CIS 4930: Secure IoT

Prof. Kaushal Kafle

Lecture 16

Class Notes

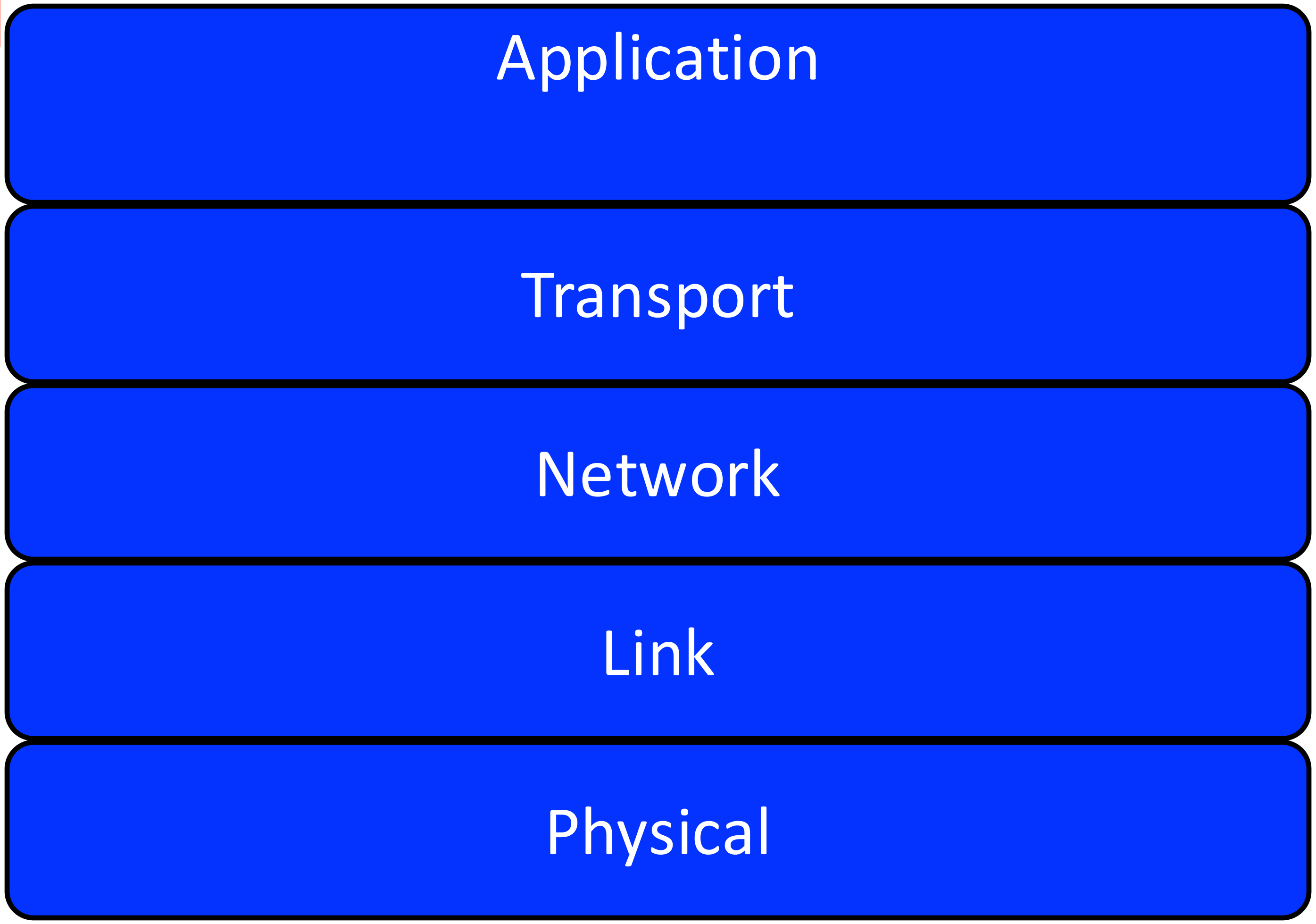
- Hw4 available from tonight.
  - Covers the network security section (up to 11/21)
- **Project phase 3: Security analysis of IoT apps (Android)**
  - Select 3 apps per person to analyze.
  - https://github.com/Secure-Platforms-Lab-W-M/IoTSpotter/tree/main?tab=readme-ov-file#1-37k-mobile-iot-apps
  - This link contains the 37k mobile IoT apps identified in the previous study. You can use this as reference.
  - OR you can find the 3 IoT apps by simply searching in the Google Play store.

Class Notes

- **Project phase 3: Security analysis of IoT apps**
  - Write a project proposal that includes:
    - The apps that your group selected, and who is analyzing which apps.
    - Why each of those apps are of interest to you
    - The plan for analysis
      - Tool used to decompile and get the source code (I recommend jadx)
        - **You cannot use apps that have obfuscated source code**
      - Tools you want to use for analysis of crypto-API misuses
        - Okay to do manual analysis without using tools
      - Your analysis can include all crypto-API misuses we discussed in class (has to include **at least 3 things**)

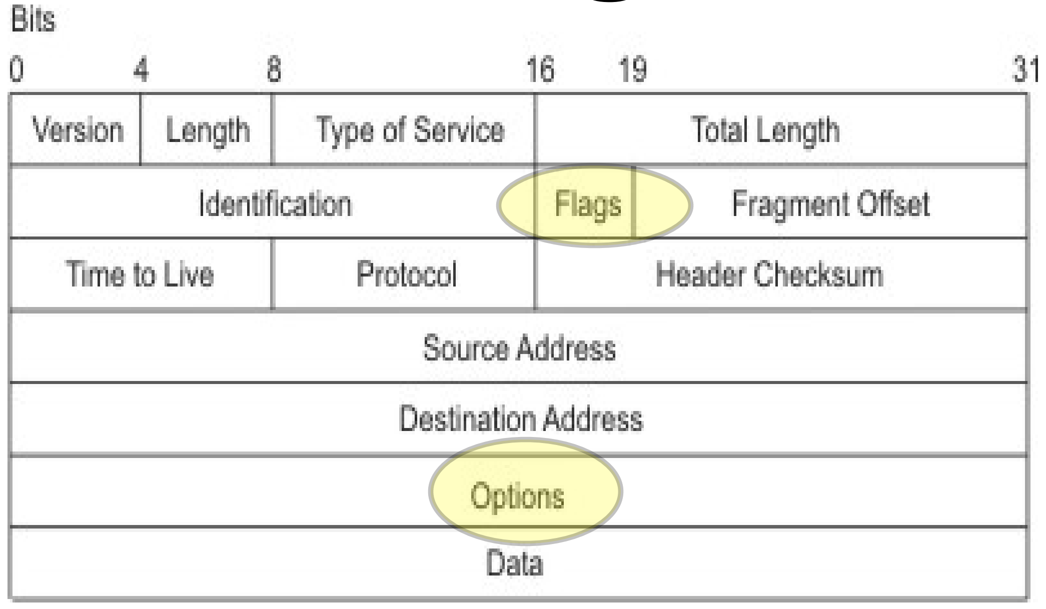
TCP/IP security

Network Stack, yet again



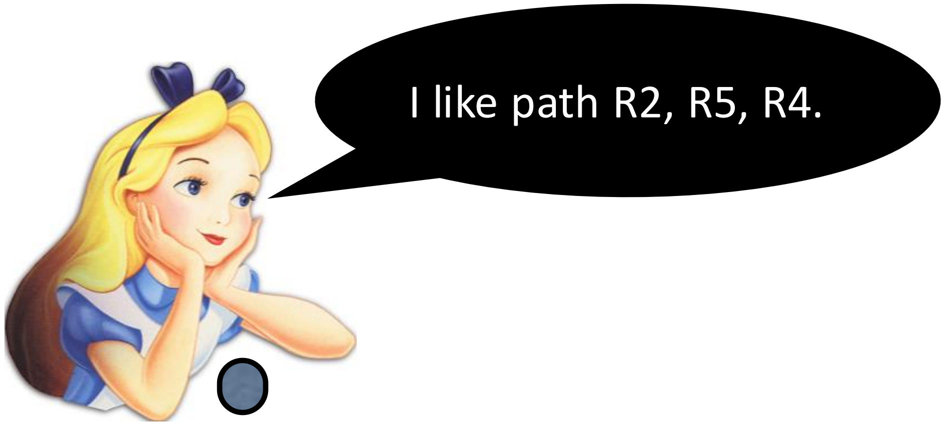
IP Source Routing

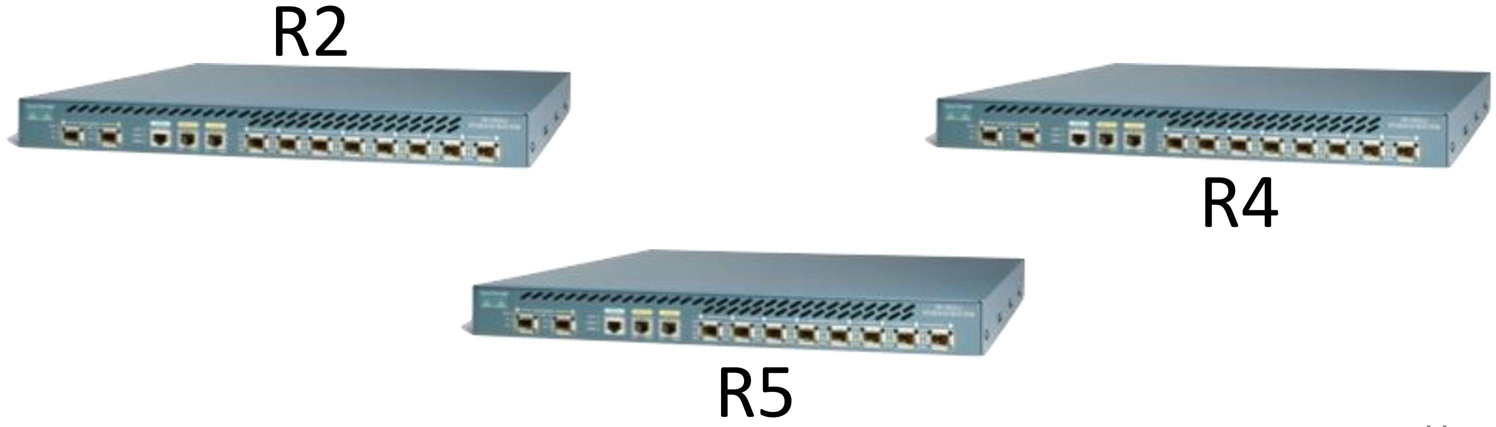
- Standard IP Packet Format (RFC791)
- Source Routing **allows sender to specify route**
  - Set flag in *Flags* field
  - Specify routes in *Options* field



- Each router examines the packet and sends it to the next router.

Source Routing





Source Routing

- Q: What are the security implications of Source Routing?
  - Access control?
  - DoS?
  - MitM?

Bypass security measures by spoofing source IP (done via modifying packets)

Attacker uses victim's IP as source to communicate with server(s). Server responses are then reflected to the victim.

Force packets to travel to a specific router, drop or modify packets.

Q: What are the possible defenses?

A: Block packets with source-routing flag

Routing Manipulation

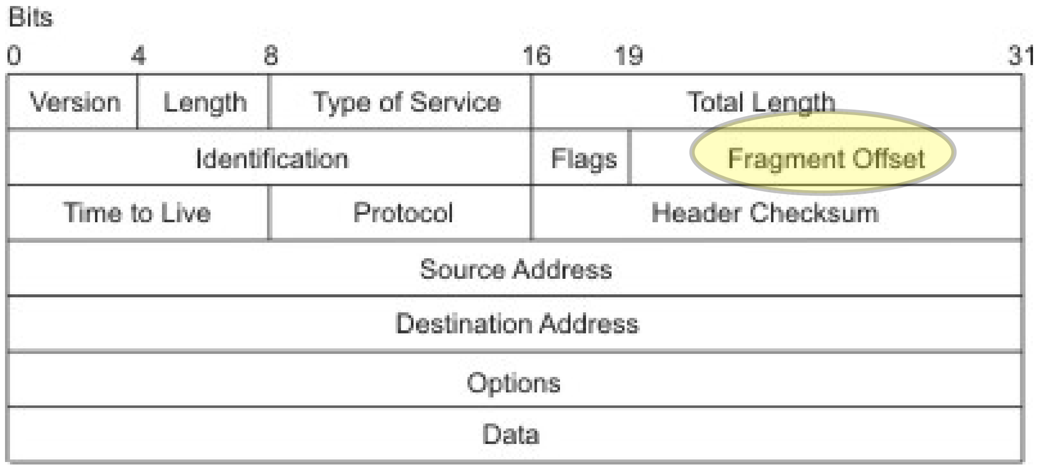
- RIP - Routing Information Protocol
  - **Distance vector routing protocol** used for the local network
    - Distance – hop count, Vector – Direction (or next hop)
  - Routers exchange reachability and "distance" vectors for all the sub-networks within (a typically small) domain
  - Use vectors to decide which route is best
- **Problem:** Data (vectors) are not authenticated
  - Forge vectors to cause traffic to be routed through adversary
  - or cause DoS
- Solutions: ?
    - Authentication – RIP v2 supports MD5 authentication (router configured with

Internet Control Message Protocol (ICMP)

- ICMP is used as a control plane for IP messages (i.e., for diagnostic and error reporting)
  - Ping (connectivity probe)
  - Destination unreachable (error notification)
  - Time-to-live exceeded (error notification)
- **ICMP messages are easy to spoof: no handshake**
- Some ICMP messages cause clients to alter behavior
  - e.g., manipulate RST flags on destination unreachable or exceed TTL limits
    - Enables attacker to <u>remotely</u> reset others' connections
- Solution:
  - Verify/sanity check sources and content
  - Filter most of ICMP

Ping-of-Death: Background: IP Fragmentation

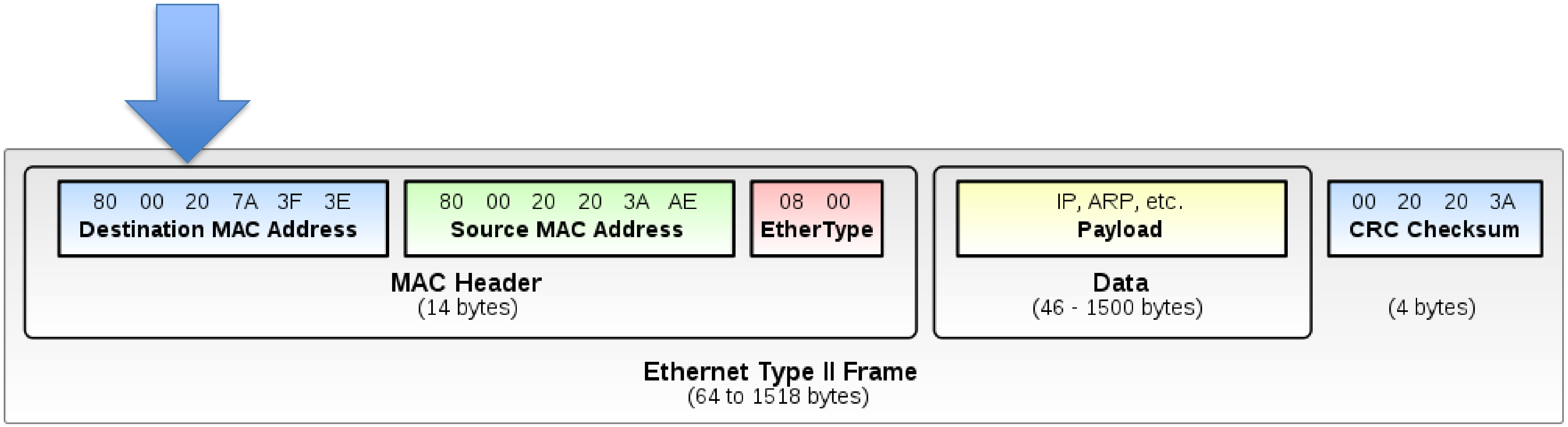
- 16-bit "Total Length" field allows $2^{16}-1$=65,535 byte packets
- Data link (layer 2) often imposes significantly smaller **Maximum Transmission Unit** (MTU) (e.g., Ethernet -1500 bytes usually)
- Fragmentation supports packet sizes greater than MTU and less than 2^{16}
- 13-bit Fragment Offset specifies offset of fragmented packet, in units of 8 bytes
- Receiver reconstructs IP packet from fragments, and delivers it to Transport Layer (layer 4) after reassembly



Ping-of-Death

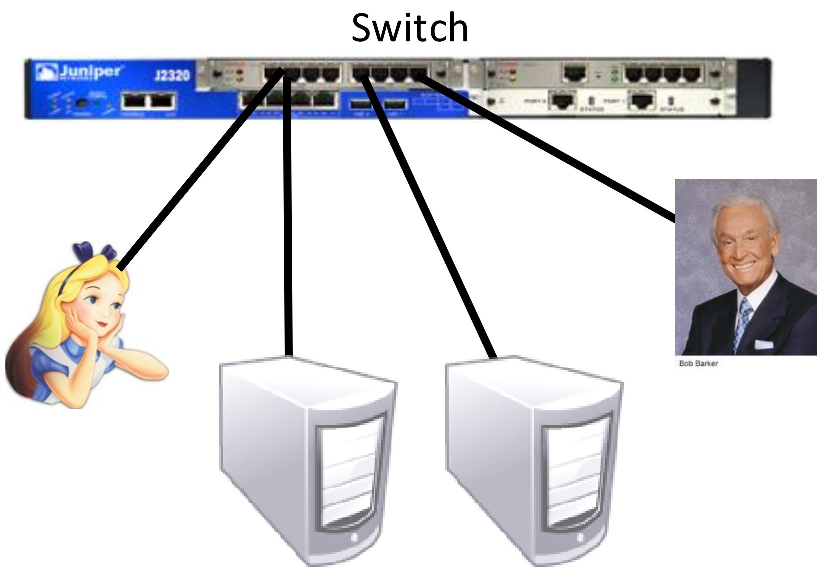
- Maximum packet size: 65,535 bytes
- Maximum 13-bit offset is $(2^{13} - 1) * 8 = 65,528$
- i.e., no fragment can have more than 65528 bytes as offset.
- **Attack**:
    - Send large size requests that has to be fragmented.
    - Fragmented data has to be reassembled in the destination using the offset value.
    - Craft the request such that data + offset exceeds the 65535 bytes size.
        - ...causing crashes, memory corruption and reboots
- Most OSes and firewalls have been hardened against PODs
- This was a popular pastime of early hackers

ARP Spoofing: Background: Ethernet Frames



ARP Spoofing: Background: ARP

- **Address Resolution Protocol (ARP):** Locates a host's link-layer (MAC) address
- Problem: How does Alice communicate with Bob over a LAN?
  - Assume Alice (10.0.0.1) knows Bob's (10.0.0.2) IP
  - LANs operate at layer 2 (there is no router inside of the LAN)
  - Messages are sent to the switch, and addressed by a host's link-layer (MAC) address
- Protocol:
  - Alice broadcasts: "Who has 10.0.0.2?"
  - Bob responses: "I do! And I'm at MAC f8:1e:df:ab:33:56."



ARP Spoofing

- Each ARP response overwrites the previous entry in ARP table -- **last response wins**!
- Attack: Forge ARP response
- Effects:
  - Man-in-the-Middle
  - Denial-of-service
- Also called **ARP Poisoning** or **ARP Flooding**

ARP Spoofing: Defenses

- Smart switches that remember MAC addresses
- Switches that assign hosts to specific ports

Worms

Worms

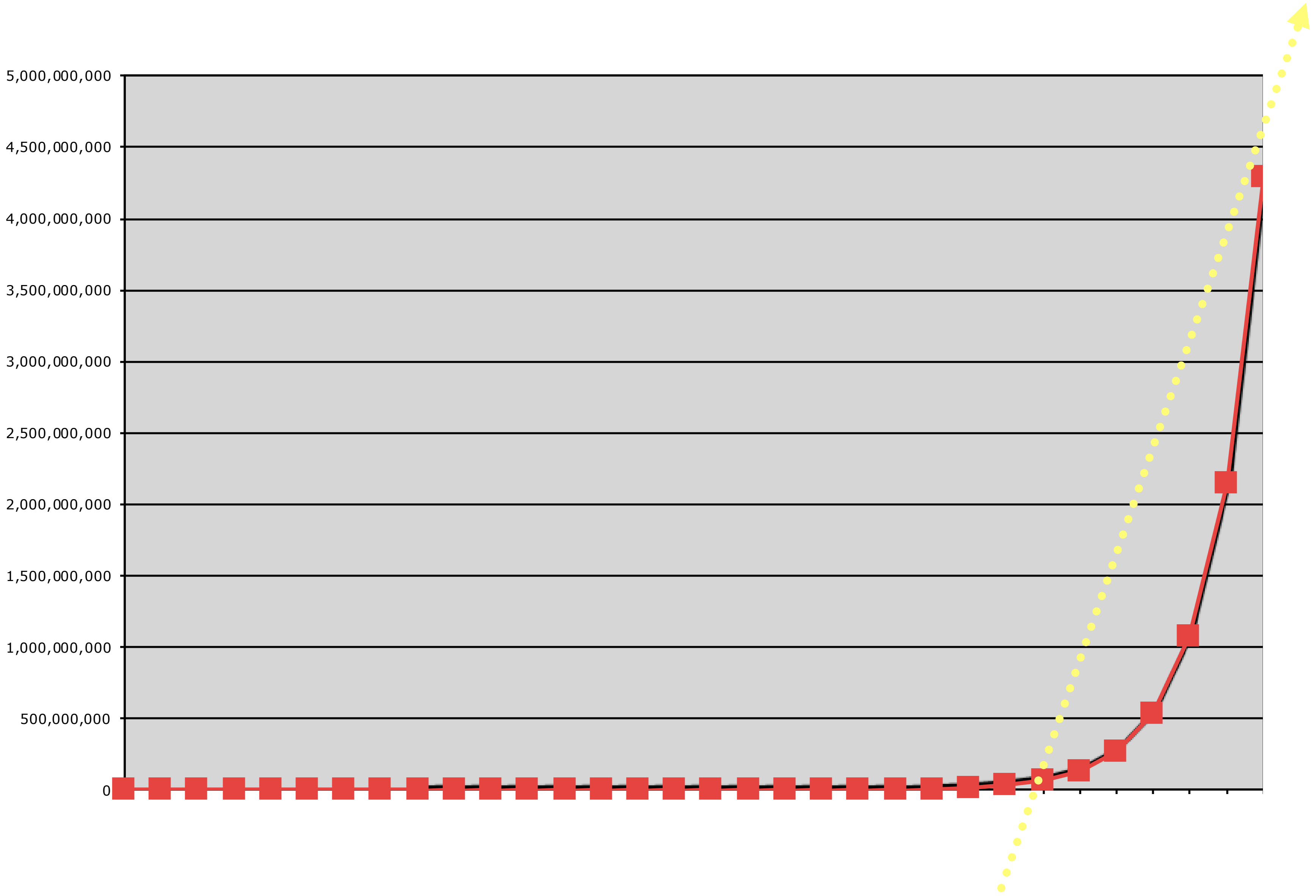
- A worm is a self-propagating program that:
  1. Exploits some vulnerability on a target host
  2. (often) imbeds itself into a host …
  3. Searches for other vulnerable hosts …
  4. Goto step 1

The Danger

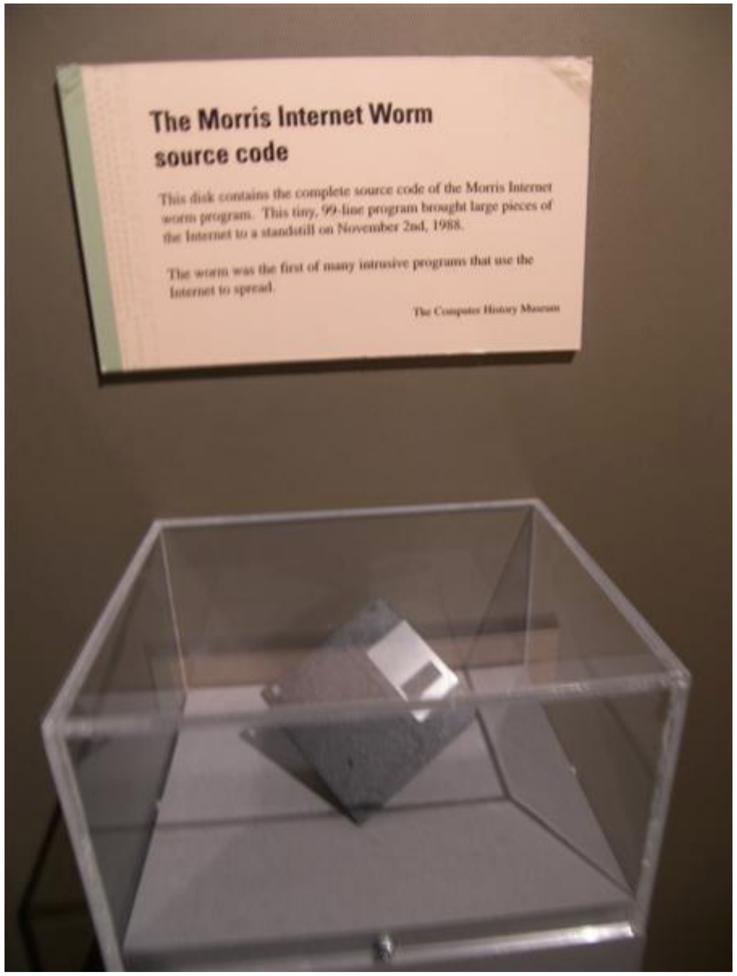
- What makes worms so dangerous is that *infection grows at an exponential rate*
  - A simple model:
    - s (search) is the time it takes to find vulnerable host
    - i (infect) is the time is take to infect a host
  - Assume that $t=0$ is the *worm outbreak*, the number of hosts at $t=j$ is

$$2^{(j/(s+i))}$$

The result


5,000,000,000
4,500,000,000
4,000,000,000
3,500,000,000
3,000,000,000
2,500,000,000
2,000,000,000
1,500,000,000
1,000,000,000
500,000,000
0

The Morris Worm



The Morris Internet Worm
source code

This disk contains the complete source code of the Morris Internet worm program. This tiny, 99-line program brought large pieces of the Internet to a standstill on November 2nd, 1988.

The worm was the first of many intrusive programs that use the Internet to spread.

The Computer History Museum

Robert Morris

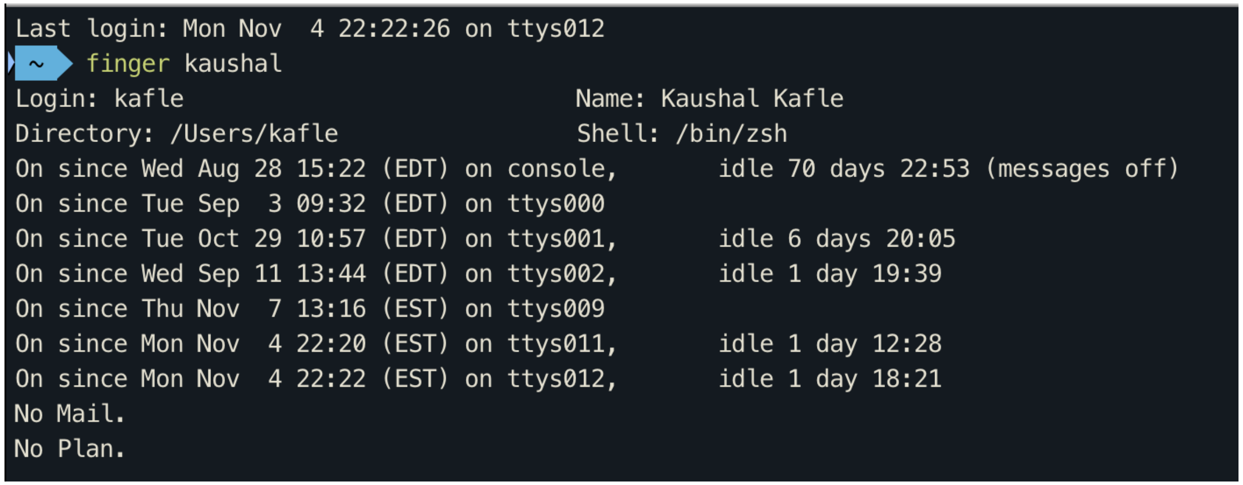
- 1988: Graduate student at Cornell University
- Son of Robert Morris, chief scientist at National Computer Security Center (division of NSA)
- Now a professor at MIT

November 2nd, 1988

- 6pm: someone ran a program at a computer at MIT
- The program collected host, network, and user info...
- ... and then spread to other machines running Sun 3, VAX, and some BSD variants
- ... rinse and repeat

Morris Worm: Attack Vectors

- rsh: terminal client with network (IP)-based authentication i.e., no passwords required
- fingerd: used *gets()* call without bounds checking (resulting in buffer overflow)
- sendmail: DEBUG mode allows remote user to run commands
  - lots of sendmail daemons running in DEBUG mode



```
Last login: Mon Nov  4 22:22:26 on ttys012
 ~  finger kaushal
Login: kafle                            Name: Kaushal Kafle
Directory: /Users/kafle                 Shell: /bin/zsh
On since Wed Aug 28 15:22 (EDT) on console,      idle 70 days 22:53 (messages off)
On since Tue Sep  3 09:32 (EDT) on ttys000
On since Tue Oct 29 10:57 (EDT) on ttys001,      idle 6 days 20:05
On since Wed Sep 11 13:44 (EDT) on ttys002,      idle 1 day 19:39
On since Thu Nov  7 13:16 (EST) on ttys009
On since Mon Nov  4 22:20 (EST) on ttys011,      idle 1 day 12:28
On since Mon Nov  4 22:22 (EST) on ttys012,      idle 1 day 18:21
No Mail.
No Plan.
```

Morris Worm: Propagation

- From victim device, generate list of IPs to infect.
  - E.g., target specific ports (e.g., port 79 for *fingerd*)
- Before infection, program checks for previous infection status.
  - If answer was no, worm would infect
  - *However,* to bypass admins who may report a false positives, the program copied itself regardless of status in 14% of cases.
- But... this allowed worm to spread much faster than anticipated, infecting the same machines multiple times
- Systems became overloaded with processes
- Swap space became exhausted, and machines failed

November 2nd, 1988

- Wednesday night: UC Berkeley captures copy of program
- 5AM Thursday: UC Berkeley builds *sendmail* patch to stop spread of worm
- Difficult to spread knowledge of fix
  - Not coincidentally, the Internet was running slow
- Economic impact in the range of $100k - $10M. Estimated to have affected ~6000 computers (*10% of ~60k total*)
  - Lesson: learn to debug! /s
    - If writing malicious code for educational purposes, use simulators!

Stuxnet

- First reported June 2010
- Exploited **unknown vulnerabilities**
  - Not one zero-day
  - Not two zero-days
  - Not three zero-days
  - But four zero-days!
    - print spooler bug
    - handful of escalation-of-privilege vulnerabilities

Stuxnet

- Spread through infected USB drives
  - bypasses “***air gaps***”
- Worm actively targeted SCADA systems (i.e., industrial control systems)
  - looked for WINCC or PCS 7 SCADA management system
    - attempted 0-day exploit
    - also tried using default passwords
  - apparently, specifically targeted Iran’s nuclear architecture

Stuxnet

- Once SCADA system compromised, worm attempts to reprogram Programmable Logic Controllers (PLCs)
- Forensics aggravated by lack of logging in SCADA systems

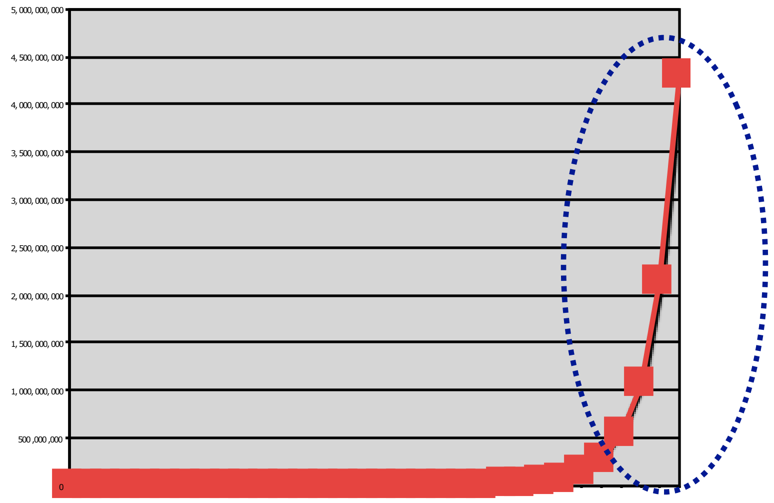
Worms and infection

- **The effectiveness of a worm is determined by how good it is at identifying vulnerable machines**
- Multi-vector worms use lots of ways to infect: e.g., network, email, drive by downloads, etc.
- Example scanning strategies:
  - **Random IP:** select random IPs; wastes a lot of time scanning “dark” or unreachable addresses (e.g., Code Red)
  - **Signpost scanning:** use info on local host to find new targets (e.g., Morris)
  - **Local scanning:** biased randomness
  - **Permutation scanning:** “hitlist” based on shared pseudorandom sequence; when victim is already infected, infected node chooses new random position within sequence

Other scanning strategies

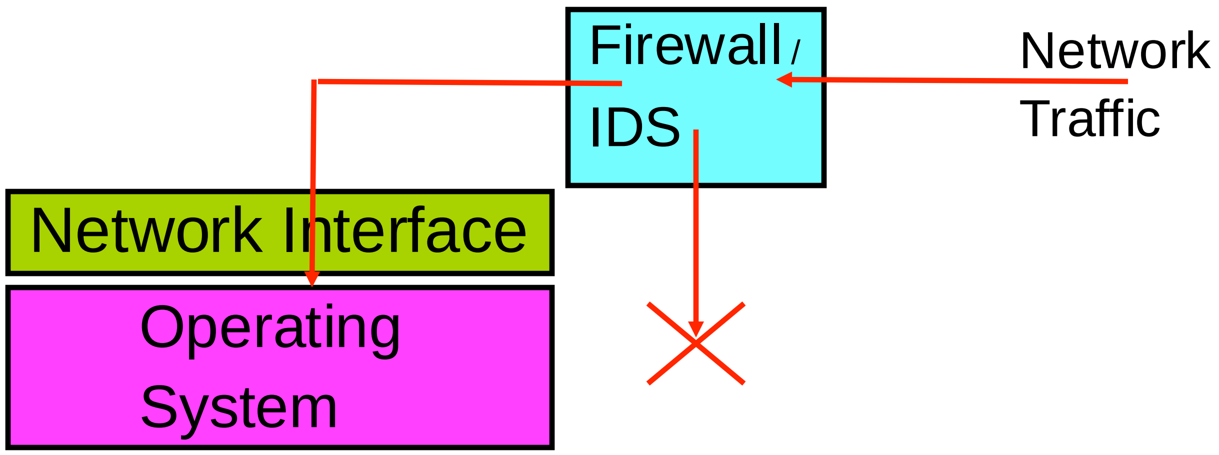
- The doomsday worm: a flash worm
- Exploration of worst case scenario (by Staniford et. al -> today's reading)
  - Create a hit list of all vulnerable hosts
    - Staniford et al. argue this is feasible
    - Would contain a 48MB list
  - Do the infect and split approach
  - Use a zero-day exploit (on Adobe flash)

- Result: saturate the Internet is less than *30 seconds*!



Worms: Defense Strategies

- (Auto) **patch** your systems: most large worm outbreaks have exploited known vulnerabilities (Stuxnet is an exception)
- **Heterogeneity**: use more than one vendor for your networks
- **IDS**: provides filtering for known vulnerabilities, such that they are protected immediately (analog to virus scanning)



- **Filtering**: look for unnecessary or unusual communication patterns, then drop them on the floor