CS261 Computer Security **Fall 2007**

Lecture 14: October 16, 2007

Lecturer: David Wagner *Scribes: Steven Houston*

14.1 Introduction and Packet-Filtering Examples

Firewalls are the network community's response to application-level software vulnerabilities. They typically serve as a reference monitor, filtering traffic sent between an untrusted and trusted network. Since routers are the nodes that forward packets from one network to another, firewalls are often located on routers. Our example of a router acting as a firewall between the malicious outside Internet and the internal UC Berkeley network is shown in Figure 14.1 below. [world] and [ucb] represent the two network interfaces of the router.

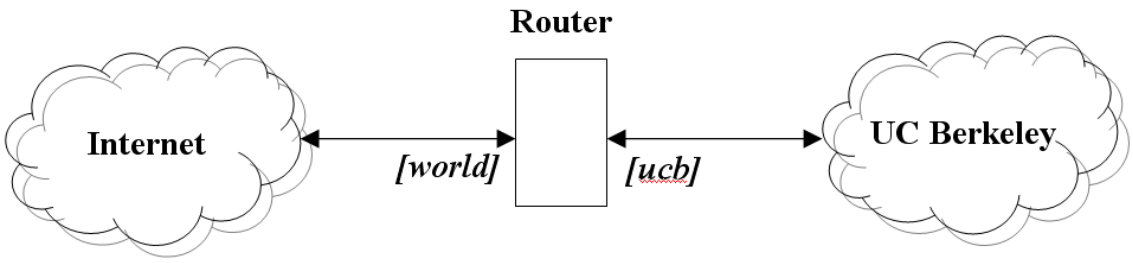

Figure 14.1: Our example firewall monitoring packets between the Internet and UC Berkeley.

Suppose a UC Berkeley system administrator notices an unusual amount of packets being sent to a local finger daemon. He might use the following rule to filter them at the router:

```
drop *:* --> *:79
```

However, this doesn't let Berkeley use finger at all since outgoing finger requests (to external finger services) are also dropped. To drop only incoming finger requests, he might try the following packet-filtering rule:

```
drop *:*/world --> *:79/ucb
```

Although this technique may work for finger, it has problems for services that do not use the well-known ports (0-1023) by default. For example, the NFS server listens on port 2049 by default, and suppose we want to block incoming NFS connections. The packet-filtering rule

```
drop *:*/world --> *:2049/ucb
```

would drop the desired packets but would also result in an unstable network. Client processes typically choose a random source port in the range 1024 to 65534. Suppose a UC Berkeley client initiates a connection to an outside SSH server, and the client chooses port 2049 as the source port (this occurs 1 in 64511 times if the port is chosen randomly). Then, all responses from the outside SSH server to the UC Berkeley client will dropped be by the filtering rule.

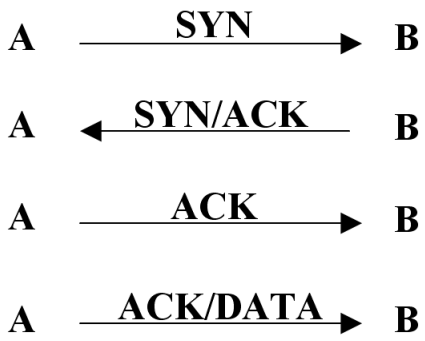

Figure 14.2: Diagram of the TCP handshaking protocol.

If the NFS connection uses TCP (see Figure 14.2), we can alleviate the problem by specifically dropping only TCP packets with the SYN bit set, since the SYN bit is only set during the TCP handshake:

```
drop *:*/world --> *:2049/ucb TCP (SYN)
```

Note that all of the above filters are blacklists (they have the implicit rule `allow default`). Can we write a secure whitelist? The following should allow internal machines to initiate any TCP connection and external machines to initiate only TCP connections to deliver mail (SMTP uses port 25 by default):

```
allow *:*/world --> *:*/ucb TCP (ACK)
```

```
allow *:*/ucb --> *:*/world
allow *:*/world --> *:25/ucb TCP
deny default
```

What if we later decide to add a publicly accessible webserver with IP address 1.2.3.4? Simply add the following line to our filter:

```
allow *:*/world --> 1.2.3.4:80/ucb TCP
```

We rely on the inside host's network stacks (e.g. TCP stacks in the OS) to drop packets that don't correspond to an existing connection, but we don't make any assumptions about the application's behavior. Remember, firewalls aim to prevent exploits of buggy applications.

14.2 Security Analysis of Firewalls

The firewall can be thought of as a reference monitor, checking all traffic between an external and internal network. How do we evaluate its security? Recall the three standard rules for evaluating reference monitors:

1. The reference monitor must always be invoked, i.e. complete mediation.
2. The reference monitor must be tamper-resistant.
3. The reference monitor must be verifiable.

Suppose we have a trust model of trusting insiders (internal users of our network) and distrusting outsiders. By trusting insiders, we mean we trust them not to purposefully do anything malicious, but they still may create a vulnerability in our network or bypass our firewall inadvertently. Let's analyze how well a firewall enforces the above rules with this trust model in mind:

1. The first rule can be violated by the following:
   (a) Insiders connecting to the external network via another gateway. For example, an insider could connect to the Internet with a modem and the insider's ISP, and consequently all traffic would bypass the firewall.
   (b) Tunneling a blocked protocol underneath a permitted protocol. For example, an attacker can encapsulate SNMP packets within the payloads of HTTP packets.

(c) Code being physically transferred into the network via mobile devices such as laptops, USB drives, etc. For example, an insider's laptop gets infected with a worm while he's at a coffee shop, and he later connects the laptop to the internal network. As a result, the worm bypasses the firewall.

(d) Connecting via a wireless access point. A user connecting wirelessly typically bypasses the main firewall monitoring wired traffic, but his access may be limited to a demilitarized zone.

2. The second rule usually holds since there are fewer routers than internal hosts, they are under constant system administrator control, and unneeded services can be turned off to minimize the attack surface.

3. The third rule is usually violated as firewalls aren't typically verifiable in practice.

Packet-filtering firewalls, in particular (as opposed to application-level firewalls), have the following weaknesses as well:

1. stateless
2. course granularity
3. some protocols are problematic: UDP, FTP
4. may allow portscanning (an attacker can send a packet with the ACK bit and a few other choice bits set and the endpoint will essentially tell the attacker whether or not an application is listening on that port)
5. cannot enforce application-level policies
6. cannot filter based on content

14.3 Conclusion

Additional thoughts on firewalls:

1. trust of inside hosts makes firewalls fragile (webbrowser with mobile code)
2. packet filters are very successful (since its security is orthogonal to what it's protecting, it's very easy to deploy and potentially make more secure by adding additional firewalls)
3. most packet filters today are mildly stateful (e.g. is a TCP connection already established or still in a handshake?)