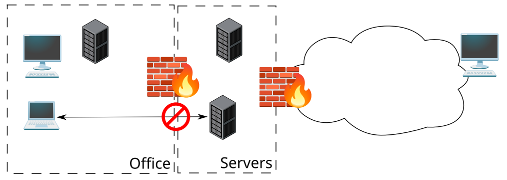
CSci 4271W
Development of Secure Software Systems
Day 15: Networking and security: firewalls and intrusion detection

Stephen McCamant (he/him)
University of Minnesota, Computer Science & Engineering

Firewalls

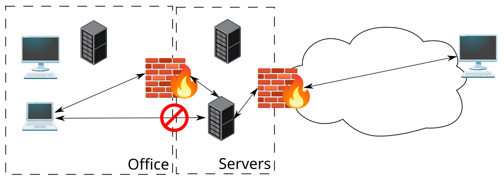
Firewalls represent a physical/logical mechanism to enforce a trust boundary in a networked system:



Traffic crossing the trust boundary must go through the firewall.

Firewalls

Firewalls represent a physical/logical mechanism to enforce a trust boundary in a networked system:



Traffic crossing the trust boundary must go through the firewall.

Firewalls: why?

A firewall might be used to mitigate:

- Spoofing:
- Tampering:
- Information disclosure:
- Elevation of privilege:

Firewalls: why?

A firewall might be used to mitigate:

- Spoofing: stop packets that affect routing decisions, DNS, or with forged source address from crossing
- Tampering: prevent injection of data into local flows
- Information disclosure: prevent unintended flows, conceal network details
- Elevation of privilege: defense-in-depth against applications with remote-to-local EoP bugs

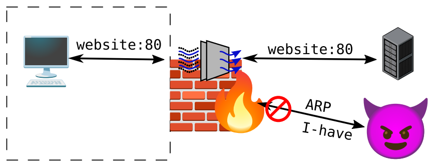
Types of firewalls

- Network layer firewalls filter packets on IP, port, and protocol
- Proxy firewalls can filter on transport or application-layer headers
- Network address translation (NAT) firewalls filter on transport headers, hide local addresses
- Software firewalls filter traffic on a local host

Packet filtering

Network layer firewalls filter packets on IP, port, and protocol

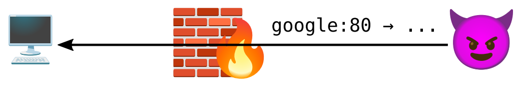
Examples: only allow TCP, UDP from external network, allow `external:*` to `webserver:80`, allow `internal:*` to `external:80`



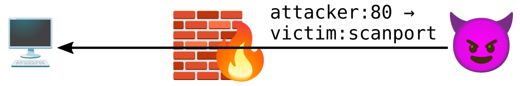
Packet filtering attacks

What can go wrong?

- Forged external addresses

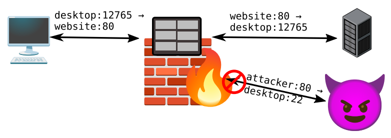


- Ephemeral port scans



Stateful packet filtering

Track internally-initiated connections:

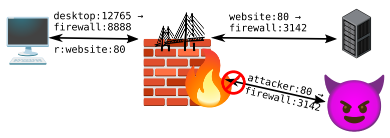


Need to maintain state for each connection (FIN, timeout, RST, active…)

State sync problems can be caused by, e.g., fragments

Proxy firewalls

Send a connection through firewall software (SOCKS, HTTP):

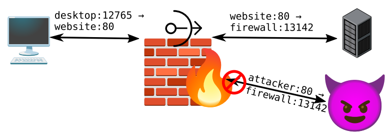

desktop:12765 → firewall:8888
r:website:80
website:80 → firewall:3142
attacker:80 → firewall:3142

Advantages: no state sync, can understand application level, authentication can be used.

Disadvantages: configuration effort, computation cost, insider attacks, …

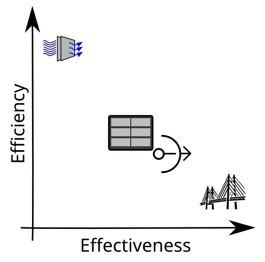
Network address translation

Firewall rewrites source address and port


desktop:12765 → website:80
website:80 → firewall:13142
attacker:80 → firewall:13142

Lower-cost and higher flexibility than proxies, potential state de-synchronization

Firewall tradeoffs



`iptables` examples (1/2)

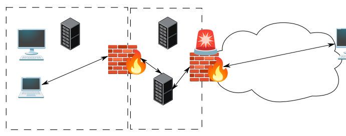
```
iptables -A INPUT -p tcp -m tcp ! --tcp-flags SYN,RST,ACK SYN \
         -j ACCEPT
iptables -P INPUT DROP

iptables -A INPUT -p TCP --destination-port 22 -j ACCEPT
iptables -I INPUT 1 -p udp --source-port 53 -j ACCEPT

# iptables -L -n
Chain INPUT (policy DROP)
   target    prot opt source     destination
   ACCEPT    udp  --  0.0.0.0/0  0.0.0.0/0  udp spt:53
   ACCEPT    tcp  --  0.0.0.0/0  0.0.0.0/0  tcp flags:!0x16/0x02
   ACCEPT    tcp  --  0.0.0.0/0  0.0.0.0/0  tcp dpt:22
```

`iptables` examples (2/2)

```
iptables -I OUTPUT -p tcp -d 192.168.0.0/24 --dport 80 -j DROP

# iptables -L -v # (L)ist rules, (v)erbose
Chain INPUT (policy DROP 129 packets, 20831 bytes)
 pkts bytes target  prot opt in  out source   destination
   25  2644 ACCEPT  udp  --  any any anywhere anywhere udp spt:domain
 523K  675M ACCEPT  tcp  --  any any anywhere anywhere tcp !SYN
    1    60 ACCEPT  tcp  --  any any anywhere anywhere tcp dpt:ssh

Chain OUTPUT (policy ACCEPT 372K packets, 25M bytes)
pkts bytes target  prot opt in  out source   destination
   5  300 DROP   tcp  --  any any anywhere 192.168.0.0/24 tcp dpt:http

iptables -P INPUT ACCEPT # reset policy
iptables -F INPUT # clears chain
iptables -F OUTPUT # clears chain
```

Network intrusion detection

NIDSes augment the "perimeter defense" approach to network security with a "burglar alarm"



Suspicious traffic triggers the alarm, prompting a response

NIDS characteristics

NIDSes can be classified/evaluated by:

Error rates	Against typical traffic?
Search type	Known intrusions or unknown behavior?
Type of sensors	Host and/or network?
Evasion	Failures against targeted attacks?

Error rates

A false positive error is when a non-intrusion raises an alarm:

- Squirrel chews through sensor cable
- Printer driver scans subnet for printer
- User mistypes password three times

A false negative is when an intrusion does not raise an alarm.

False Positive Rate (FPR) = #FPs / #Normal Events

False Negative Rate (FNR) = #FNs / #Intrusions

Base rate problems

Suppose the BCI network has 10M network flows/day, and 100 flows are attacks.

If BCNIDS has a 0.1% FPR, then:

- How many false alarms per day?
- What fraction of alarms are FPs?

Even with 0% FNR, what FPR is needed to equally balance FPs and TPs?

Base rate problems

Suppose the BCI network has 10M network flows/day, and 100 flows are attacks.

If BCNIDS has a 0.1% FPR, then:

- How many false alarms per day? 10K
- What fraction of alarms are FPs?

Even with 0% FNR, what FPR is needed to equally balance FPs and TPs?

Base rate problems

Suppose the BCI network has 10M network flows/day, and 100 flows are attacks.

If BCNIDS has a 0.1% FPR, then:

- How many false alarms per day? 10K
- What fraction of alarms are FPs? $>$99.9%

Even with 0% FNR, what FPR is needed to equally balance FPs and TPs?

Signature matching IDS

The misuse detection problem is to find behavior matching known intrusions. Basic strategy:

- Collect many examples of known attacks.
- Divide them into groups matching a signature.
- Match new flows against these signatures.

Example rule (Snort): `alert tcp any any -> myip 21 (content: "site exec"; content:"%"; msg:"site exec buffer overflow attempt";)`

Anomaly detection

Anomaly detection tries to identify "normal" traffic patterns.

Traffic that does not fit these patterns causes an alarm.

- Advantage: more robust to slight attack changes
- Disadvantage: people do crazy things on the Internet

IDS tradeoffs

	Signature	Anomalies
FPs:	low	high
New attacks:	missed	"sounds fishy"
Need to know:	existing attacks	normal traffic
	+ automated extraction	− delayed response
	− easy to evade	− mimic attacks
		− changes in normal

Network-based NIDSes

Monitoring for "network" attacks: DoS, protocol/application bugs, worms, viruses and software.

Example: port scanning. Signature is multiple connections to the same network in short time period.

Examples of "what can go wrong" include fragmentation, volume of network data, "low and slow" attacks, etc.

Example: Snort

Snort is a signature-based portable open-source NIDS with millions of downloads/installs (including at UMN OIT)

It scans packet logs, matching connections against sigs

Snort signatures are extended regular expressions that should match many variants of an attack, for example:

```
alert tcp any any -> [a.b.0.0/16,c.d.e.0/24] 80
(msg:"WEB-ATTACKS conf/httpd.conf attempt"; nocase; sid:1373;
flow:to_server,established; content:"conf/httpd.conf"; [...] )
```

Example: Zeek

Zeek (formerly “Bro”) is a “policy-based” NIDS that uses scripts to monitor connection protocol state.

Zeek logs “connection events” specific to the protocols for each connection, e.g. TCP handshake, SSH authentication, SSH records, SSH shutdown, TCP shutdown.

Scripts can alert when known attacks are detected or when unusual protocol states occur.