Computer Security

11. Firewalls & VPNs

Paul Krzyzanowski
Rutgers University
Spring 2017
Conversation Isolation
Fundamental Layer 2 & 3 Problems

• IP relies on store-and-forward networking
  – Network data passes through untrusted hosts
  – Routes may be altered to pass data through malicious hosts

• Packets can be sniffed

• TCP session state can be examined or guessed …
  … and TCP sessions can be hijacked

• No source authentication on IP packets
Solution: Use private networks

Connect multiple geographically-separated private subnetworks together

192.168.1.0/24

Gateway Router

Private network line

Gateway Router

192.168.2.0/24

Internal subnet

But this is expensive … and not feasible in many cases (e.g., cloud servers)
Tunneling

Tunnel = Packet encapsulation
Treat an entire IP datagram as payload on the public network

192.168.1.0/24
192.168.2.0/24

Internal subnet

Gateway Router
68.36.210.57
Gateway Router
128.6.4.2

Internet

Src: 192.168.1.11
Dest: 192.168.2.22
Data: [--------]

Src: 68.36.210.57
Dest: 128.6.4.2
Data: [--------]

From: 192.168.1.11
To: 192.168.2.22
Data: [--------]

Src: 192.168.1.11
Dest: 192.168.2.22
Data: [--------]
Tunnel mode vs. transport mode

• Tunnel mode
  – Communication between gateways: network-to-network
  – Or host-to-network
  – Entire datagram is encapsulated

• Transport mode
  – Communication between hosts
  – IP header is not modified
IPsec

• Internet Protocol Security

• End-to-end solution at the IP layer

• Two protocols:
  – **IP Authentication Header** Protocol (AH)
    • Authentication & integrity of payload and header
  – **Encapsulating Security Payload** (ESP)
    • AH + Confidentiality of payload
IPsec Authentication Header (AH)

Guarantees integrity & authenticity of IP packets
- MAC for the contents of the entire IP packet
- Over unchangeable IP datagram fields (e.g., not TTL or fragmentation)

Protects from:
- Tampering
- Forging addresses
- Replay attacks (signed sequence number in AH)

Layered directly on top of IP (protocol 51) - not UDP or TCP
IPsec Encapsulating Security Payload (ESP)

Encrypts entire payload

– Plus authentication of payload + IP header (everything AH does)
  (may be optionally disabled – but you don’t want to)

Directly on top of IP (protocol 51) - not UDP or TCP
IPsec algorithms

• Integrity protection & authenticity
  – HMAC-SHA1
  – HMAC-SHA2

• Confidentiality
  – 3DES-CBC
  – AES-CBC

• Authentication
  – Kerberos, certificates, or pre-shared key authentication

• Key generation
  – Diffie-Hellman to exchange keying material for key generation
  – Key lifetimes determine when new keys are regenerated
  – Perfect forward secrecy
    • “Main mode master key PFS” – requires reauthentication & is CPU-intensive
    • “Quick mode session key PFS” – no reauthentication
Conversation Isolation: Transport Layer
SSL/TLS
Transport Layer Security

- Provide a transport layer security protocol
- After setup, applications feel like they are using TCP sockets

SSL: Secure Socket Layer

- Created with HTTP in mind
  - Web sessions should be secure
  - Mutual authentication is usually not needed
    - Client needs to identify the server but the server won’t know all clients
    - Rely on passwords after the secure channel is set up

- SSL evolved to TLS (Transport Layer Security)
  - SSL 3.0 was the last version of SSL … and is considered insecure
  - We use TLS now … but often still call it SSL
TLS Protocol

• Goal
  – Provide authentication (usually one-way), privacy, & data integrity between two applications

• Principles
  – Use symmetric cryptography to encrypt data
  – Keys generated uniquely at the start of each session
  – Include a MAC with transmitted data to ensure message integrity
  – Use public key cryptography & X.509 certificates for authentication
    • Optional – can authenticate 0, 1, or both parties
  – Support many different key exchange, encryption, integrity, & authentication protocols – negotiate what to use at the start of a session
TLS Protocol & Ciphers

Two sub-protocols
1. Authenticate & establish key
2. Communicate
   • HMAC used for message authentication

• Key exchange
  – Public keys (RSA or Elliptic Curve)
  – Diffie Hellman keys
  – Ephemeral Diffie-Hellman keys (generated for each session)
  – Pre-shared key

• Data encryption
  – AES GCM, AES CBC, ARIA (GCM/CBC), ChaCha20-Poly1305, …

• Data integrity
  – HMAC-MD5, HMAC-SHA1, HMAC-SHA256/384, …
TLS Protocol

(1) Client hello
Version & crypto information

(2) Server hello
Server certificate
[client certificate request]

(3) Verify server certificate

(4) Client key exchange
Send encrypted session key

(5) Send client certificate

(7) Client done

(8) Server done

(9) Communicate
Symmetric encryption + HMAC

[ (6) Verify server certificate ]

April 17, 2017
CS 419 © 2017 Paul Krzyzanowski
Benefits of TLS

• Benefits
  – Protects integrity of communications
  – Protects the privacy of communications
  – Validates the authenticity of the server (if you trust the CA)
Problems with TLS

• Attacks
  – Man-in-the-middle: BEAST attack in TLS 1.0
    • Attacker was able to see Initialization Vector (IV) for CBC and deduce plaintext (known HTML headers & cookies)
    • Fixed by using explicit IVs for each new block
  – Man-in-the-middle: crypto renegotiation
    • Attacker can renegotiate the handshake protocol to disable encryption
    • Proposed fix: have client & server verify info about previous handshakes
  – THC-SSL-DoS attack
    • Attacker initiates a TLS handshake & requests a renegotiation of the encryption key – repeat over & over, using up server resources
Problems with TLS

- Client authentication Problem
  - Client authentication is almost never used
    - Generating keys & obtaining certificates is not an easy process
    - Any site can request the certificate: user will be unaware anonymity is lost
    - Moving private keys around can be difficult (what about public systems?)
  - We usually rely on other authentication mechanisms
    (usually user name and password)
Firewalls
Network Security Goals

- **Confidentiality**: sensitive data & systems not accessible
- **Integrity**: data not modified during transmission
- **Availability**: systems should remain accessible
Firewall

• Separate your local network from the Internet
  – Protect the border between trusted internal networks and the untrusted Internet

• Approaches
  – Packet filters
  – Application proxies
  – Intrusion detection / intrusion protection systems
Screening router

• **Border router** (gateway router)
  – Router between the internal network(s) and external network(s)
  – Any traffic between internal & external networks passes through the border router

Instead of just routing the packet, decide *whether* to route it

• **Screening router** = Packet filter
  Allow or deny packets based on
  – Incoming interface, outgoing interface
  – Source IP address, destination IP address
  – Source TCP/UDP port, destination TCP/UDP port, ICMP command
  – Protocol (e.g., TCP, UDP, ICMP, IGMP, RSVP, etc.)
Filter chaining

• An IP packet entering a router is matched against a set of rules: access control list (ACL) or chain

• Each rule contains criteria and an action
  – Criteria: packet screening rule
  – Actions
    • Accept – and stop processing additional rules
    • Drop – discard the packet and stop processing additional rules
    • Reject – and send an error to the sender (ICMP Destination Unreachable)
  – Also
    • Route – rereoute packets
    • Nat – perform network address translation
    • Log – record the activity
Filter structure is vendor specific

Examples

– Windows
  • *Allow*, *Block*
  • Options such as
    – Discard all traffic except packets allowed by filters (*default deny*)
    – Pass through all traffic except packets prohibited by filters (*default allow*)

– OpenBSD
  • *Pass* (allow), *Block*

– Linux nftables (netfilter)
  • Chain types: *filter*, *route*, *nat*
  • Chain control
    – *Return* – stop traversing a chain
    – *Jump* – jump to another chain (*goto* = same but no return)
Network Ingress Filtering: incoming packets

Basic firewalling principle

Never have a direct inbound connection from the originating host from the Internet to an internal host – all traffic must flow through a firewall and be inspected

• Determine which services you want to expose to the Internet
  – e.g., HTTP & HTTPS: TCP ports 80 and 443

• Create a list of services and allow only those inbound ports and protocols to the machines hosting the services.

• Default Deny model - by default, "deny all"
  – Anything not specifically permitted is dropped
  – May want to log denies to identify who is attempting access
Network Ingress Filtering

• Disallow IP source address spoofing
  – Restrict forged traffic (RFC 2827)

• At the ISP
  – Filter upstream traffic - prohibit an attacker from sending traffic from forged IP addresses
  – Attacker must use a valid, reachable source address

• Disallow incoming/outgoing traffic from private, non-routable IP addresses
  – Helps with DDoS attacks such as SYN flooding from lots of invalid addresses

```
access-list 199 deny ip 192.168.0.0 0.0.255.255 any log
access-list 199 deny ip 224.0.0.0 0.0.0.255 any log
....
access-list 199 permit ip any any
```
Network Egress Filtering (outbound)

• Usually we don’t worry about outbound traffic.
  – *Communication from a higher security network (internal) to a lower security network (Internet) is usually fine*

• Why might we want to restrict it?
  – Consider: if a web server is compromised & all outbound traffic is allowed, it can connect to an external server and download more malicious code ... or launch a DoS attack on the internal network

  – Also, log which servers are trying to access external addresses
Stateful Inspection

• Retain state information about a stream of related packets

• Examples
  – TCP connection tracking
    • Disallow TCP data packets unless a connection is set up
  – ICMP echo-reply
    • Allow ICMP echo-reply only if a corresponding echo request was sent.
  – Related traffic
    • Identify & allow traffic that is related to a connection
    • Example: related ports in FTP
Network Design: DMZ

Internal subnet

Security Appliance (screening router)

DMZ subnet

Internet

Network Design: DMZ

Clients from the Internet:
- Can access allowed services in the DMZ
- Cannot access internal hosts

The router:
- Blocks impersonated packets

Security Appliance (screening router)

Internet

DMZ subnet

Internal subnet

Network Design: DMZ

Clients in the internal subnet:
- Can access the Internet
- Can access allowed services in the DMZ
- May access extra services in the DMZ

Secure Appliance
(screening router)

DMZ subnet

Internet
Clients in the DMZ:
- Can access Internet services only to the extent required
- Can access internal services only to the extent required

Goal: 
Limit possible damage if DMZ machines are compromised
Network Address Translation

- Most organizations use private IP addresses

- External traffic goes through a NAT router
  - Network Address Translation

- NAT is an implicit firewall (sort of)
  - Arbitrary hosts and services on them (ports) cannot be accessed unless
    - They are specifically mapped to a specific host/port by the administrator
    - Internal services have initiated outgoing traffic
      - Return traffic from the same address/port will be accepted
Application-Layer Filtering

• Firewalls don’t work well when everything’s a web service

• Deep packet inspection
  – Look beyond layer 3 & 4 headers
  – Need to know something about application protocols & formats

• Example
  – URL filtering
    • Normal source/destination host/port filtering +
      URL pattern/keywords, rewrite/truncate rules, protocol content filters
    • Detect ActiveX and Java applets; configure specific applets as trusted
      – Remove others from the HTML code
  – Keyword detection
    • Prevent classified material from leaving the organization
    • Prevent banned content from leaving or entering an organization
IDS/IPS

• Intrusion Detection/Prevention Systems
  – Identify threats and attacks

• Types of IDS
  – Protocol-based
  – Signature-based
    • We know what is bad; anything else is good
  – Anomaly-based
    • We know what is good; anything else is bad
Protocol-Based IDS

• Reject packets that do not follow a prescribed protocol
• Permit return traffic as a function of incoming traffic
• Define traffic of interest (filter), filter on traffic-specific protocol/patterns

• Examples
  – **DNS inspection**: prevent spoofing DNS replies: make sure they match IDs of sent DNS requests
  – **SMTP inspection**: restrict SMTP command set (and command count, arguments, addresses)
  – **FTP inspection**: restrict FTP command set (and file sizes and file names)
Signature-based IDS

- Don't search for protocol violations but for exploits in programming
- Match patterns of known “bad” behavior
  - Viruses
  - Malformed URLs
  - Buffer overflow code
Anomaly-based IDS

• Search for statistical deviations from normal behavior
  – Measure baseline behavior first

• Examples:
  – Port scanning
  – Imbalance in protocol distribution
  – Imbalance in service access
Application proxies

- Proxy servers
  - Intermediaries between clients and servers
  - Stateful inspection and protocol validation

- Dual-homed host
- Bastion host
Deperimeterization

- Boundaries & access between internal & external systems are harder to identify
  - Mobile systems
  - Cloud-based computing
  - USB flash memory
  - Web-based applications
Host-based firewalls

• Run on the user’s systems, not as dedicated firewalls

• Manage network-facing effects of malware
  – Allow only approved applications to send or receive data over the network

• Problem
  – If malware gets elevated privileges, it can reconfigure or disable the firewall

• Personal IDS
  – E.g., **fail2ban** on Linux
    • Scan log files to detect & ban suspicious IP addresses
    • High number of failed logins, probes, URLs that try to target exploits
Intrusion detection & prevention problems

• There’s a lot of stuff going on
  – People visit random websites with varying frequencies
  – Software accesses varying services
  – Buggy software may create bad packets
  – How do you detect what is hostile?

• Attack rates is miniscule … compared to legitimate traffic
  – Even a small % of false positives can be annoying and hide true threats

• Environments are dynamic
  – Content from CDNs or other large server farms has a broad range of IP addresses
  – Malicious actors can coexist with legitimate ones
Intrusion detection & prevention problems

• Encrypted traffic cannot be easily inspected
  – Just because you visit a web site using HTTPS doesn’t mean the site is secure … or hasn’t been compromised

• Packet inspection is limiting
  – You may need to reconstruct sessions, which is time consuming

• Threats & services change
  – Rules have to be updated
<table>
<thead>
<tr>
<th>Security Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firewall (screening router)</td>
<td>1&lt;sup&gt;st&lt;/sup&gt; generation packet filter that filters packets between networks. Blocks/accepts traffic based on IP addresses, ports, protocols</td>
</tr>
<tr>
<td>Stateful inspection firewall</td>
<td>Like a screening router but also takes into account TCP connection state and information from previous connections (e.g., related ports for TCP)</td>
</tr>
<tr>
<td>Application proxy</td>
<td>Gateway between two networks for a specific application. Prevents direct connections to the application from outside the network. Responsible for validating the protocol.</td>
</tr>
<tr>
<td>IDS/IPS</td>
<td>Can usually do what a stateful inspection firewall does + examine application-layer data for protocol attacks or malicious content</td>
</tr>
<tr>
<td>Host-based firewall</td>
<td>Typically screening router with per-application awareness. Sometimes includes anti-virus software for application-layer signature checking</td>
</tr>
<tr>
<td>Host-based IPS</td>
<td>Typically allows real-time blocking of remote hosts performing suspicious operations (port scanning, ssh logins)</td>
</tr>
</tbody>
</table>
DDoS
DDoS: Distributed Denial of Service

• Compromise machines (create a botnet)
  – Use *amplification* techniques to generate a lot of traffic for targets
    • Exploit services that generate a lot of traffic to a small query
    • DNS amplification:
      Small UDP query with forged source address results in large response

• Some targets were too huge to hurt with traffic
  – Amazon, Google, sites using CDNs such as Akamai

• Vast quantities of compromised systems reduce need for amplification
  – Create a botnet of millions of systems
Dealing with DDoS

Really difficult in general

- Bandwidth management routers
  - Either in data center or ISP
  - Limit outbound or inbound traffic on a per-IP basis

- Detect DNS attack and set null routing
  - Traffic to attacked DNS goes nowhere

- Egress filtering by ISPs
  - Attempt to find malicious hosts participating in DDoS or sending spam

- Identify incoming attackers & block traffic at firewall
  - Difficult with a truly distributed DDoS attack
The end