Firewalls and Security Hardening

Zack Orndorff

What is “security hardening”?

Reducing attack surface!

Wikipedia lists the following examples:

- Keeping security patches updated
- Installing firewall
- Closing certain ports
- Not allowing file sharing among programs
- Installing virus and spyware protection
- Using containers or virtual machines
- Creating strong passwords
- Keeping a backup
- Disabling cookies
- Using encryption when possible
- Disabling weak encryption

Principle of least privilege

Each entity should have the fewest privileges required to accomplish its task

Can be applied to users: Your Helpdesk does not need Domain Admin

Can be applied to processes: Your web server does not need to be run as root.

Can be applied to network access: outside users do not need to be able to SSH to your server

We use firewalls to accomplish this last objective

Principle of least privilege: *Examples*

For users:

- Your OS has more granular permissions than user/admin. Use them.
- I don’t need to see what processes other users run
- Set permissions on folders to make use of all built-in protection. Don’t chmod -R 777 /var/www

For programs:

- Run them as their own user. Not root. Not you.
- In general, they don’t have to be able to overwrite themselves.
- If the program can be split into pieces, some with less privileges than the other, do it (see: SSH, Chrome)
- Don’t listen on all interfaces if not necessary!

There will be occasion to practice this in the lab!

Firewalls

Turn the heat up on the attacker

Firewalls look at traffic and alter or remove it, based on some criteria.

Types of firewall:

- Packet inspection
- Stateful
- Proxies
- “Next-gen” (think deep packet inspection & flashy dashboards)

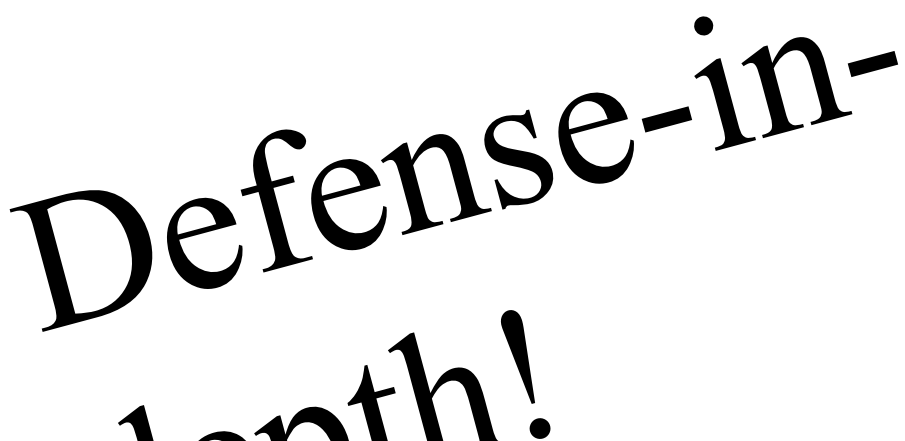
Each of these can be host-based or network-based

But why do I need a firewall?

I just configured all my services to run as random users!

They’re only listening on localhost, except for the publicly-accessible services!

What does the firewall add?



Zack, we’re tired of theory!

Can we get a little bit more concrete?

Enter: iptables

Interface to the Linux kernel netfilter subsystem

Support for all kinds of packet mangling

iptables is the tool you use to interact with it

We'll be using it as a host-based stateful firewall

Things you might hear of:

- ipchains - old predecessor to iptables
- nftables - new successor to iptables
- pf - BSD's packet filtering subsystem
- Windows Firewall - well, it's in the name

How to use iptables, part 1

Gets its name from the tables that it runs traffic thru

Within each table, there are chains

We'll care about the default (filter) table, and the INPUT and OUTPUT chains

For future reference, the nat table is useful for NAT, the mangle table is useful for … well … mangling packets. The FORWARD chain in the filter table is for systems that act as gateways.

So, for simple uses, you pick the thing you want to filter, say INPUT traffic, and add rules to the chain.

How to use iptables, part 2

- iptables -A INPUT -i lo -j ACCEPT
  - Allows incoming traffic from the loopback interface. This prevents breaking programs that expect working loopback.
- iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
  - Allows existing connections to continue. This prevents blocking responses to outgoing connections, among other stuff.
- iptables -A INPUT -m state --state NEW -p tcp --dport 80 -j ACCEPT
  - Allows NEW connections to TCP port 80. You'll probably find this on a web server
- iptables -A INPUT -m state --state NEW -s 130.85.0.0/16 -p tcp --dport 22 -j ACCEPT
  - Allows NEW connections to TCP port 22 (SSH) from UMBC's ip range
- iptables -I INPUT -s 130.85.300.4 -p tcp --dport 25 -j DROP
  - Some jerk was spamming me from this (not actually valid) IP, so drop all their SMTP traffic
- iptables -P INPUT DROP
  - Sets the policy for the INPUT chain to DROP all packets not explicitly ACCEPTed
- iptables -P FORWARD DROP
  - Ditto for FORWARD

For goodness sake, don't memorize the syntax. Memorize the ideas, syntax can be looked up.

How to craft firewall rules

1. Evaluate what network traffic is required for proper use of the system
2. Block everything else

Beware of traffic that is not immediately obvious, say a database server on another computer or something.

There is a rigidity tradeoff between security and ease of adding new services.

Lab

Download rekt2.ova from the link posted in Slack

“Scored” services:

- Web app on port 80
- DNS
- SSH
- A system at IP address subnet.56 needs to use the Redis instance running on port 6379

Craft firewall rules with iptables and apply them

Turn off unnecessary services (note some are necessary but not mentioned above)

When you think you’re done, ask someone how you did

There are at least 24 things wrong with this VM. Keep notes :)