Defending Computer Networks
Lecture 10: Firewalls

Stuart Staniford

Adjunct Professor of Computer Science

Logistics

- HW2 out later today

Latest News

SECURITY

Patch Bash NOW: 'Shell Shock' bug blasts Linux, OS X systems wide open

CGI scripts to DHCP clients hit by Heartbleed-grade remote-code exec vuln

By John Leyden, 24 Sep 2014 Follow 2,932 followers

110

RELATED STORIES

OpenSSL promises devs advance notice of future bugs, slaps if they blab

Oh SNAP! Old-school '80s Unix hack to smack OSX, iOS, Red Hat?

Intelligent flash storage arrays

Updated A bug discovered in the widely used Bash command interpreter poses a critical security risk to Unix and Linux systems – and, thanks to their ubiquity, the wider internet.

It lands countless websites, servers, PCs, OS X Macs, various home routers, and more, in danger of hijacking by hackers.

The vulnerability is present in Bash up to and including version 4.3, and was discovered by Stephane Chazelas. It puts Apache web servers, in particular, at risk of compromise: CGI scripts that use

http://www.theregister.co.uk/2014/09/24/bash_shell_vuln/

Bash supports exporting not just shell variables, but also shell functions to other bash instances, via the process environment to (indirect) child processes. Current bash versions use an environment variable named by the function name, and a function definition starting with "() {" in the variable value to propagate function definitions through the environment. The vulnerability occurs because bash does not stop after processing the function definition; it continues to parse and execute shell commands following the function definition. For example, an environment variable setting of

```
VAR=() { ignored; }; /bin/id
```

So far, HTTP requests to CGI scripts have been identified as the major attack vector.

A typical HTTP request looks like this:

```
GET /path?query-param-name=query-param-value HTTP/1.1
Host: www.example.com
Custom: custom-header-value
```

The CGI specification maps all parts to environment variables. With Apache httpd, the magic string "() {" can appear in these places:

* Host ("www.example.com", as REMOTE_HOST)
* Header value ("custom-header-value", as HTTP_CUSTOM in this example)
* Server protocol ("HTTP/1.1", as SERVER_PROTOCOL)

http://seclists.org/oss-sec/2014/q3/650

Main Goals for Today

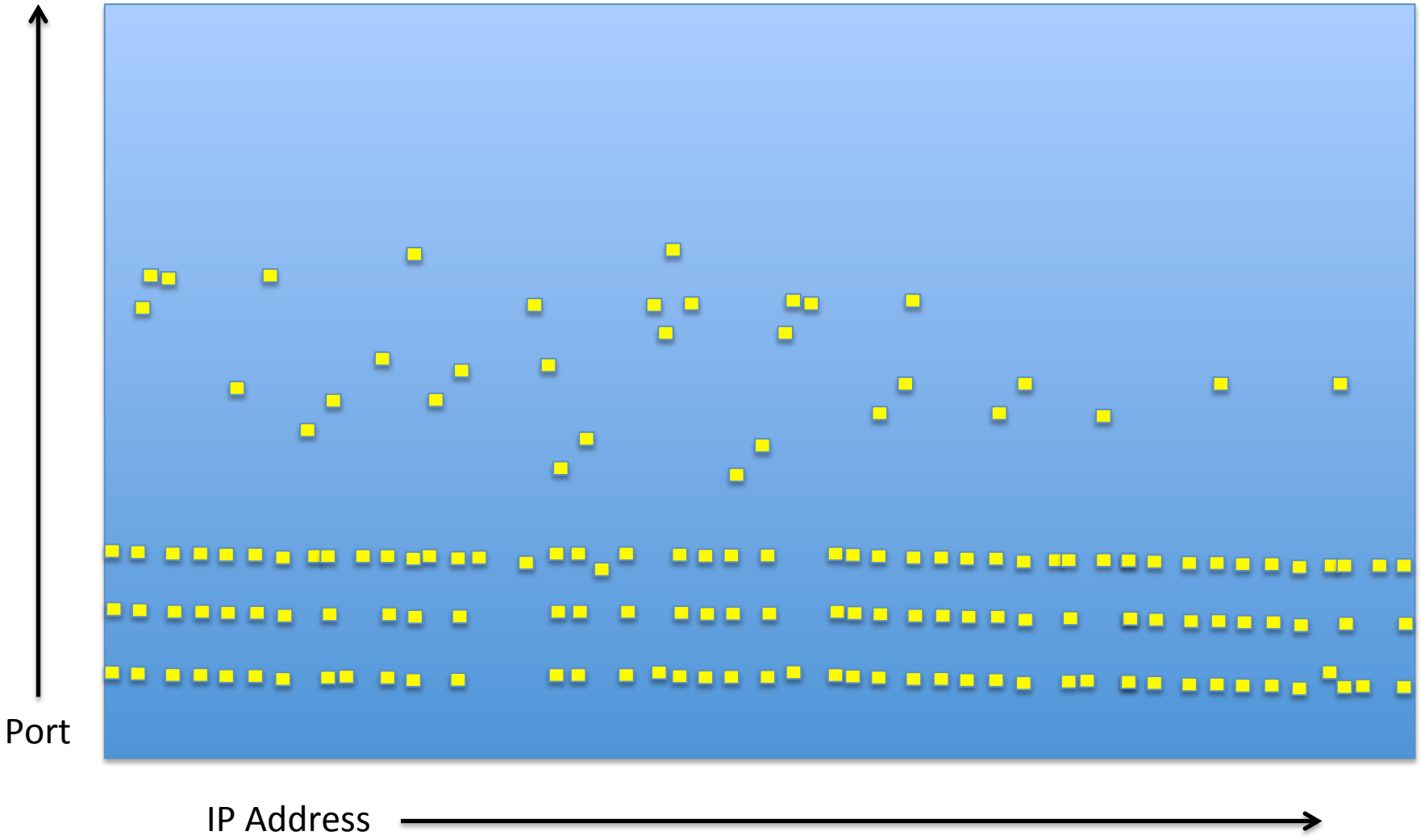
- Firewalls.
- Maybe start Distributed Denial of Service.

Where We Are in Syllabus

Rough Lecture Syllabus:

✔ 1. The technical nature of software vulnerabilities and techniques used for exploiting them.
✔ 2. The pressures of commercial software development, and why firms very rarely produce secure software, even though they should.
✔ 3. Basics of monitoring a network, intro/refresher on TCP/IP. Switches, wireless access devices, routers.
✔ 4. Network reconnaissance techniques – ping sweeps, port scans, etc.
✔ 5. Algorithms for detecting port scans on the network.
☞ 6. Firewalls and network segmentation as a defense against inbound attacks.
7. Detecting exploits with string matching approaches (Snort and similar).
8. Network layer approaches to evading detection.
☞ 9. Large scale attacks – worms and distributed denial of service.
10. HTTP attacks as a way around the firewall. Drive-by downloads and social engineering.
11. Defending against HTTP attacks. Web-proxies, in-browser defenses, anti-virus systems.
12. SMTP attacks – spear-phishing, and defenses against it.
13. HTTPS: Encryption and virtual private networks as a means to maintain confidentiality.
14. The modern enterprise network: what a large-scale network looks like, and emerging trends affecting it (BYOD, cloud).
15. Legal and ethical issues in defending networks.

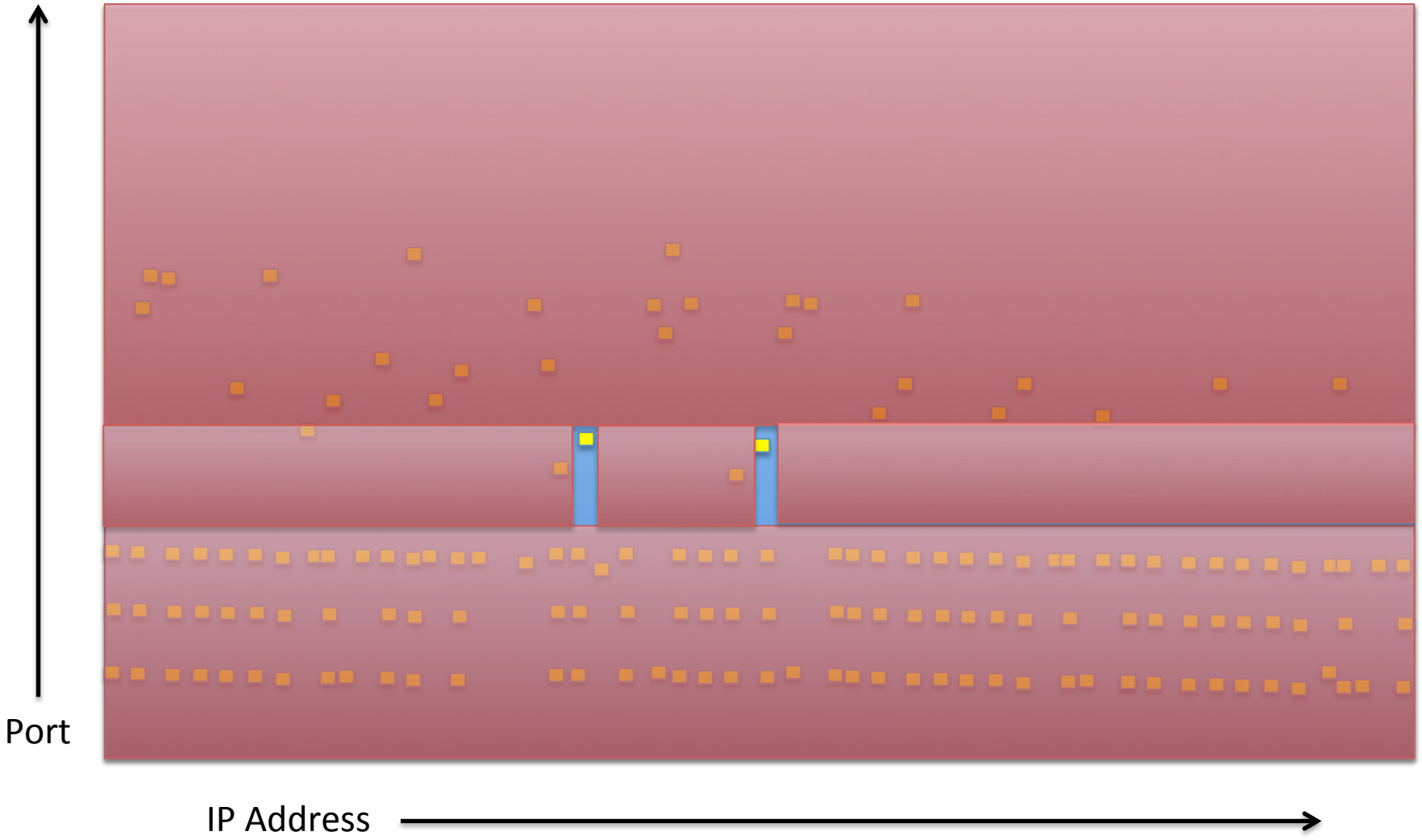
Open Network From the Internet



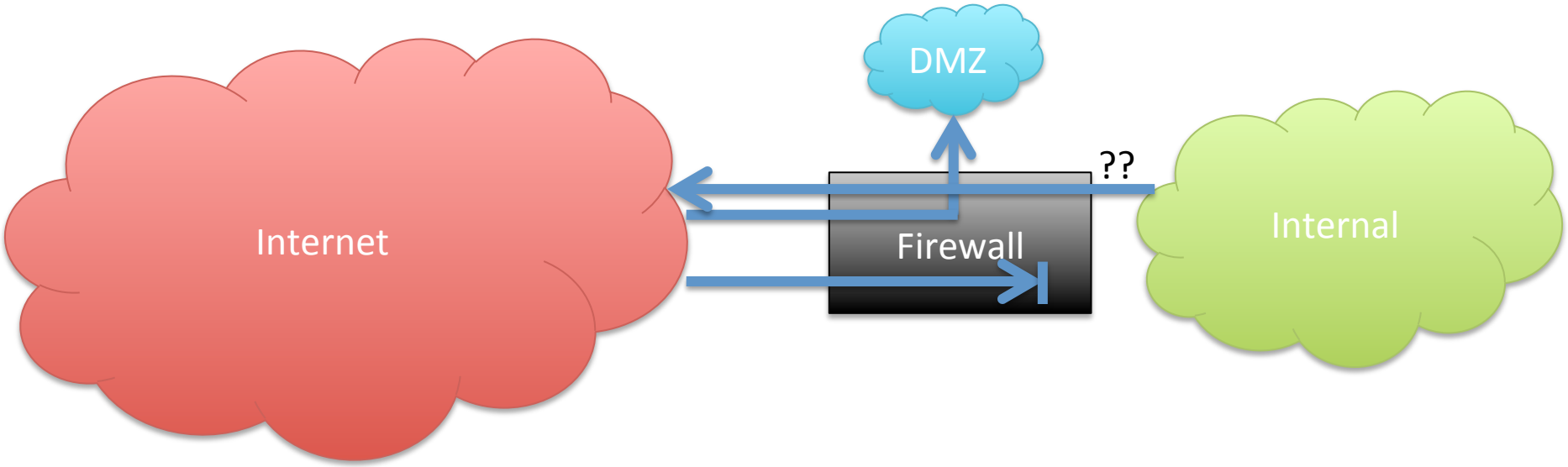
Scale of the Problem

- Big network might have $O(10^5)$-(10^8) machines
- Most will have some open ports
- Many, many versions of many, many codebases.
- Many different departments with differing needs/politics.
- Extremely hard to keep everything patched/ configured correctly
- But trivial to scan/exploit from the internet.

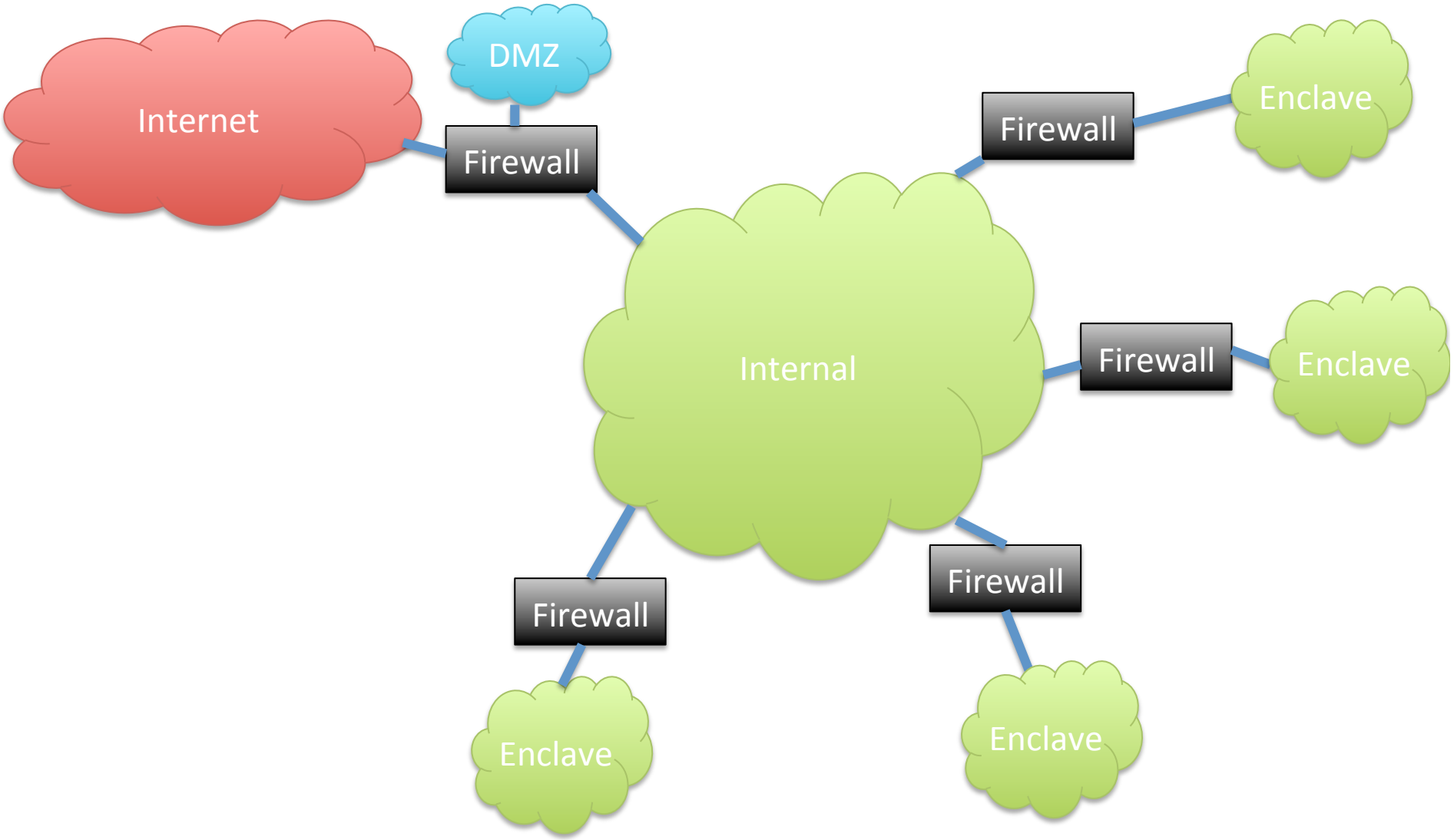
Establish Central Control



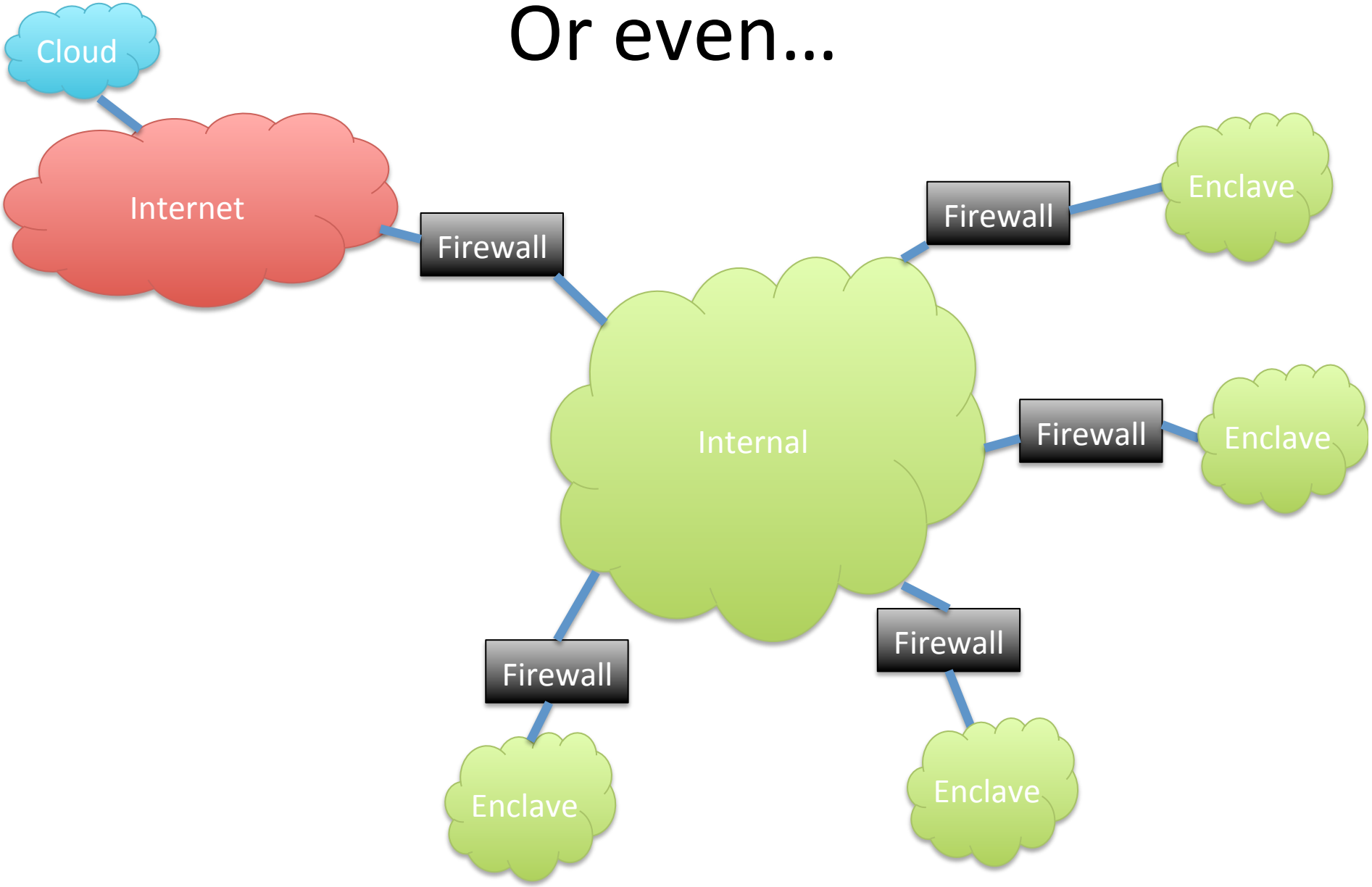
Better Yet



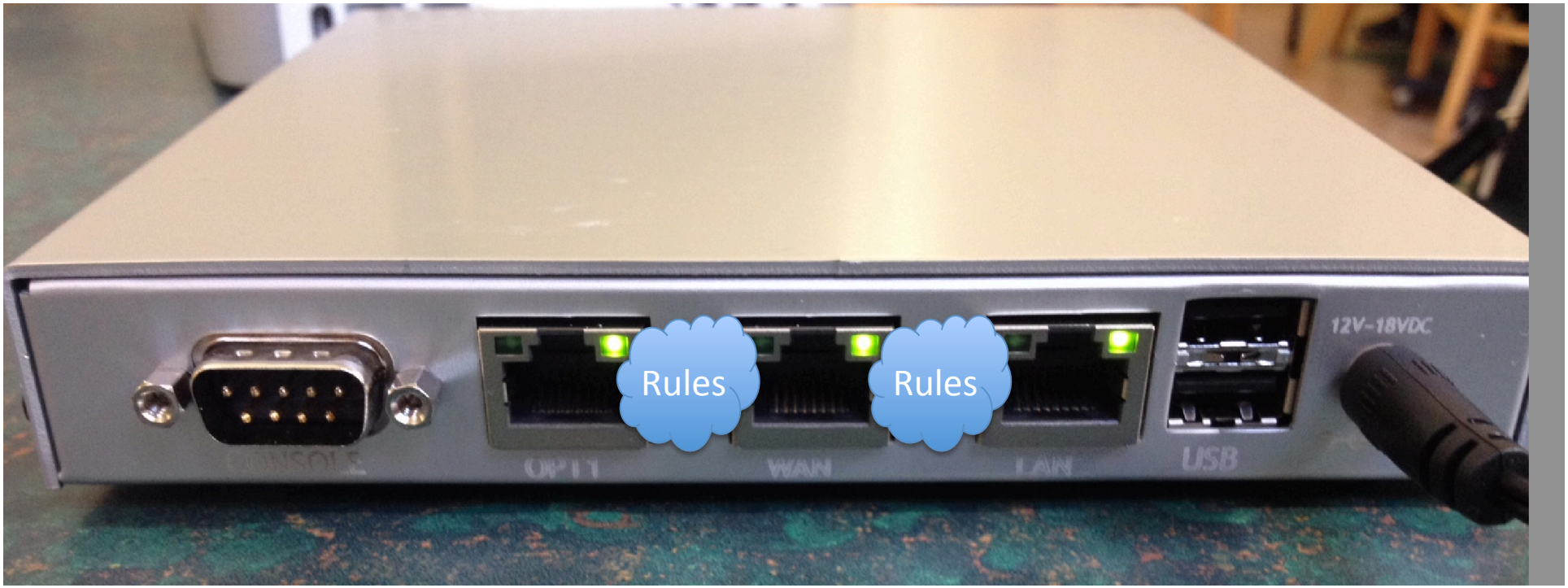
Or…


Internet
DMZ
Firewall
Firewall
Enclave
Internal
Firewall
Enclave
Firewall
Firewall
Enclave
Enclave

Or even…


Cloud
Internet
Firewall
Internal
Firewall
Enclave
Firewall
Enclave
Firewall
Enclave
Firewall
Enclave

Firewall Basic Concept



(This is Netgate M1N1Wall – low-cost, low-power open source firewall using FreeBSD/pfSense. Runs on AMD Geode cpu.)

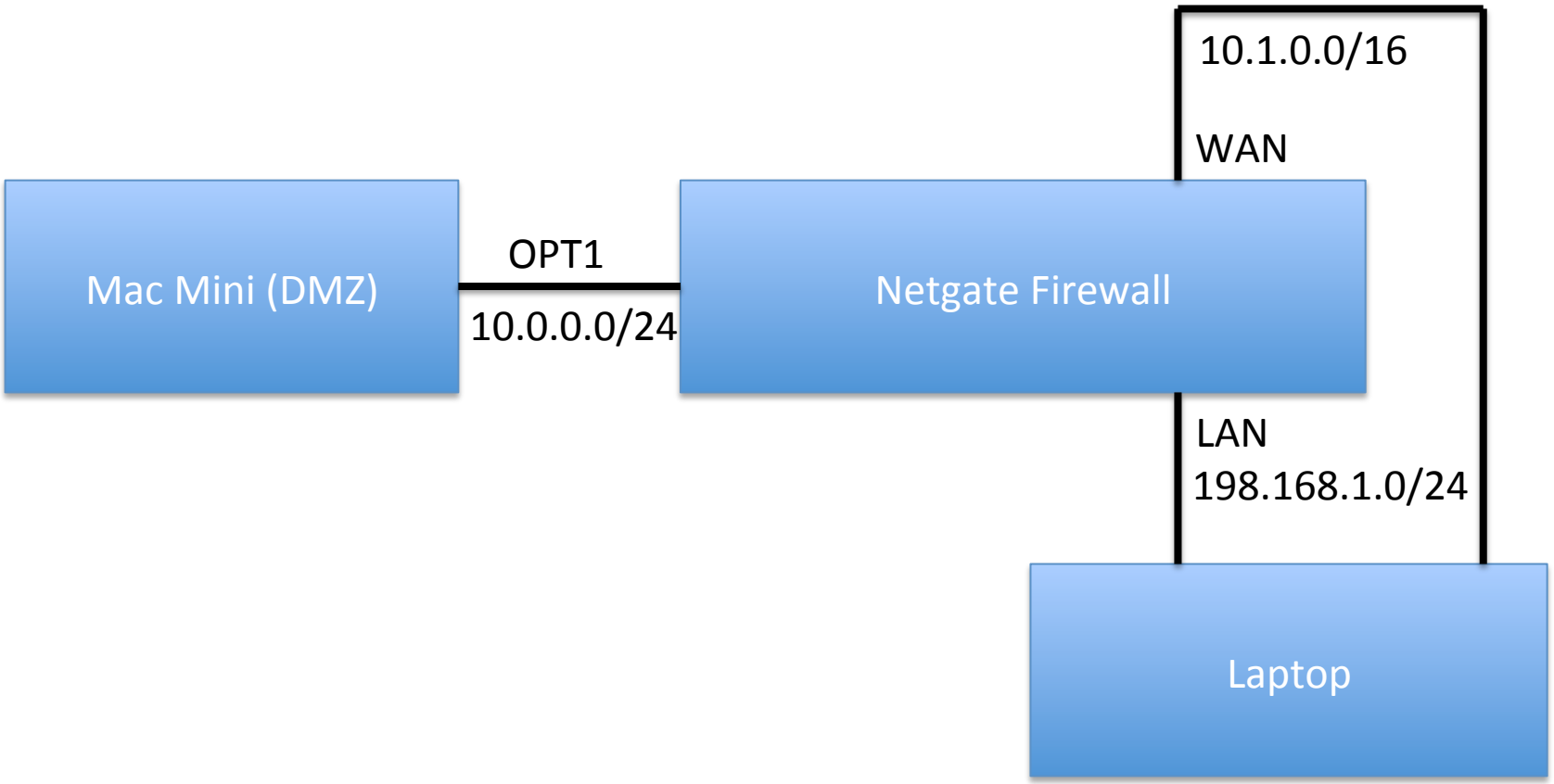
Typical Firewall Rule

- Block in on LAN from 192.168.1.0/24 port any to 0.0.0.0/0 port 53
  - Any packets coming from LAN to port 53 will be dropped.
    - Effect of rule in isolation
    - Could be part of strategy to force clients to use only officially sanctioned DNS servers

Firewall Rulesets

- Typically a significant number of rules, that together enforce the policy.
- Some firewalls take “last match” as dispositive, others take “first match”.
- Generally want first/last to be “block all” to ensure only permitted traffic is allowed.
- Stateful firewalls apply rules only to first packet of connection,
  - then will allow rest of connection to proceed
  - Performance benefit: looking up in flow table much faster than applying all of rules to packet.

Firewall Demo Wiring Diagram



Tour of a Firewall GUI

- Dashboard
  - Let's check basic setup
    - Check IP addresses on laptop match
    - Dashboard
    - Routes correct
    - Make sure we can ping Mac Mini from firewall
    - Check arp table
    - Make sure we can ping Mac Mini from LAN network.
    - Have a quick look at state table

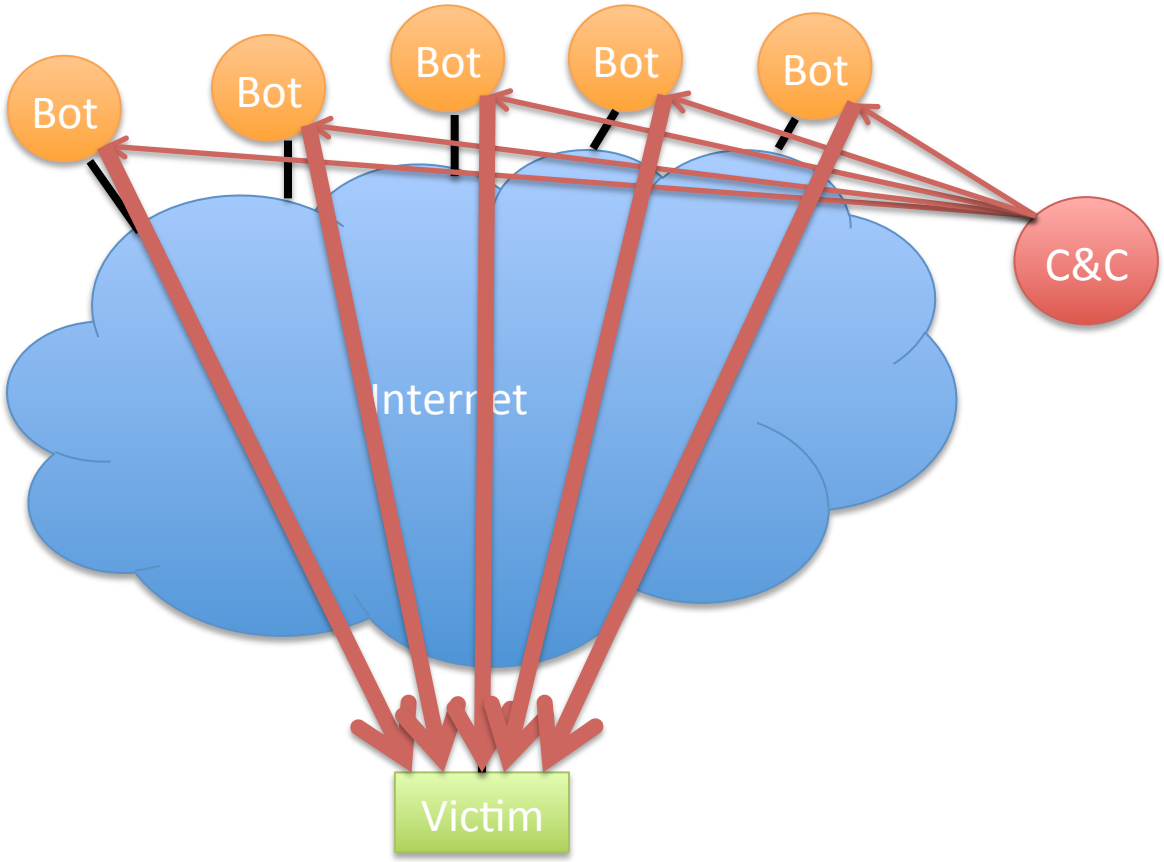
Firewall Rules

- Inspect the Rules
- Nmap through the firewall from WAN
  - Unplug LAN wire
  - sudo nmap -Pn -n -sS –T5 10.0.0.2
  - Replug LAN wire
- Change a rule
- Nmap through the firewall and see we can no longer see ports
- Inspect the state table in the fw
- Add a rule to reject (reset) connections
  - See how the nmap result changes

DDOS – Distributed Denial-Of-Service

- Main goal
  - take out an Internet site (“denial of service”)
  - By flooding with bad traffic
  - From many source (“distributed”)
- Could also be used on internal network,
  - Not seen much so far, if at all.
  - Obvious cyber-war/cyber-terrorism tactic

Basic Setup of a DDOS Botnet



Illustrative only: practical attacks will have many more bots

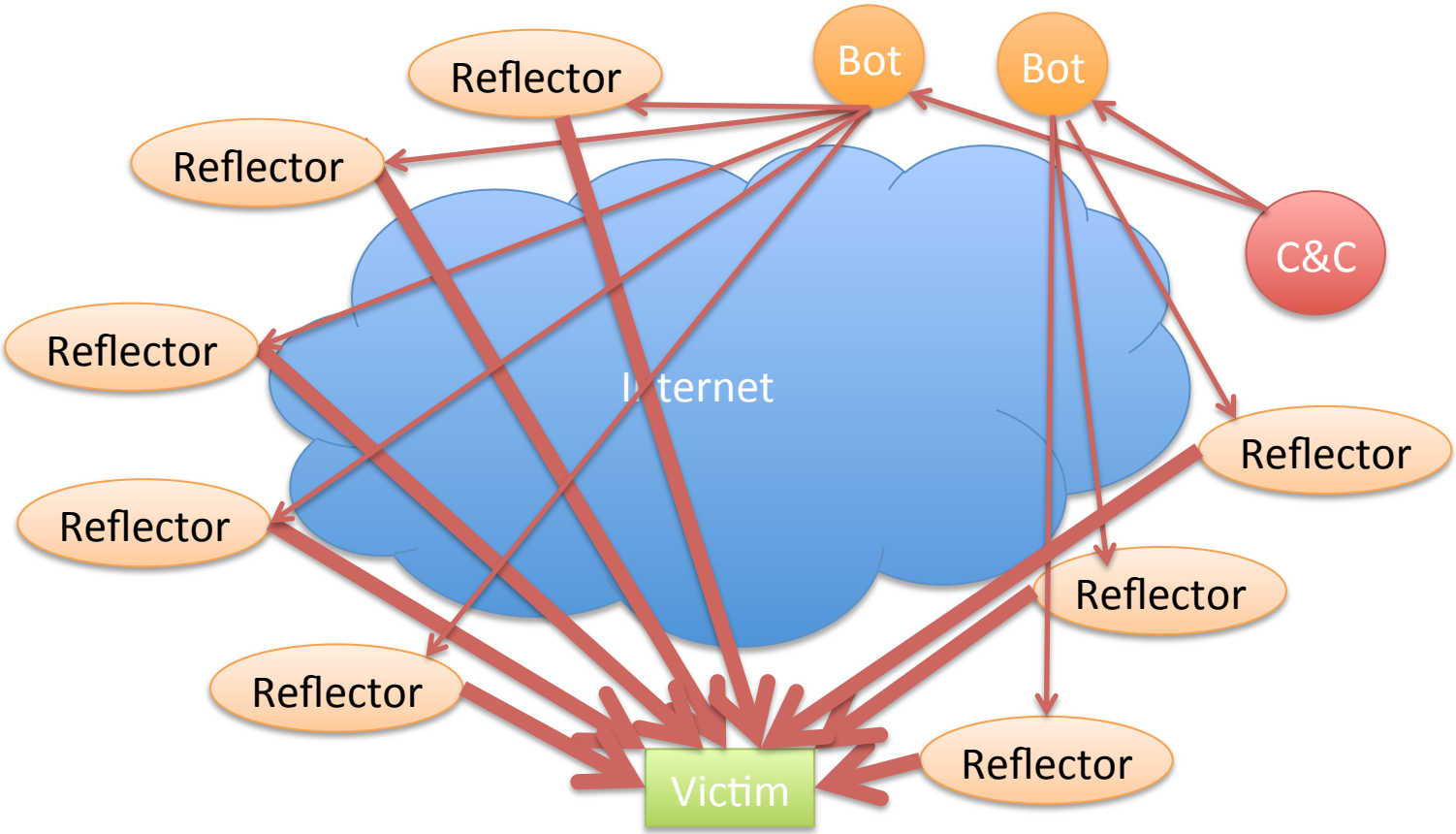
What Packets Should We Send?

- Ping flood
  - ICMP echo request
- Syn flood
  - Exploit limitations in handling of half-open connections in older stacks
- Genuine looking requests
  - The more genuine and randomized, the harder to block
- Application layer exploits
  - ASLR etc will prevent exploitation, but not crash

Reflectors

- A Reflector is anything that
  - If you send it a packet, will respond with pkts
  - Preferably lots of big packets
  - Then send it a packet with src spoofed as the victim
  - Get it to send lots of packets back to the victim
  - Can amplify a DDOS greatly
  - Also makes it harder to trace

Reflection Attacks



Illustrative only: practical attacks will have many more bots/reflectors

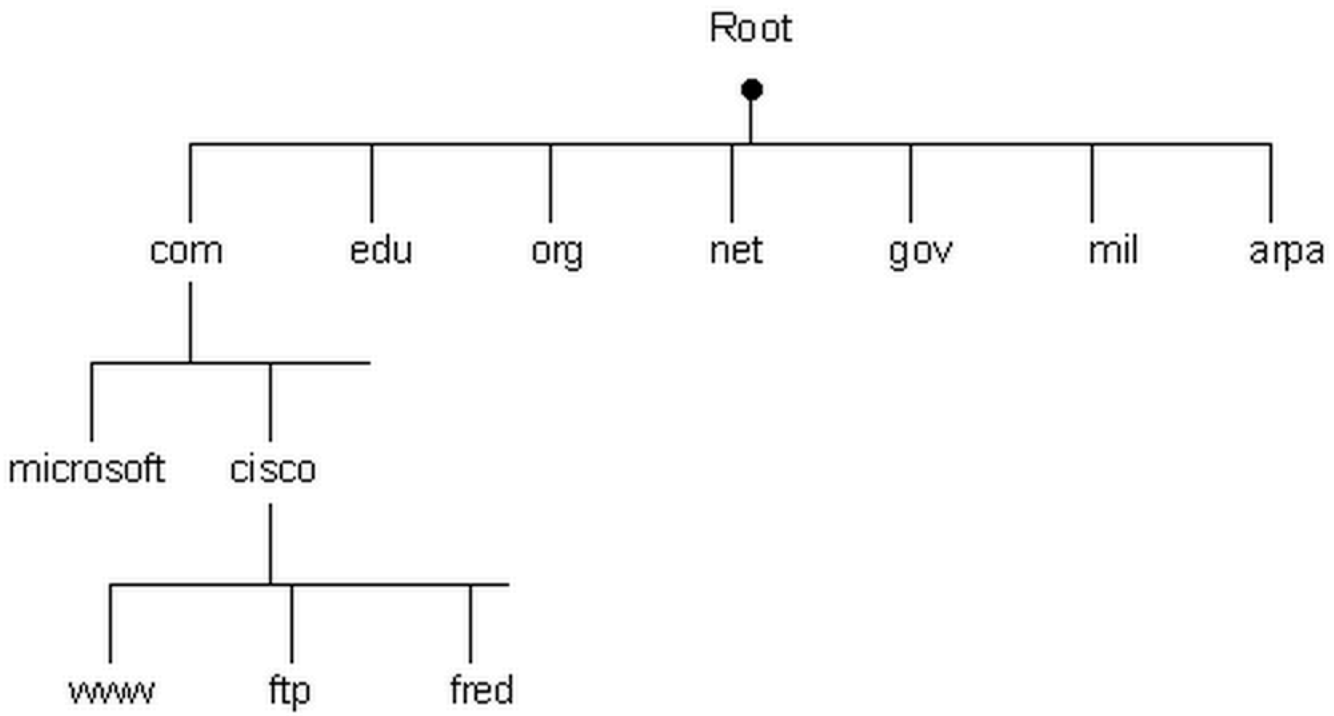
What Will Work as a Reflector?

- Any TCP host (send SA or R in response to S)
- ICMP (eg echo response to echo request)
- DNS – especially with recursion
  - Issue on campus recently
  - Let's look at this in more detail

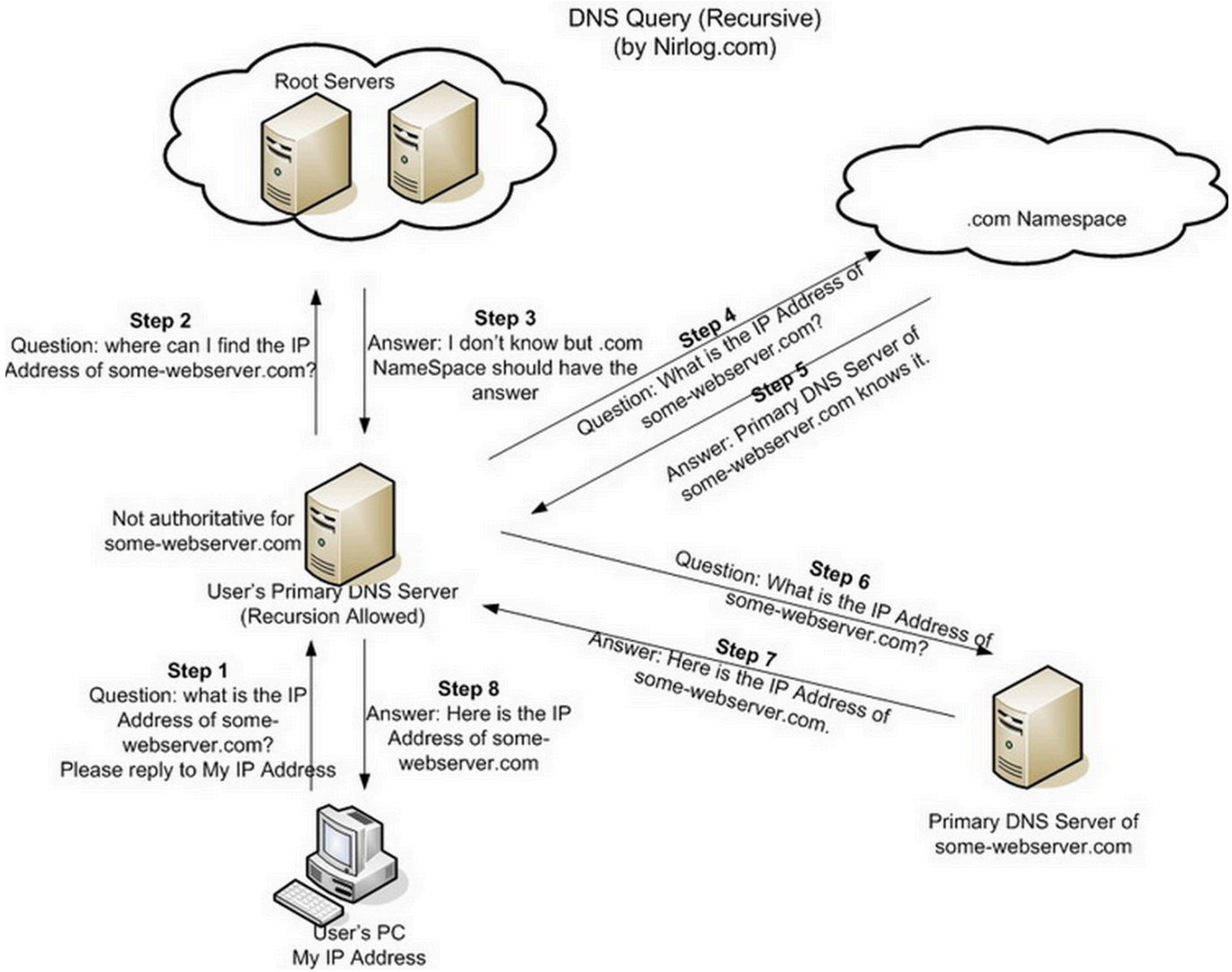
Domain Name Service

- Global Internet service to map names to IP addresses.
- Part of core TCP/IP suite of protocols
  - RFC 882 (1983) updated by RFC 1034 (1987)
  - Replaced manually maintained “hosts.txt” of all Internet connected computer’s IP addresses.
- Let’s do it
  - unplug from fw demo
  - dig www.nytimes.com

The DNS Hierarchical Name Tree



How a DNS Query Works



Credit: http://securitytnt.com/dns-amplification-attack/