CSE508 Network Security

2/15/2016 Firewalls and Gateways

Michalis Polychronakis
Stony Brook University
Typical Network Topology
**Firewalls:** separate local networks from the Internet
Firewalls: Reality
Firewalls

Filter traffic according to a predefined *policy*

- Mostly statically defined, but dynamic updates are possible (e.g., to block an ongoing DoS attack)

Barrier between administrative domains

- Internal networks vs. the outside world
- Mission-specific subnets/VLANs (publicly accessible servers, machine clusters, user groups, printers, VoIP, …)
- Less trusted segments (guest WiFi network, contractors, …)

Main strategies

- Default-deny: drop everything unless explicitly allowed
- Default-allow: block specific protocols/hosts/ports/…
Firewalls: why are they needed?

Hosts may run vulnerable services: prevent outside attackers from accessing them
   Limit the “attack surface” ➔ expose less services

Internal hosts may get compromised: damage control
   Prevent propagation, outgoing attacks, exfiltration, …

No reason to reveal the structure of private networks: hinder network reconnaissance
   Block port scanning, service fingerprinting, …

Network intelligence: log interesting events
   Troubleshooting, monitoring/tuning, auditing, forensics, …

Simply block unwanted traffic: **policy enforcement**
   Noise, backscatter, spoofed packets, DoS attacks, brute-force password guessing, Bittorrent, Facebook, …
Stateless Filtering

Decide by considering each packet in isolation
  Rules mostly based on network and transport layer fields
  Simple implementation: no need to keep state

Limitations
  Dynamically negotiated/non-standard port numbers
    (FTP, SIP, BitTorrent, …)
  Connectionless protocols (e.g., UDP): cannot distinguish
    between queries and replies
  IP fragmentation: port numbers are present only in 1st fragment
  Rule sets can get complex and hard to understand

Still useful for simple scenarios
  Ingress/egress filtering, strict configurations, …
Stateless Firewalls and TCP

Common configuration: block incoming but allow outgoing connections
   - Incoming (externally initiated) connections should be blocked
   - Incoming packets of established connections should be allowed

Can be achieved without keeping state
   - Block incoming SYN-only packets
   - Allow incoming packets with the ACK bit set

Not a perfect solution
   - ACK-scanning (nmap -sA) can determine whether a stateless firewall is used
   - SYN ➔ SYN/ACK, ACK ➔ no response: stateful filter
   - SYN ➔ no response, ACK ➔ RST: stateless filter
Stateful Filtering

Firewall keeps per-connection state

Track TCP three-way handshake, UDP query/responses, …
Decisions are made by considering each packet in the context of the connection/session it belongs to

Most common firewall type

More flexible policies

Internally vs. externally initiated connections/sessions

Still cannot handle dynamically negotiated port numbers and higher-level protocol semantics

Missing application-level context
Network Address Translation

Share a public IP address with many internal hosts

In general: remap an IP address space into another
Global shortage of IPv4 addresses
Widely used (home networks, wireless networks, …)

Rewrite packet address and port information
Keep per-connection state
NAT vs. Stateful Firewall

Similar functionality and state

- NAT additionally performs address/port translation
- Typically consolidated into the same device

Implicit default configuration: allow only outgoing connections

Internal hosts can become accessible through *port forwarding*

- Explicitly map a local IP:port to a public IP:port

![Diagram showing NAT box with IP addresses and ports](image)

- SSH server: 192.168.0.10:1234
- Web server: 192.168.0.10:80
- NAT box: 130.245.27.2:22, 130.245.27.2:80
- Internet IP: 130.245.27.2

![Cloud icon representing Internet](image)
UPnP

Universal Plug and Play

Widely supported protocol by home routers to enable device discovery and NAT traversal

“Please allow external hosts to reach me on port 12345”

Skype, Bittorrent, games, …

No authentication!

Malware can easily punch holes

Worse: Flash, XSS, …

Even worse: external requests (!)
Security Flaws in Universal Plug and Play: Unplug, Don't Play

This morning we released a whitepaper entitled Security Flaws in Universal Plug and Play. This paper is the result of a research project spanning the second half of 2012 that measured the global exposure of UPnP-enabled network devices. The results were shocking to the say the least. Over 80 million unique IPs were identified that responded to UPnP discovery requests from the internet. Somewhere between 40 and 50 million IPs are vulnerable to at least one of three attacks outlined in this paper. The two most commonly used UPnP software libraries both contained remotely exploitable vulnerabilities. In the case of the Portable UPnP SDK, over 23 million IPs are vulnerable to remote code execution through a single UDP packet. All told, we were able to identify over 6,900 product versions that were vulnerable through UPnP. This list encompasses over 1,500 vendors and only took into account devices that

- 2.2% of public IPv4 addresses respond to UPnP discovery requests from the Internet.
- 81 million unique IP addresses respond to UPnP discovery requests, slightly more than all IPs allocated to Canada.
- 20% of those 81 million systems also expose the SOAP API to the internet at large. This service can allow an attacker to target systems behind the firewall.
- 4 software development kits account for 73% of all discovered UPnP instances.
- 332 products use MiniiPnPd version 1.0, which is remotely exploitable. Over 69% of all MiniiPnPd fingerprints were version 1.0 or older.
- 23 million fingerprints match a version of libupnp that exposes the system to remote code execution.
- 1 UDP packet is all it takes to exploit any of the 8 newly-discovered libupnp vulnerabilities. This packet can be spoofed.

breach compliance
cybersecurity exploit federal metasploit
microsoft network-security newsletter nexpose
patch-tuesday pci rapid7
security social-engineering
Generic Port Forwarding

Bypass firewall policies!

Example: connect from a private network to a host that is blocked by a local firewall
   Remote host: nc -l -p 12345 -c ‘nc blocked.com 80’
   Local host: wget remote.edu:12345

Or using SSH local port forwarding
   ssh -L 12345:blocked.com:80 remote.edu

Also the other way around: remote port forwarding

Example: allow public access to a server running in a private network
   ssh -R 8080:localhost:80 remote.edu
Proxies

Intermediate “stepping stones”
  Operate at the application layer
  Act as both a client and a server

Application-level filtering
  Example: HTTP-level filtering (domains, URLs, …)

Many non-security uses as well
  HTTP content caching (one of the first uses of web proxies)
  Reverse proxies (in front of application servers): quickly serve the same dynamically-generated content
  Transcoding

Explicit vs. transparent proxies
  The former require application configuration
SOCKS Proxies

Also known as circuit-level gateways

Socket Secure (SOCKS): protocol for generic forwarding of packets through a proxy

Supported by many applications/protocols
  HTTP, FTP, SMTP, POP3, NNTP, …

Example: dynamic application-level port forwarding

  ssh -D 12345 sshserver.com
  chrome --proxy-server='socks://localhost:12345'
Application-level Firewalls

Similar to proxies, but less generic
  Application-specific filtering
  Often built into applications

Example: SMTP
  Spam filtering, phishing detection, attachment scanning, …

Begin to overlap with the area of intrusion detection

Recent buzzword: web application firewalls (WAF)
  Server-side HTTP filtering for common attack patterns
    (XSS, SQL injection, …)
  A specific instance of application-level filtering/scanning
Host-based Firewalls

Firewalls running on end hosts
  Windows firewall
  IPtables

“Personal” firewalls: apply common-sense policies (deny incoming, allow outgoing)
  Particularly important for home users, laptops, etc.

On-by-default deployment contributed significantly in ending the era of internet worms
Simple IPtables Example

# flush all chains
iptables -F
iptables -X

# defaults for predefined chains
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# allow anything on localhost interface
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# allow all traffic from specific subnets
iptables -A INPUT -s 128.59.0.0/255.255.0.0 -j ACCEPT
iptables -A INPUT -s 160.39.0.0/255.255.0.0 -j ACCEPT
Simple IPtables Example

# allow all inbound traffic for specific services
iptables -A INPUT -p tcp -m tcp --syn --dport 22 -j ACCEPT
iptables -A INPUT -p tcp -m tcp --syn --dport 80 -j ACCEPT

# allow inbound established and related outside communication
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

# allow ICMP
iptables -A INPUT -p icmp -j ACCEPT

# allow all outgoing traffic
iptables -A OUTPUT -j ACCEPT

Is that a good idea?
Before Host-based Firewalls:

After Host-based Firewalls:
Per-process Firewall

Most “personal” firewalls still allow all outgoing traffic by default

Severe usability problems otherwise

Do all programs really need to communicate with the outside world?

Deny by default and whiltelist only what is needed

No easy solution for this in most OSes – need to rely on hacks or third party solutions
Virtual Private Networks

Users may not always be behind the firewall, but still need full access to an internal network

  Offices at different locations, employees on the move, remote access to home “cloud,” …

VPNs bridge private networks across a public (untrusted) network

  Virtual point-to-point secure connections (encryption)
  Create a *trusted* shared network among them

Remote host/network virtually becomes part of the local network
VPN Examples

Regional Office

Regional Office

Internet

Head-office

Remote / roaming users
VPN Implementations

Tunneling/encapsulation: packets of one network protocol are transferred as data over another protocol

Three major families in wide use today:

PPTP: L2, commonly used in Windows
  Broken

IPsec: L3, widely supported
  Authenticate and encrypt IP packets of a communication session
  Completely transparent to applications
  Tunnel is handled directly by the TCP/IP stack

SSL: Application layer – OpenVPN
  User-space implementation, multiplatform
  Typically requires installation of a software client
“Secure Gateways”

Nowadays most of the discussed technologies are consolidated into a single box

Routing, Firewall, NAT, VPN, Proxy, …

Common in home and enterprise settings

Routers and firewalls used to be “simple” devices – not anymore

Features ➔ complexity ➔ security issues

Critical hosts in the network that need to be protected

Administrative interface, OS patches/updates, service vulnerabilities, …
Welcome to the internet's largest and most updated default router passwords database.

Select Router Manufacturer:

CISCO

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Model</th>
<th>Protocol</th>
<th>Username</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>CISCO</td>
<td>CACHE ENGINE</td>
<td>CONSOLE</td>
<td>admin</td>
<td>diamond</td>
</tr>
<tr>
<td>CISCO</td>
<td>CONFIGMAKER</td>
<td></td>
<td>cmaker</td>
<td>cmaker</td>
</tr>
<tr>
<td>CISCO</td>
<td>CNR Rev. ALL</td>
<td>CNR GUI</td>
<td>admin</td>
<td>changeme</td>
</tr>
<tr>
<td>CISCO</td>
<td>NETRANGER/SECURE IDS</td>
<td>MULTI</td>
<td>netranger</td>
<td>attack</td>
</tr>
<tr>
<td>CISCO</td>
<td>BBSM Rev. 5.0 AND 5.1</td>
<td>TELNET OR NAMED PIPEs</td>
<td>bbsd-client</td>
<td>changeme2</td>
</tr>
<tr>
<td>CISCO</td>
<td>BBSD MSDE CLIENT Rev. 5.0 AND 5.1</td>
<td>TELNET OR NAMED PIPEs</td>
<td>bbsd-client</td>
<td>NULL</td>
</tr>
</tbody>
</table>
Owning Modems And Routers Silently

Modems

Do you have cable internet? Own a surfboard modem? Since most of my buddies in AZ do, I sent them to this page and to my amusement, they got knocked off the net for a few minutes. How? Javascript. Specifically a CSRF in the Motorola Surfboard.

The Surfboard cable modem offers little in functionality besides rebooting unless of course I wanted to be malicious and remove all settings on the cable modem and essentially turn it into a door stop until the thing can be activated again by the ISP.

![Cable Modem Configuration](image.png)

**Configuration**

- **Frequency Plan**: North American Standard/HRC/IRC
- **Custom Frequency Ordering**: Default
- **Upstream Channel ID**: 2
- **Favorite Frequency (Hz)**: 25000000
- **DOCSIS MIMO**: Honor MDD IP Mode
- **Modem’s IP Mode**: IPv4 Only
- **DHCP Server Enabled**: The SURFboard cable modem can be used as a gateway to the internet by a maximum of 32 users on a Local Area Network (LAN). When the Cable Modem is disconnected from the internet, users on the LAN can be dynamically assigned IP addresses by the Cable Modem DHCP Server. These addresses are assigned from an address pool which begins with 192.168.100.11 and ends with 192.168.100.42. Statically assigned IP addresses for other devices on the LAN should be chosen from outside of this range.

**Note:**

Resetting the cable modem to its factory default configuration will remove all stored parameters learned by the cable modem during prior initializations. The process to get back online from a factory default condition could take from 5 to 30 minutes. Please reference the cable modem User Guide for details on the power up sequence.
Discussion Topic: Port Knocking

Open firewalled ports on demand by “knocking” the right combination of ports

- Firewall opens the port once connection attempts to the right combination of ports are seen
- Variation: single packet authorization
- Sometimes recommended for securing SSH servers etc.

*Is port knocking useful or pointless?*