
3. router filter blocks all telnet connections not originating from gateway



Limitations of firewalls, gateways

- IP spoofing: router can't know if data "really" comes from claimed source
- if multiple apps need special treatment, each has own app. gateway
- client software must know how to contact gateway
 - e.g., must set IP address of proxy in Web browser
- filters often use all or nothing policy for UDP
- tradeoff: degree of communication with outside world, level of security
- many highly protected sites still suffer from attacks

Intrusion detection systems

- packet filtering:
 - operates on TCP/IP headers only
 - no correlation check among sessions
- **IDS: intrusion detection system**
 - **deep packet inspection**: look at packet contents (e.g., check character strings in packet against database of known virus, attack strings)
 - **examine correlation** among multiple packets
 - port scanning
 - network mapping
 - DoS attack