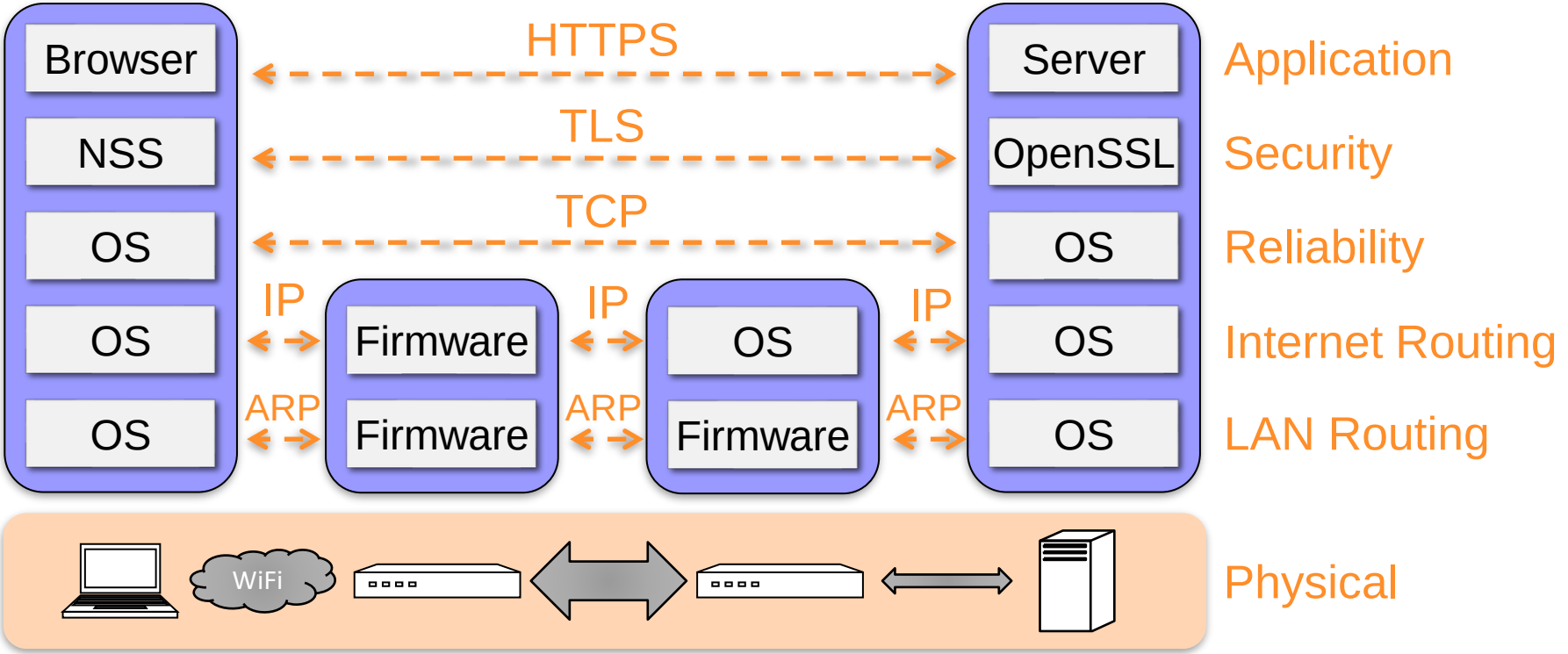
Network Security: ARP, IP, TCP, UDP

COMPUTER SECURITY

TARIQ ELAHI

Some slides adapted from those by Markulf Kohlweiss, Myrto Arapinis, Kami Vaniea, and Roberto Tamassia

Internet Stack (simplified)



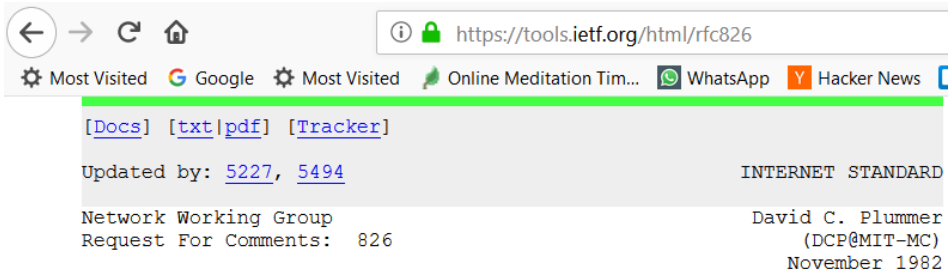
IP and MAC Addresses

- Devices on a local area network have
  - IP addresses (network layer)
  - MAC addresses (data link layer)
- IP addresses are used by high level protocols
- MAC addresses are used by low level protocols
- How to translate IP Addresses into MAC addresses?

The Problem:

The world is a jungle in general, and the networking game contributes many animals.

At nearly every layer of a network architecture there are several potential protocols that could be used.



For example, at a high level, there is TELNET and SUPDUP for remote login. Somewhere below that there is a reliable byte stream protocol, which might be CHAOS protocol, DOD TCP, Xerox BSP or DECnet. Even closer to the hardware is the logical transport layer, which might be CHAOS, DOD Internet, Xerox PUP, or DECnet.

Address Resolution Protocol (ARP)

- Connects the network layer to the data link layer
- Maps IP addresses to MAC addresses
- Based on broadcast messages and local caching
- Does not support confidentiality, integrity, or authentication
- Defined as a part of RFC 826 (IETF, Request For Comments)

ARP Messages

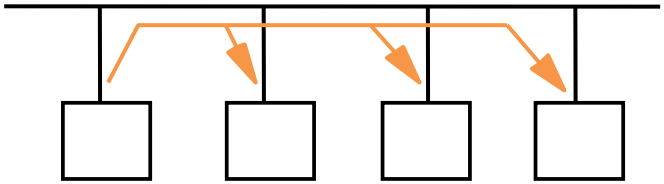
- ARP broadcasts requests of type

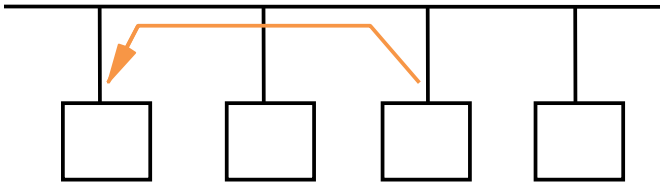
  who has <IP addressC >
  tell <IP addressA >

- Machine with <IP addressC> responds

  <IP addressC > is at <MAC address>

- Requesting machine caches response
- Network administrator configures IP address and subnet on each machine





ARP Cache

- The Linux, Windows and OSX command arp - a displays the ARP table

Internet Address	Physical Address	Type
128.148.31.1	00-00-0c-07-ac-00	dynamic
128.148.31.15	00-0c-76-b2-d7-1d	dynamic
128.148.31.71	00-0c-76-b2-d0-d2	dynamic
128.148.31.75	00-0c-76-b2-d7-1d	dynamic
128.148.31.102	00-22-0c-a3-e4-00	dynamic

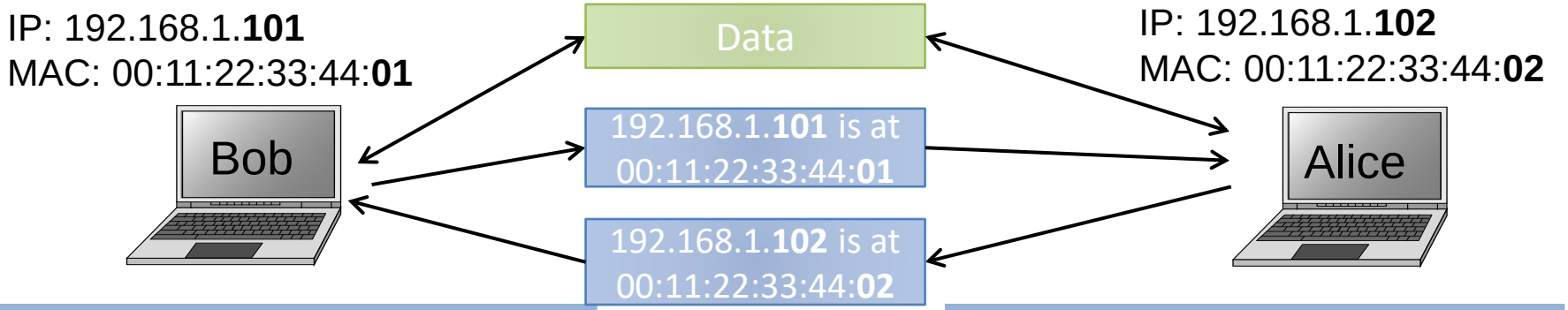
- Command arp –a –d flushes the ARP cache (Windows)
- ARP cache entries are stored for a configurable amount of time

ARP Cache Poisoning (aka ARP Spoofing)

- The ARP table is updated whenever an ARP response is received
- Requests are not tracked
- ARP announcements are not authenticated
- Machines trust each other
- A rogue machine can spoof other machines

ARP Normal Operation

- Normal operation
  - Alice communicates with Bob

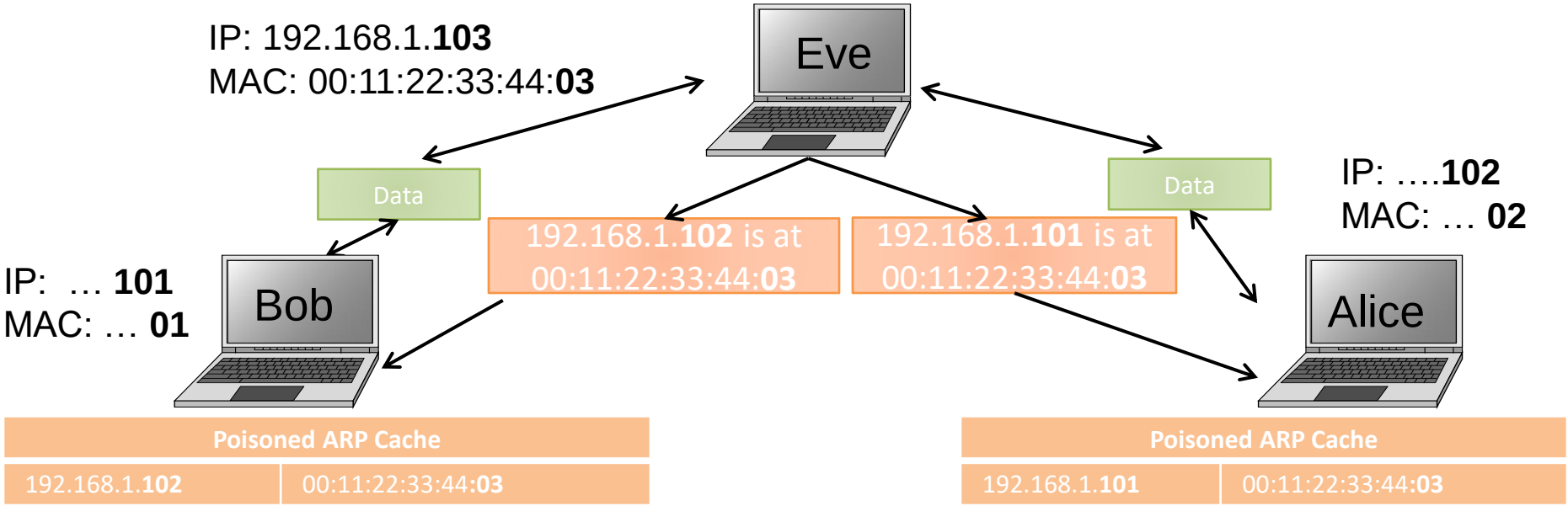


ARP Cache	
192.168.1.**102**	00:11:22:33:44:**02**

ARP Cache	
192.168.1.**101**	00:11:22:33:44:**01**

ARP Cache Poisoning Attack

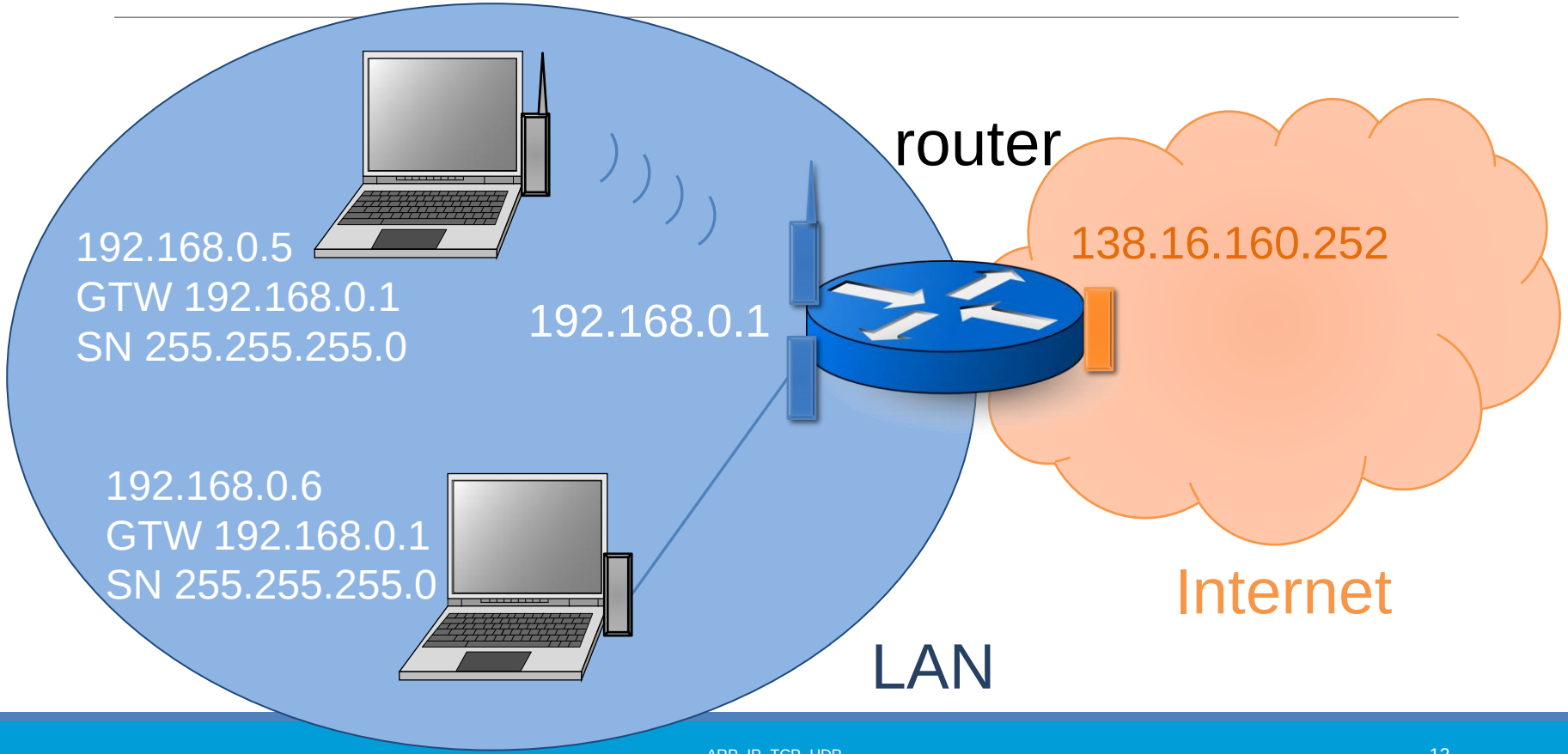
- Mal actor-in-the-middle attack (MITM)
  - ARP cache poisoning leads to eavesdropping



ARP Cache Poisoning (ARP Spoofing)

- Almost all ARP implementations are stateless
- An ARP cache updates every time that it receives an ARP reply
  - … even if it did not send any ARP request!
- Can “poison” ARP cache with gratuitous ARP replies
- Using static entries solves the problem but it is almost impossible to manage!

From the LAN to the Internet

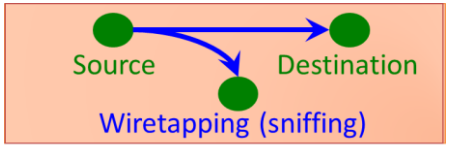


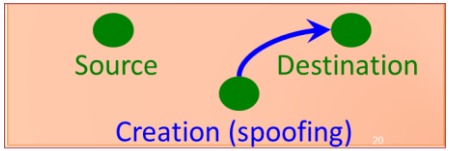
Edinburgh's IP Space

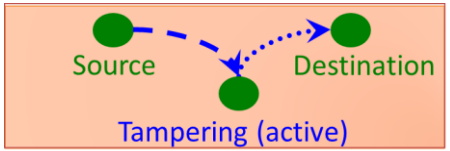
- Edinburgh is part of the autonomous system (AS786) of Jisc, for Joint Information Systems Committee, operate Janet
  - Class B network 129.215.0.0/16 (64K addresses)
- School of Informatics
  - 40 or so sub-networks, class C (/24) with 254 addresses or slightly larger
  - Server machines: 129.215.33.0/24
  - DICE desktop machines: 129.215.24.0/22
  - Laptops without a fixed IP address: 129.215.90.0/23

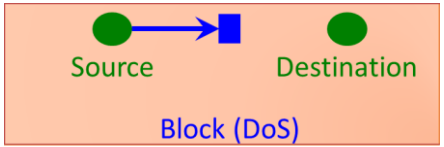
IP Vulnerabilities

- Unencrypted transmission
- No source authentication
  - Sender can spoof source address, making it difficult to trace packet back to attacker
- No integrity checking
  - Entire packet, header and payload, can be modified, enabling content forgeries, redirections, and mal actor-in-the-middle attacks
- No bandwidth constraints
  - Large number of packets can be injected into network to launch a denial-of-service attack
  - Broadcast addresses provide additional leverage









User Datagram Protocol

- UDP is a stateless, unreliable datagram protocol built on top of IP, i.e. it is at the transport layer
- UDP does not provide delivery guarantees or acknowledgments, which makes it efficient
- Can however distinguish data for multiple concurrent applications on a single host
- A lack of reliability implies applications using UDP must be ready to accept a fair amount of corrupted and lost data
  - Most applications built on UDP will suffer if they require reliability
  - VoIP, streaming video, and streaming audio all use UDP

Transmission Control Protocol

- Transport layer protocol for reliable data transfer, in-order delivery of messages and ability to distinguish multiple applications on same host
  - HTTP and SSH are built on top of TCP
- TCP is stateful: it keeps track of connection state in memory
- TCP packages a data stream into segments transported by IP
  - Order maintained by marking each packet with a sequence number
  - Every time TCP receives a packet, it sends out an acknowledgement (ACK) to indicate successful receipt of the packet
- TCP generally checks data transmitted by comparing a checksum of the data with a checksum encoded in the packet

Ports

- TCP (& UDP) supports concurrent applications on the same server
- Ports are 16 bit numbers identifying where data is directed
  - `>telnet 192.168.0.1:80` `https://example.co.uk:8080`
- The TCP header includes both a source and a destination port
- Ports 0 through 1023 are reserved for use by known protocols
  - E.g., HTTPS uses 443 and SSH uses 22
- Ports 1024 through 49151 are known as user ports, and are used for listening to connections

TCP Packet Format

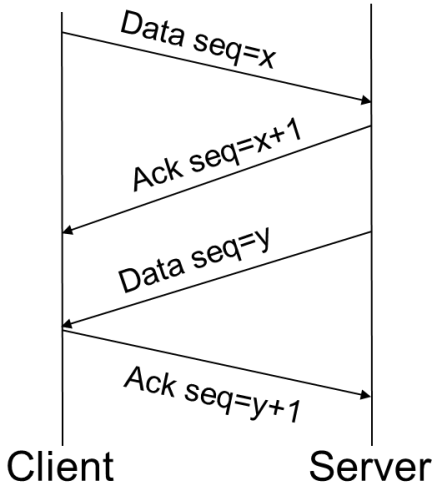
Bit Offset	0-3	4-7	8-15	16-18	19-31
0	Source Port			Destination Port	
32	Sequence Number				
64	Acknowledgment Number				
96	Offset	Reserved	Flags	Window Size	
128	Checksum			Urgent Pointer	
160	Options				
>= 160	Payload				

TCP Data Transfer

- During connection initialization using the three way handshake, initial sequence numbers are exchanged
- The TCP header includes a 16 bit checksum of the data and parts of the header, including the source and destination
- ACKs (or lack thereof) and window size are used by

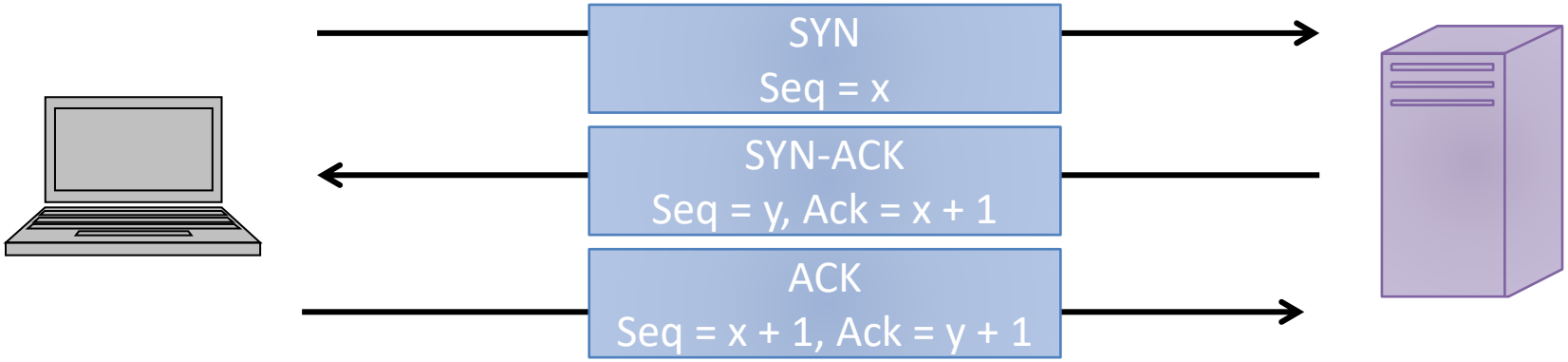
TCP to keep track of:

- packet loss
- network congestion
- flow control



Establishing TCP Connections

- TCP connections are established through a three-way handshake.
- The server generally is a passive listener, waiting for a connection request
- The client requests a connection by sending out a SYN packet
- The server responds by sending a SYN/ACK packet, acknowledging the connection
- The client responds by sending an ACK to the server, thus establishing connection

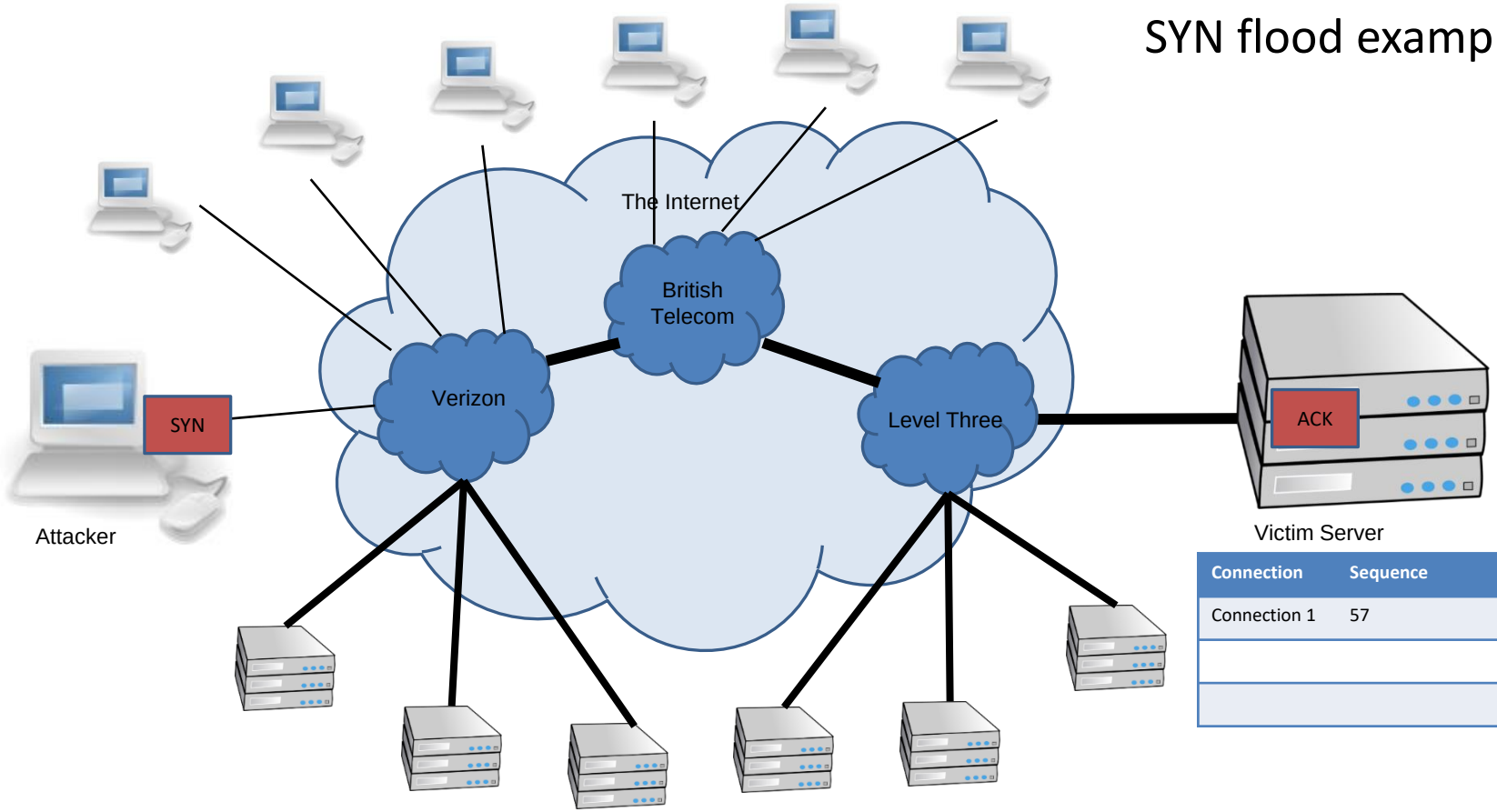


SYN Flooding

Send thousands of SYN requests to the victim

- Alice sends many SYN packets, without acknowledging any replies. Bob accumulates more SYN packets than he can handle (i.e. runs out of space in state table).

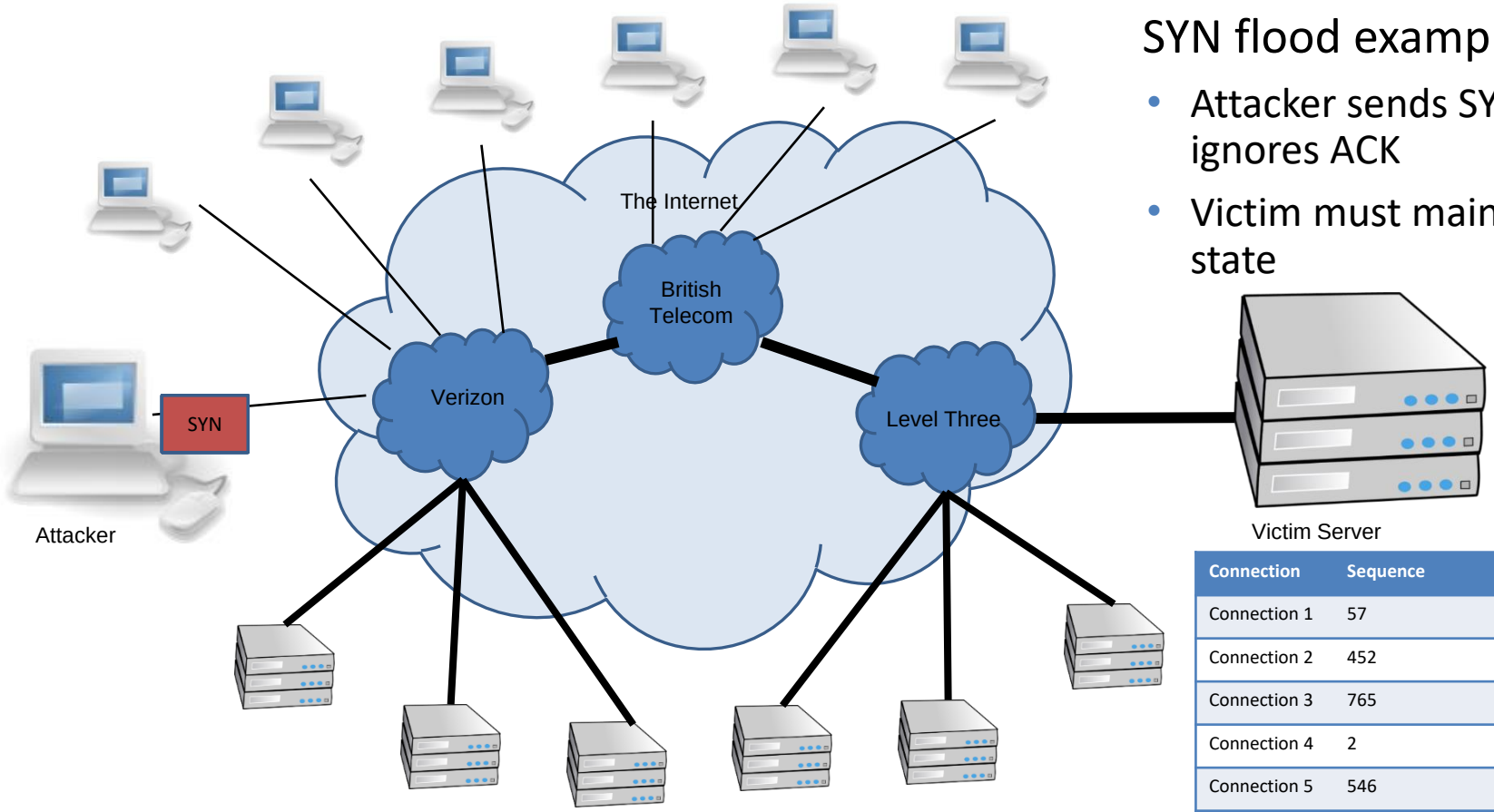
SYN flood example



Connection	Sequence	IP
Connection 1	57	1.1.1.1

SYN flood example

- Attacker sends SYN and ignores ACK
- Victim must maintain state



Connection	Sequence	IP
Connection 1	57	1.1.1.1
Connection 2	452	1.1.1.1
Connection 3	765	1.1.1.1
Connection 4	2	1.1.1.1
Connection 5	546	1.1.1.1
Connection 6	97	1.1.1.1
Connection 7	56	1.1.1.1
Connection 8	15	1.1.1.1

SYN Flooding

- Problems
  - Attribution – attacker uses their own IP which could be traced
  - Bandwidth – attacker uses their own bandwidth which is likely smaller than a server's
- Effective against a small target
  - Someone running a game server in their home
- Not effective against a large target
  - Company website

Spoofing: forged TCP packets

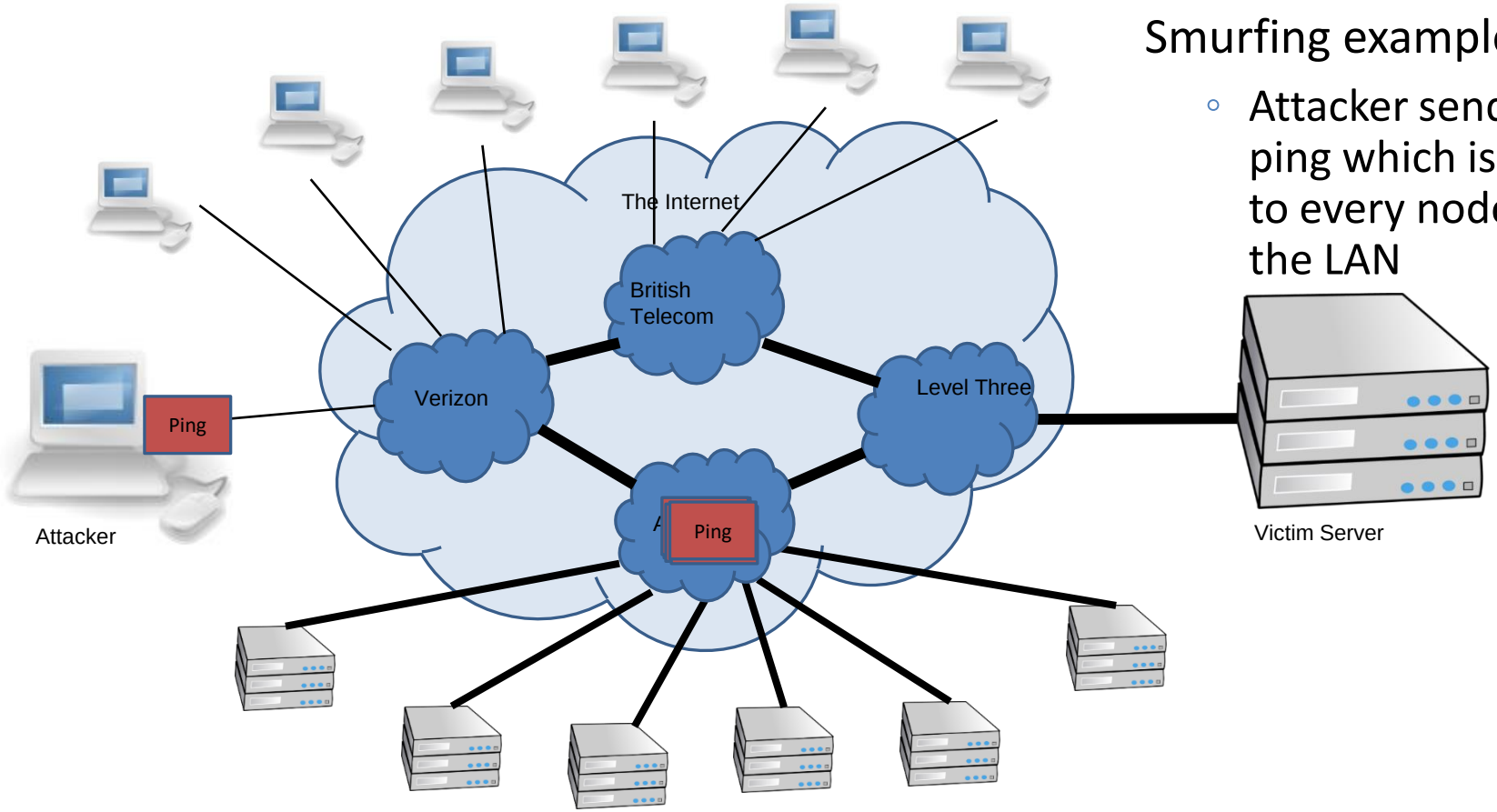
- Same as SYN flooding, but forge the source of the TCP packet
- Advantages:
  - Harder to trace
  - ACKs are sent to a second computer, less attacker bandwidth used
- Problems:
  - Ingress filtering is commonly used to drop packets with source addresses outside their origin network fragment.

Smurfing (directed broadcast)

- The smurfing attack exploits ICMP (Internet Control Message Protocol) ping requests whereby remote hosts respond to echo packets to say they are online
- Some networks respond to pings to broadcast addresses. We call these networks “Smurf amplifiers”.
- Idea: Ping a LAN on a broadcast address, then all hosts on the LAN reply to the sender of the ping
- Attack
  - Make a forged packet with the victim’s IP address as the source
  - Send it to a Smurf amplifier, which then causes a huge number of replies to the victim
- This is a form of reflection attack

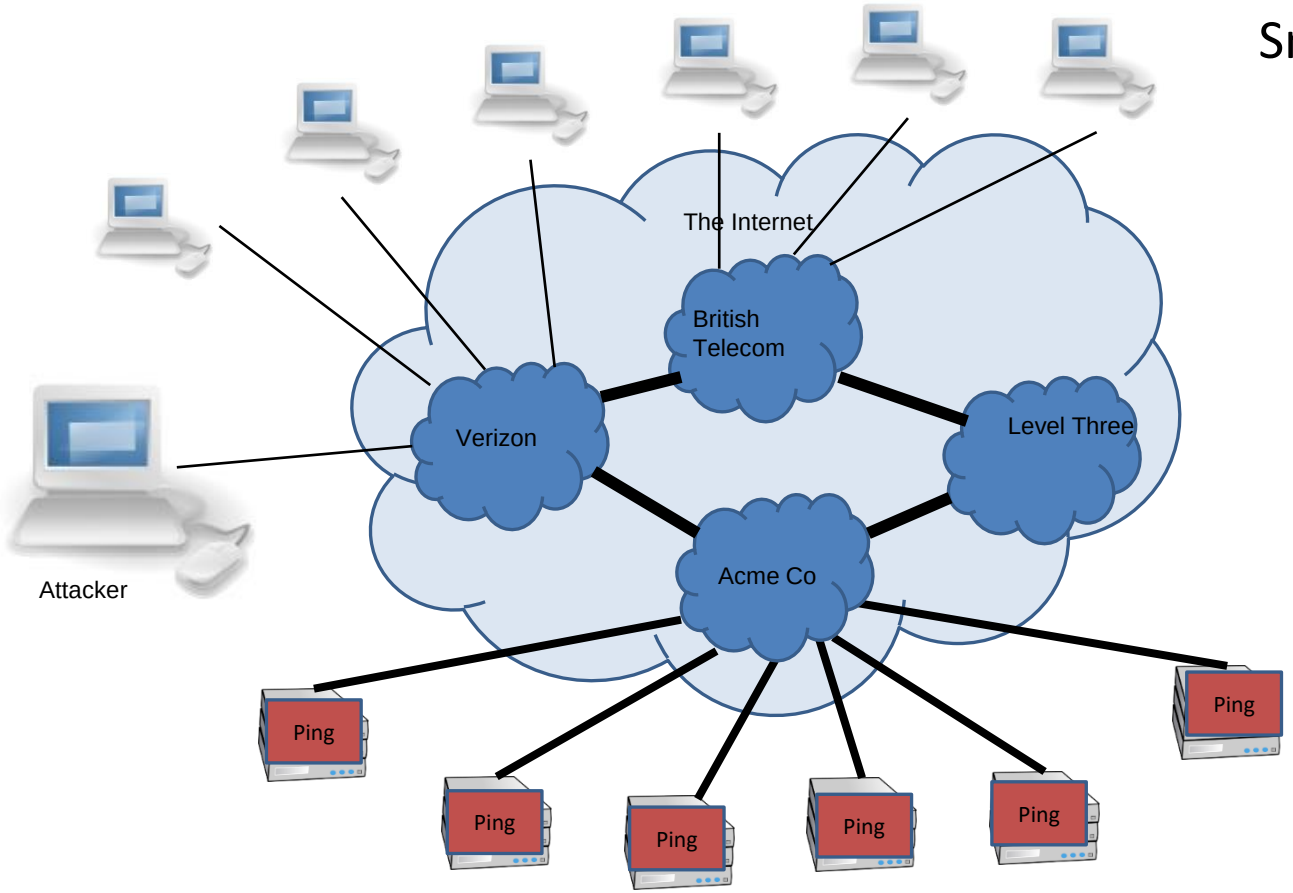
Smurfing example

- Attacker sends 1 ping which is sent to every node on the LAN



Smurfing example

- Each node responds to victim



Victim Server

LANs that allow Smurf attacks are badly configured. One approach is to blacklist these LANs.

powertech

Smurf Amplifier Registry (SAR)
http://www.powertech.no/smurf/

Current top ten smurf amplifiers (updated every 5 minutes)
(last update: 2016-01-17 23:31:02 CET)

Network	#Dups	#Incidents	Registered at	Home AS
212.1.130.0/24	38	0	1999-02-20 09:41	AS9105
204.158.83.0/24	27	0	1999-02-20 10:09	AS3354
209.241.162.0/24	27	0	1999-02-20 08:51	AS701
159.14.24.0/24	20	0	1999-02-20 09:39	AS2914
192.220.134.0/24	19	0	1999-02-20 09:38	AS685
204.193.121.0/24	19	0	1999-02-20 08:54	AS701
198.253.187.0/24	16	0	1999-02-20 09:34	AS22
164.106.163.0/24	14	0	1999-02-20 10:11	AS7066
12.17.161.0/24	13	0	2000-11-29 19:05	not-analyzed
199.98.24.0/24	13	0	1999-02-18 11:09	AS6199

2457713 networks have been probed with the SAR
56 of them are currently broken
193885 have been fixed after being listed here

Distributed Denial of Service (DDoS)

A large number of machines work together to perform an attack that prevents valid users from accessing a service.

Common examples:

- Slashdot effect – a large number of valid users all try and access at once.
- Botnets
- Amazon web services

Article Talk Read Edit View history Search Wikipedia

Mirai (malware)

From Wikipedia, the free encyclopedia

Mirai (Japanese: 未来, lit. 'future') is a malware that turns networked devices running Linux into remotely controlled "bots" that can be used as part of a botnet in large-scale network attacks. It primarily targets online consumer devices such as IP cameras and home routers.[1] The Mirai botnet was first found in August 2016[2][3] by MalwareMustDie,[4] a whitehat malware research group, and has been used in some of the largest and most disruptive distributed denial of service (DDoS) attacks, including an attack on 20 September 2016[5] on computer security journalist Brian Krebs' web site, an attack on French web host OVH,[6] and the October 2016 Dyn cyberattack.[7][8][9] According to a chat log between Anna-senpai and Robert Coelho, Mirai was named after the 2011 TV anime series *Mirai Nikki*.[10]

Mirai

Original author(s)	Paras Jha, Josiah White and Dalton Norman
Repository	github.com/jgamblin/Mirai-Source-Code
Written in	C (agent), Go (controller)
Operating system	Linux
Type	Botnet
License	GNU General Public License v3.0
Website	github.com/jgamblin/Mirai-Source-Code

Mirai spread by first entering a rapid scanning phase (where it asynchronously and "statelessly" sent TCP SYN probes to pseudorandom IPv4 addresses, excluding those in a hard-coded IP blacklist, on Telnet TCP ports 23 and 2323.

Attack Type	Attacks	Targets	Class
HTTP flood	2,736	1,035	A
UDP-PLAIN flood	2,542	1,278	V
UDP flood	2,440	1,479	V
ACK flood	2,173	875	S
SYN flood	1,935	764	S
GRE-IP flood	994	587	A
ACK-STOMP flood	830	359	S
VSE flood	809	550	A
DNS flood	417	173	A
GRE-ETH flood	318	210	A

Table 9: **C2 Attack Commands**—Mirai launched 15,194 attacks between September 27, 2016–February 28, 2017. These include [A]pplication-layer attacks, [V]olumetric attacks, and TCP [S]tate exhaustion, all of which are equally prevalent.

What We Have Learned

- ARP protocol
- ARP poisoning attack
  - MitM attack on a LAN
- Network and transport layer protocols
  - ICMP
  - TCP for reliable transmission
  - UDP when packet loss/corruption is tolerated
  - DoS Attacks: SYN flooding, Smurf
- Lack of built-in security in network protocols
  - In future lectures we'll see how security must be incorporated at the application layer (e.g. TLS)