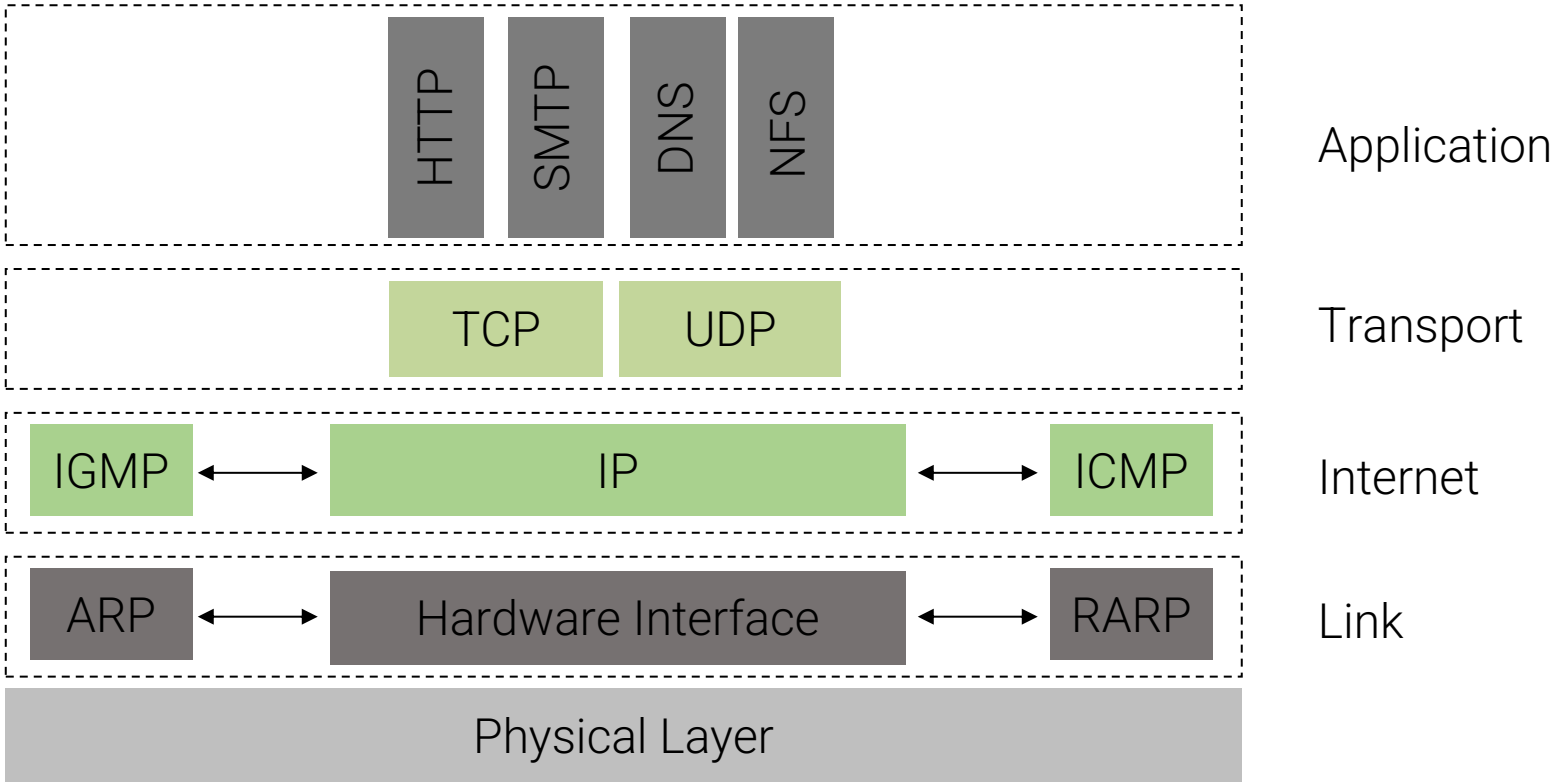
Network Security

The Internet Protocol Suite

- Set of protocols used to transport data between nodes of a network
- Also known as the TCP/IP Protocol Suite
- Based on abstraction and encapsulation
- Link protocols
- Internet protocols
- Transport protocols
- Application protocols

TCP/IP Layering



IP Addresses

- Each host has one or more IP addresses for each network interface
- IPv4 addresses are composed of 32 bit
- Represented in dotted-decimal notation: 128.111.48.69
- Initially divided in classes:
  - Class A (0): netid=7 bit (128 networks, actually 1-126), hostid=24 bit (16,777,216 hosts)
  - Class B (10): netid=14 bit (16384 networks), hostid=16 bit (65536 hosts)
  - Class C (110): netid=21 bit (2097152 networks), hostid=8 bit (256 hosts)
  - Class D - Multicast (1110): multicast addresses
  - Class E (1111): reserved or future use

Classless Inter-Domain Routing (CIDR)

- Allocation of large chunks of IP addresses to single organizations wasted an enormous number of IP addresses
  - The public IPv4 space has been exhausted in February 2011
- CIDR, introduced in 1993, is an addressing scheme which allows for more efficient allocation of IP addresses than the old class-based address scheme
- The netid/hostid boundary can be placed on any bit between 13 and 27
  - 32 hosts minimum
  - 524,288 hosts maximum

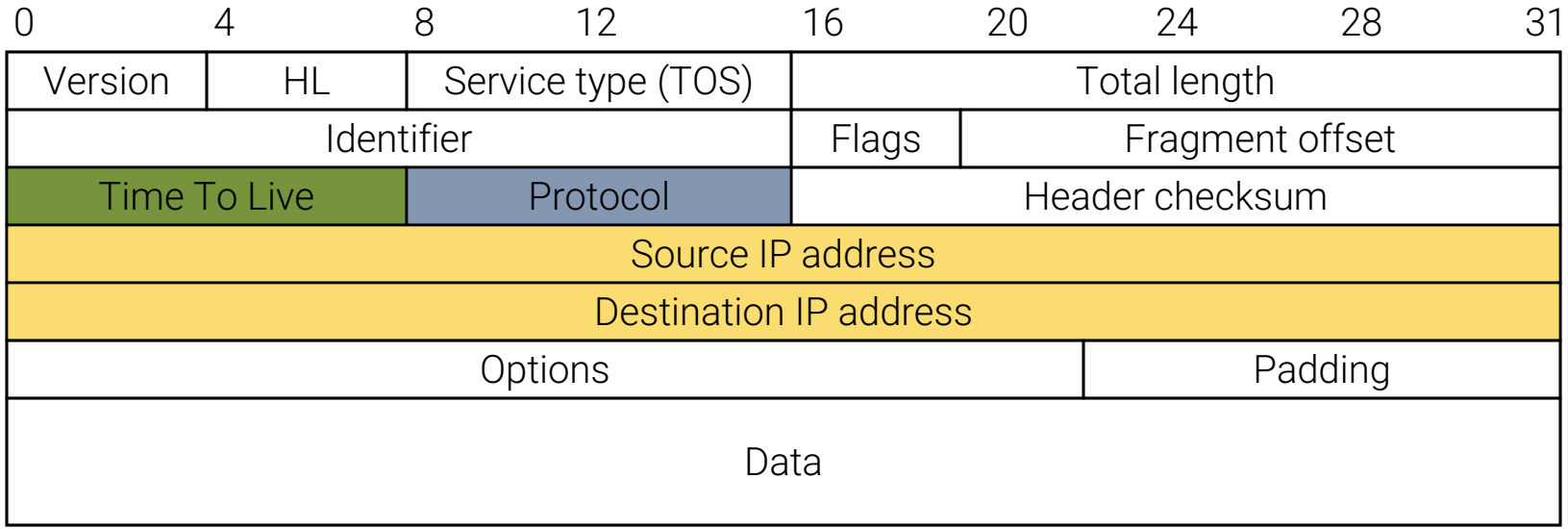
Special Addresses

- As source and destination address
  - Loopback interface: 127.X.X.X (usually 127.0.0.1)
- As source address
  - netid=0, hostid=0 or hostid=XXX: this host on this net (used in special cases, such as booting procedures)
- As destination address
  - All bits set to 1: local broadcast
  - netid + hostid with all bits set to 1: net-directed broadcast to netid
- Reserved addresses (RFC 1597):
  - 10.0.0.0 - 10.255.255.255
  - 172.16.0.0 - 172.31.255.255
  - 192.168.0.0 - 192.168.255.255

Internet Protocol (IP)

- The IP protocol represents the “glue” of the Internet
- The IP protocol provides a connectionless, unreliable, best-effort datagram delivery service (delivery, integrity, ordering, non-duplication, and bandwidth is not guaranteed)
- IP datagrams can be exchanged between any two nodes (provided they both have an IP address)
- For direct communication IP relies on a number of different lower-level protocols, e.g., Ethernet, FDDI, RS-232

IP Datagram


0
4
8
12
16
20
24
28
31
Version
HL
Service type (TOS)
Total length
Identifier
Flags
Fragment offset
Time To Live
Protocol
Header checksum
Source IP address
Destination IP address
Options
Padding
Data

IP Header

- Normal size: 20 bytes
- Version (4 bits): current value=4 (IPv4)
- Header length (4 bits): number of 32-bit words in the header, including options (max header size is 60 bytes)
- Type of service (8 bits): priority (3 bits), quality of service (4 bits), and an unused bit
- Total length (16 bits): datagram length in bytes (max size is 65535 bytes)
- Id (16 bits): unique identifier for the datagram (usually incremented by one)

IP Header

- Flags (3 bits) and offset (13 bits): used for fragmentation
- Time To Live (8 bits): specifies the max number of hops in the delivery process
- Protocol (8 bits): specifies the protocol encapsulated in the datagram data (e.g., TCP)
- Header checksum (16 bits): checksum calculated over the IP header
- Addresses (32+32 bits): IP addresses of the source and destination of the datagram

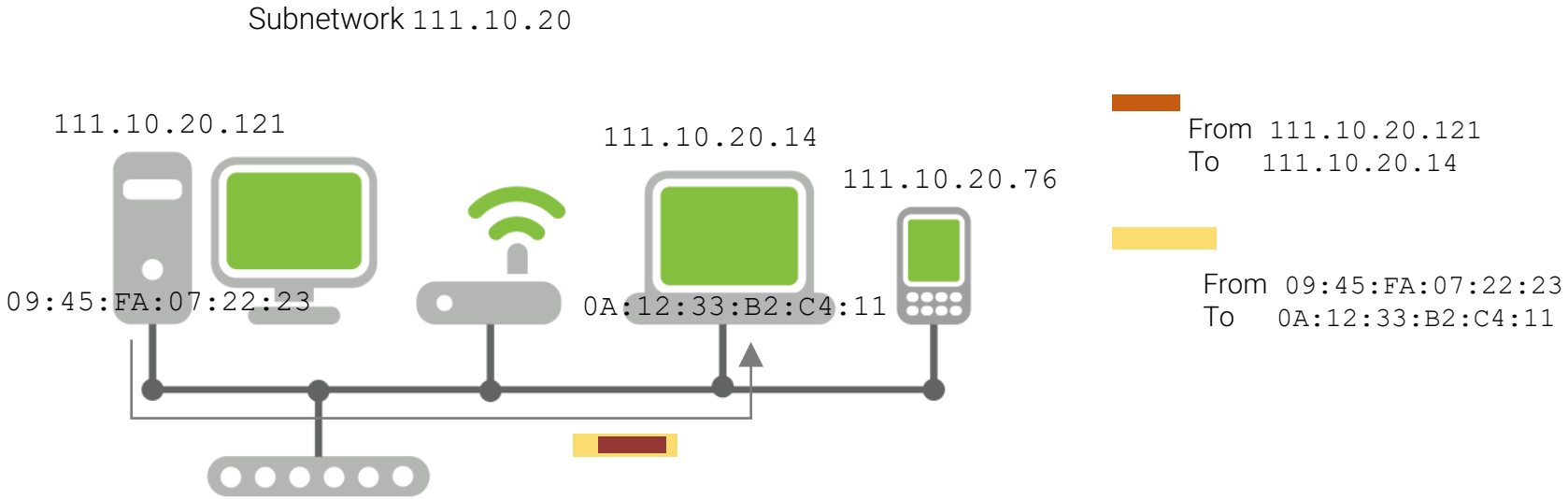
IP Options

- Variable length
- Identified by first byte
  - Record route: each router records its IP address
  - Time stamp: each router records its IP address and time
  - Source route: specifies a list of IP addresses that must be traversed by the datagram
  - Many others…

IP Encapsulation

IP: Direct Delivery

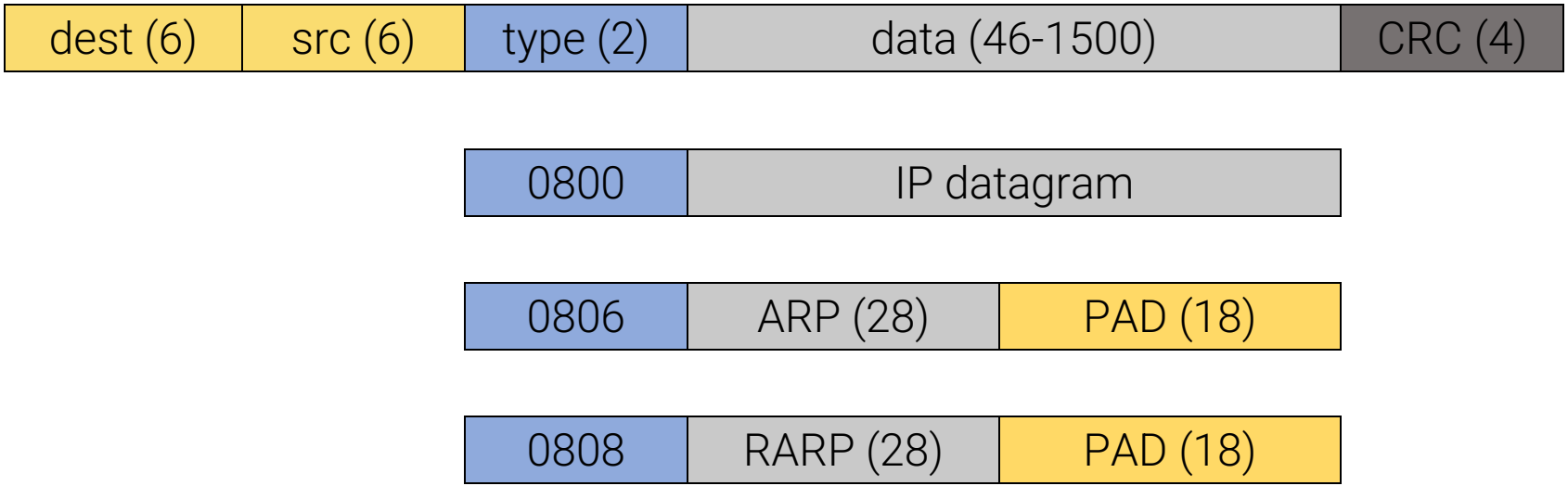
- If two hosts are in the same physical network the IP datagram is encapsulated in a lower-level protocol and delivered directly



Ethernet

- Widely-used link-layer protocol
- Uses CSMA/CD (Carrier Sense, Multiple Access with Collision Detection)
- Destination address: 48 bits (e.g., 09:45:FA:07:22:23)
- Source address: 48 bits
- Type: 2 bytes (IP, ARP, RARP)
- Data:
  - Min 46 bytes (padding may be needed)
  - Max 1500 bytes
- CRC: Cyclic Redundancy Check, 4 bytes

Ethernet Frame



Address Resolution Protocol

- The address resolution protocol allows a host to map the IP address to the link-level address associated with the peer's hardware interface (e.g., Ethernet) to be used in direct delivery
- ARP messages are encapsulated in the underlying link-level protocol

Address Resolution Protocol

- Host A wants to know the hardware address associated with the IP address of host B
- Host A broadcasts a special message to all the hosts on the same physical link
- Host B answers with a message containing its own link-level address
- Host A keeps the answer in its cache
- To optimize traffic exchange, when host A sends its request it also includes its own IP address
- The receiver of the ARP request will cache the requester mapping

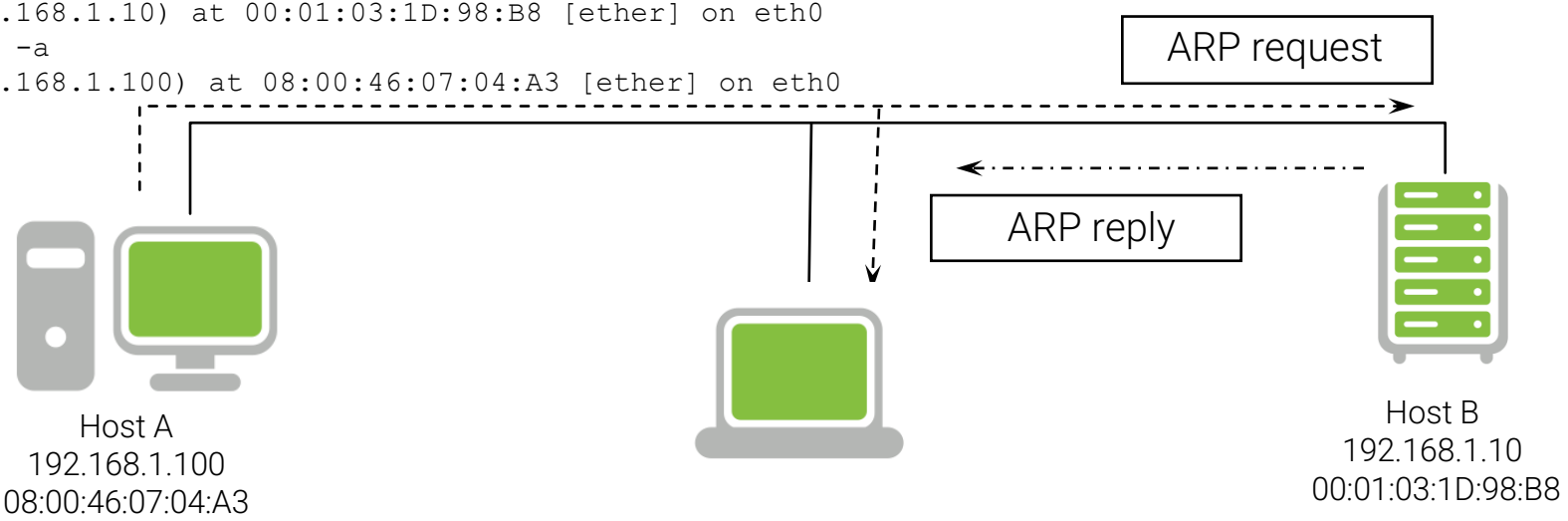
ARP Messages

Hw type	Prot type	Hw size	Prot size	Op	Sender Ether	Sender IP	Target Ether	Target IP

- Hardware (2 bytes), protocol (2 bytes), hardware size (1 byte), and protocol size (1 byte) specify the link and network addresses to be mapped (usually Ethernet and IP, respectively) [0x0001, 0x0800, 6, 4]
- OP field specifies if this is an ARP request or an ARP reply (1= ARP request, 2=ARP reply)
- Sender Ethernet/IP: data of the requester
- Target Ethernet: empty in a request
- Target IP: requested IP address

ARP Request

```
hosta# arp -a
hosta# ping 192.168.1.10
8:0:46:7:4:a3  ff:ff:ff:ff:ff:ff arp 60: arp who-has 192.168.1.10 tell 192.168.1.100
0:1:3:1d:98:b8 8:0:46:7:4:a3     arp 60: arp reply 192.168.1.10 is-at 0:1:3:1d:98:b8
8:0:46:7:4:a3  0:1:3:1d:98:b8    ip  98: 192.168.1.100 > 192.168.1.10: icmp: echo request
0:1:3:1d:98:b8 8:0:46:7:4:a3     ip  98: 192.168.1.10 > 192.168.1.100: icmp: echo reply
hosta# arp -a
hostb (192.168.1.10) at 00:01:03:1D:98:B8 [ether] on eth0
hostb# arp -a
hosta (192.168.1.100) at 08:00:46:07:04:A3 [ether] on eth0
```



Local Area Network Attacks

- Goals
  - Impersonation of a host
  - Denial of service
  - Access to information
  - Tampering with delivery mechanisms
- Sniffing
- Spoofing
- Hijacking

Hubs vs. Switches

- Early network switches were simple hubs
  - All traffic was broadcasted to all ports
- Modern network switches keep track of which interface is connected to each port
  - All broadcast traffic is sent to all connected hosts
  - All directed traffic is sent to the ports associated with the referenced hardware address

Network Sniffing

- Technique at the basis of many attacks
- The attacker sets his/her network interface in promiscuous mode
- If switched Ethernet is used, then the switch must be “convinced” that a copy of the traffic needs to be sent to the port of the sniffing host

Why Sniffing?

- Many protocols (FTP, POP, HTTP, IMAP) transfer authentication information in the clear
- By sniffing the traffic it is possible to collect usernames/passwords, files, mail, etc.
- Even encrypted sessions can leak information (e.g., SSL parameters can be used to fingerprint hosts or operating systems)

Sniffing Tools

- Tools to collect, analyze, and reply traffic
- Routinely used for traffic analysis and troubleshooting
- Command-line tools
  - tcpdump: collects traffic
  - tcpflow: reassembles TCP flows
  - tcpreplay: re-sends recorded traffic
- GUI tools
  - Wireshark
    - Supports TCP reassembling
    - Provides parsers for a number of protocols

TCPDump: Understanding the Network

- TCPDump is a tool that analyzes the traffic on a network segment
- One of the most used/most useful tools
- Based on libpcap, which provides a platform-independent library and API to perform traffic sniffing
- Allows one to specify an expression that defines which packets have to be printed
- Requires root privileges to be able to set the interface in promiscuous mode (privileges not needed when reading from file)

TCPDump: Command Line Options

- -e: print link-level addresses
- -n: do not translate IP addresses to FQDN names
- -x: print each packet in hex
- -i: use a particular network interface
- -r: read packets from a file
- -w: write packets to a file
- -s: specify the amount of data to be sniffed for each packet (e.g., set to 65535 to get the entire IP packet)
- -f: specify a file containing the filter expression

TCPDump: Filter Expression

- A filter expression consists of one or more primitives
- Primitives are composed of a qualifier and an id
- Qualifiers
  - type: defines the kind of entity
    - host (e.g., “host longboard”, where “longboard” is the id)
    - net (e.g., “net 128.111”)
    - port (e.g., “port 23”)
  - dir: specifies the direction of traffic
    - src (e.g., “src host longboard”)
    - dst (e.g., “dst port 22”)

TCPDump: Filter Expression

- Qualifiers (continued)
  - proto: specifies a protocol of interest
    - ether (e.g., “ether src host 00:65:FB:A6:11:15”)
    - ip (e.g., “ip dst net 192.168.1”)
    - arp (e.g., “arp”)
    - rarp (e.g., “rarp src host 192.168.1.100”)
- Operators can be used to create complex filter expression
  - and, or, not (e.g., “host shortboard and not port ssh”)
- Special keywords
  - gateway: checks if a packet used a host as a gateway
  - less and greater: used to check the size of a packet
  - broadcast: used to check if a packet is a broadcast packet

TCPDump: Filter Expression

- Other operators
  - Relational: <, >, >=, <=, =, !=
  - Binary: +, -, *, /, &, |
- Access to packet data
  - proto [expr : size] where expr is the byte offset and size is an optional indicator of the number of bytes if interest (1, 2, or 4)
    - ip[0] & 0xf != 5 to filter only IP datagrams with options

TCPDump: Examples

- # tcpdump -i eth0 -n -x
- # tcpdump -s 65535 -w traffic.dump src host hitchcock
- % tcpdump -r traffic.dump arp
- # tcpdump arp[7] = 1
- # tcpdump gateway csgw and \(port 21 or port 20 \)

Libpcap

- Library to build sniffers in C
- pcap_lookupdev
  - looks up a device
- pcap_open_live
  - opens a device and returns a handle
- pcap_open_offline and pcap_dump_open
  - read from and save packets to files
- pcap_compile and pcap_setfilter
  - set a tcpdump-like filter
- pcap_loop
  - register a callback to be invoked for each received packet

Packet Structure

- Header is returned in structure

```
struct pcap_pkthdr {
        struct timeval ts; /* time stamp */
        bpf_u_int32 caplen; /* length of portion */
        bpf_u_int32 len; /* length this packet (off wire) */
};
```

- The actual packet is returned as a pointer to memory
- Packet can be parsed by “casting” it with protocol-specific structs
- Whenever dealing with packets take into account endianness
  - Use ntohs, htons, ntohl, htonl

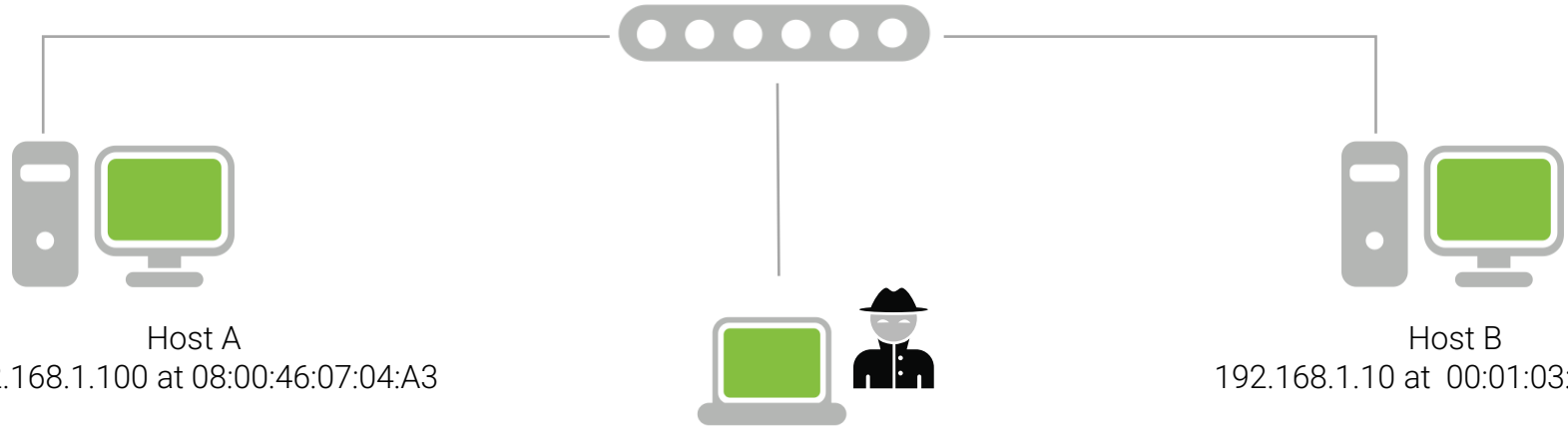
Switched Environments

- Switched Ethernet does not allow direct sniffing
- MAC flooding
  - Switches maintain a table with MAC address/port mappings
  - In some cases, flooding the switch with bogus MAC addresses will overflow the table's memory and revert the behavior from "switch" to "hub"
- MAC duplicating/cloning
  - Attacker reconfigures his/her host to have the same MAC address as the target machine
  - The switch will record this in its table and send the traffic to the attacker machine (or possibly both)

ARP Spoofing

- Goal: sniff all traffic between two hosts in a switched environment
- The attack leverages the stateless nature of the ARP protocol
  - Replies without a request will be accepted
- The attacker host sends spoofed ARP messages to the two victim hosts, poisoning their cache
- The victim host sends their IP packets to the attacker host
- The attacker host acts has a router

ARP Spoofing



Host A
192.168.1.100 at 08:00:46:07:04:A3

192.168.1.10	00:01:03:1D:98:B8

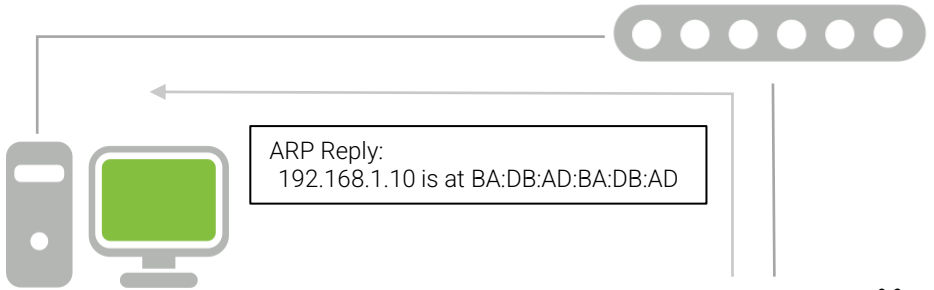
Host C
192.168.1.137 at BA:DB:AD:BA:DB:AD

192.168.1.100	08:00:46:07:04:A3
192.168.1.10	00:01:03:1D:98:B8

Host B
192.168.1.10 at 00:01:03:1D:98:B8

192.168.1.100	08:00:46:07:04:A3

ARP Spoofing



Host A
192.168.1.100 at 08:00:46:07:04:A3

192.168.1.10	00:01:03:1D:98:B8

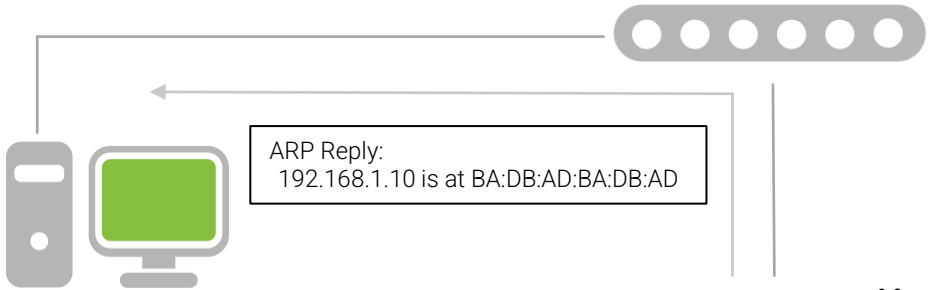
Host C
192.168.1.137 at BA:DB:AD:BA:DB:AD

192.168.1.100	08:00:46:07:04:A3
192.168.1.10	00:01:03:1D:98:B8

Host B
192.168.1.10 at 00:01:03:1d:98:b8

192.168.1.100	08:00:46:07:04:A3

ARP Spoofing



Host A
192.168.1.100 at 08:00:46:07:04:A3

192.168.1.10	BA:DB:AD:BA:DB:AD

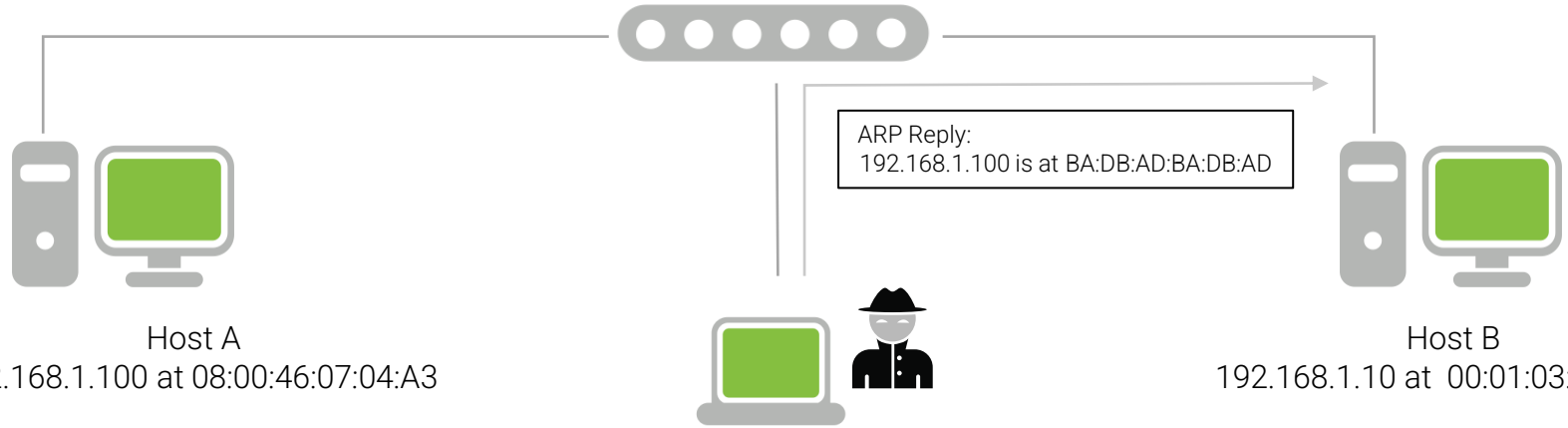
Host C
192.168.1.137 at BA:DB:AD:BA:DB:AD

192.168.1.100	08:00:46:07:04:A3
192.168.1.10	00:01:03:1D:98:B8

Host B
192.168.1.10 at 00:01:03:1d:98:b8

192.168.1.100	08:00:46:07:04:A3

ARP Spoofing

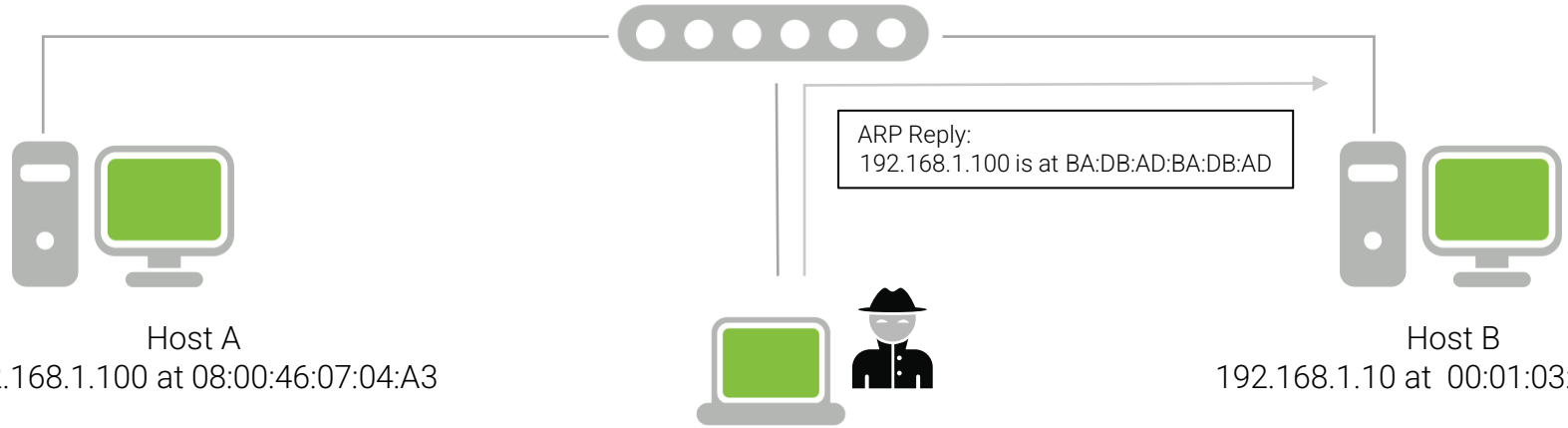


192.168.1.10	BA:DB:AD:BA:DB:AD

192.168.1.100	08:00:46:07:04:A3
192.168.1.10	00:01:03:1D:98:B8

192.168.1.100	08:00:46:07:04:A3

ARP Spoofing

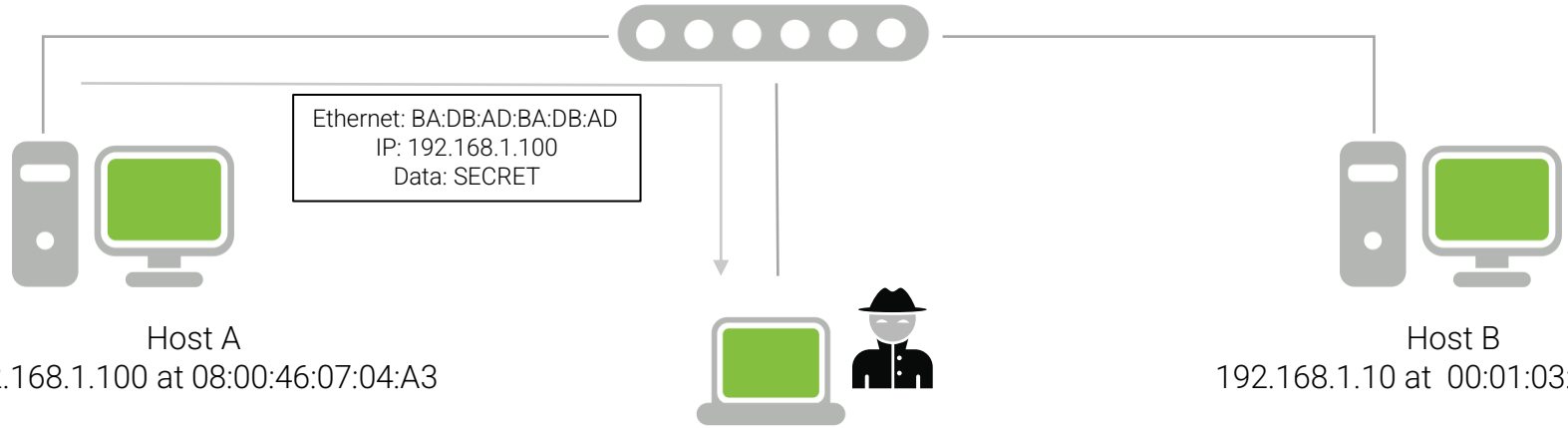


192.168.1.10	BA:DB:AD:BA:DB:AD

192.168.1.100	BA:DB:AD:BA:DB:AD

192.168.1.100	08:00:46:07:04:A3
192.168.1.10	00:01:03:1D:98:B8

ARP Spoofing

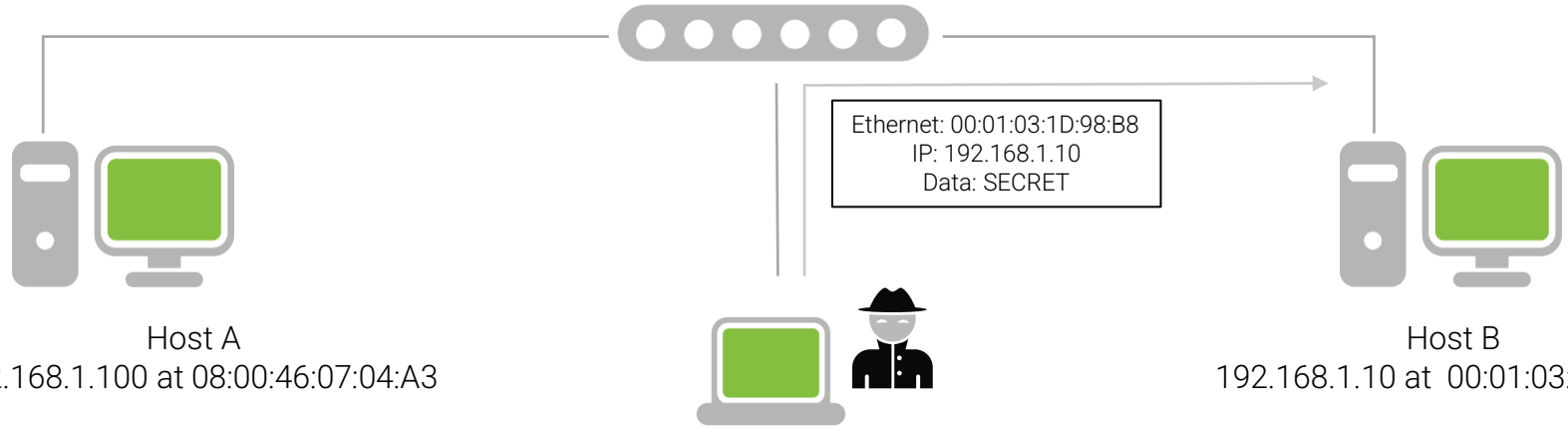


192.168.1.10	BA:DB:AD:BA:DB:AD

192.168.1.100	BA:DB:AD:BA:DB:AD

192.168.1.100	08:00:46:07:04:A3
192.168.1.10	00:01:03:1D:98:B8

ARP Spoofing



192.168.1.10	BA:DB:AD:BA:DB:AD

192.168.1.100	08:00:46:07:04:A3
192.168.1.10	00:01:03:1D:98:B8

192.168.1.100	BA:DB:AD:BA:DB:AD

ARP Spoofing

- Legitimate ARP replies might restore the ARP cache to the correct value
- Most ARP-spoofing tool repeatedly send spoofed ARP replies to keep the ARP cache in the desired state

Ettercap

- Tool for performing man-in-middle attacks in LANs
- Provides support for ARP spoofing attacks
  - Automatically forwards packets that are not directed to the IP of the host running the attack
- Provides support for the interception of SSH1 and SSL connections
- Supports the collection of passwords for a number of protocols

ARP Defenses

- Static ARP entries
  - The ARP cache can be configured to ignore dynamic updates
  - Difficult to manage in large deployments
    - Could be used for a subset of critical addresses (e.g., DNS servers, gateways)
- Cache poisoning resistance
  - Ignore unsolicited ARP replies (still vulnerable to hijacking)
  - Update on timeout (limited usefulness)
- Monitor changes (e.g., arpwatch)
  - Listen for ARP packets on a local Ethernet interface
  - Keep track for Ethernet/IP address pairs
  - Report suspicious activity and changes in mapping

Detecting Sniffers on Your Network

- Sniffers are typically passive programs
- They put the network interface in promiscuous mode and listen for traffic
- They can be detected by programs that provide information on the status of a network interface (e.g., ifconfig)
  - `# ifconfig eth0`

```
eth0      Link encap:Ethernet  HWaddr 00:10:4B:E2:F6:4C
          inet addr:192.168.1.20  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:1016 errors:0 dropped:0 overruns:0 frame:0
          TX packets:209 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
```

- A kernel-level rootkit can easily hide the presence of a sniffer

Detecting Sniffers on Your Network

- Suspicious ARP activity
  - ARP cache poisoning attacks are noisy
  - Tools like arpwatch and XArp detect a variety of ARP attacks
- Suspicious DNS lookups
  - Sniffer attempts to resolve names associated with IP addresses (may be part of normal operation)
  - Trap: generate connection from fake IP address not in local network and detect attempt to resolve name
- Latency
  - Assumption: Since the NIC is in promiscuous mode EVERY packet is processed
  - Use ping to analyze response time of host A
  - Generate huge amount of traffic to other hosts and analyze response time of host A

Detecting Sniffers on Your Network

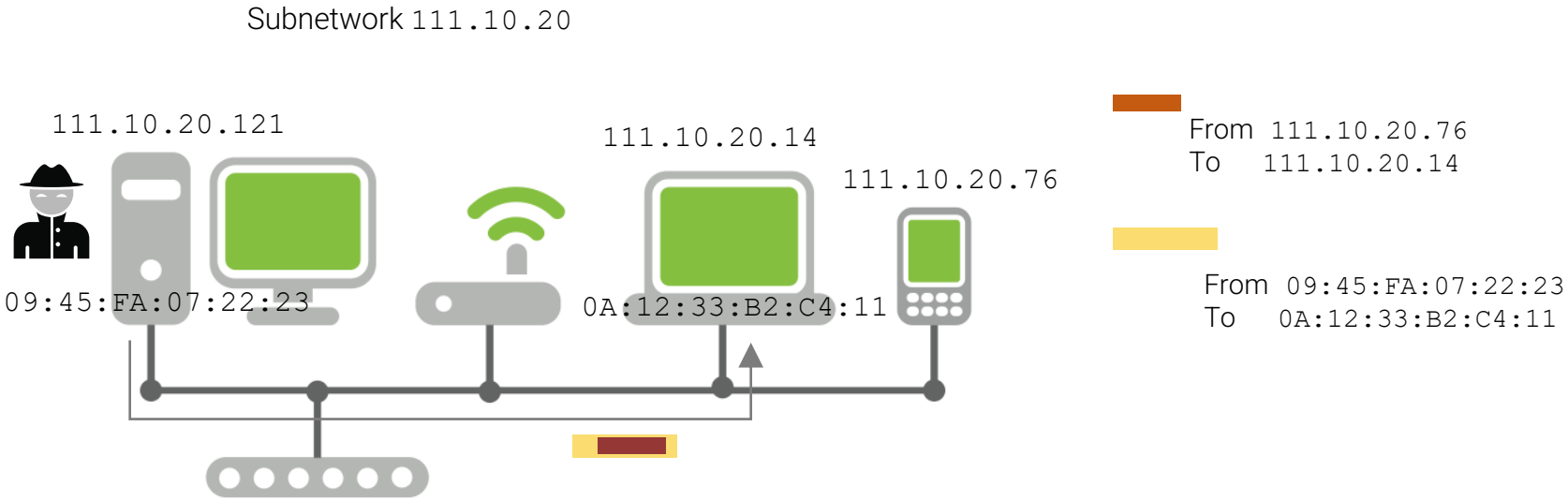
- Kernel behavior
  - Linux
    - When in promiscuous mode, some kernels will accept a packet that has the wrong Ethernet address but the right destination IP address
    - If sending an ICMP request to a host using the wrong Ethernet address but the correct IP address causes an ICMP reply, the host is sniffing the network
- AntiSniff tool (written in 2000!)
  - Covers some of the techniques above
  - Uses TCP SYN and TCP handshake forged traffic to overload sniffer when testing latency

Controlling Network Access

- Sniffing and hijacking attacks (e.g., ARP attacks) require physical access
- It is important to control who can access your network
- IEEE 802.1X is port-based access control protocol
  - A “supplicant” (e.g., a laptop) connects to an “authenticator” (e.g., a switch)
  - The “supplicant” has minimal traffic access until it presents the right credentials (through the authenticator) to an authentication server
    - Protocol based on the Extensible Authentication Protocol (EAP) over LAN (EAPOL)
  - Once the right credentials are provided network access will be granted

IP Spoofing

- In an IP spoofing attack a host impersonates another host by sending a datagram with the address of the impersonated host as the source address



Why IP Spoofing?

- IP spoofing is used to impersonate sources of security-critical information (e.g., a DNS server)
- IP spoofing is used to exploit address-based authentication in higher-level protocols
- Many tools available
  - Protocol-specific spoofers (DNS spoofers, NFS spoofers, etc)
  - Generic IP spoofing tools (e.g., hping)

Libnet

- Provides a platform-independent library of functions to build (and inject) arbitrary packets
- Allows to write Ethernet spoofed frames
- Steps in building a packet
  1. Memory Initialization (allocates memory for packets)
  2. Network Initialization (initializes the network interface)
  3. Packet Construction (fill in the different protocol headers/payloads)
  4. Packet Checksums (compute the necessary checksums - some of them could be automatically computed by the kernel)
  5. Packet Injection (send the packet on the wire)

Scapy

- Python library for the manipulation of packets
- Allows for the fast prototyping of network attack tools
- Provides support for sniffing and spoofing
- Slower than libpcap/libnet but easier to use
- For example, to send a spoofed ICMP packet:
  > send(IP(src="128.111.40.59", dst="128.111.40.54")/ICMP())

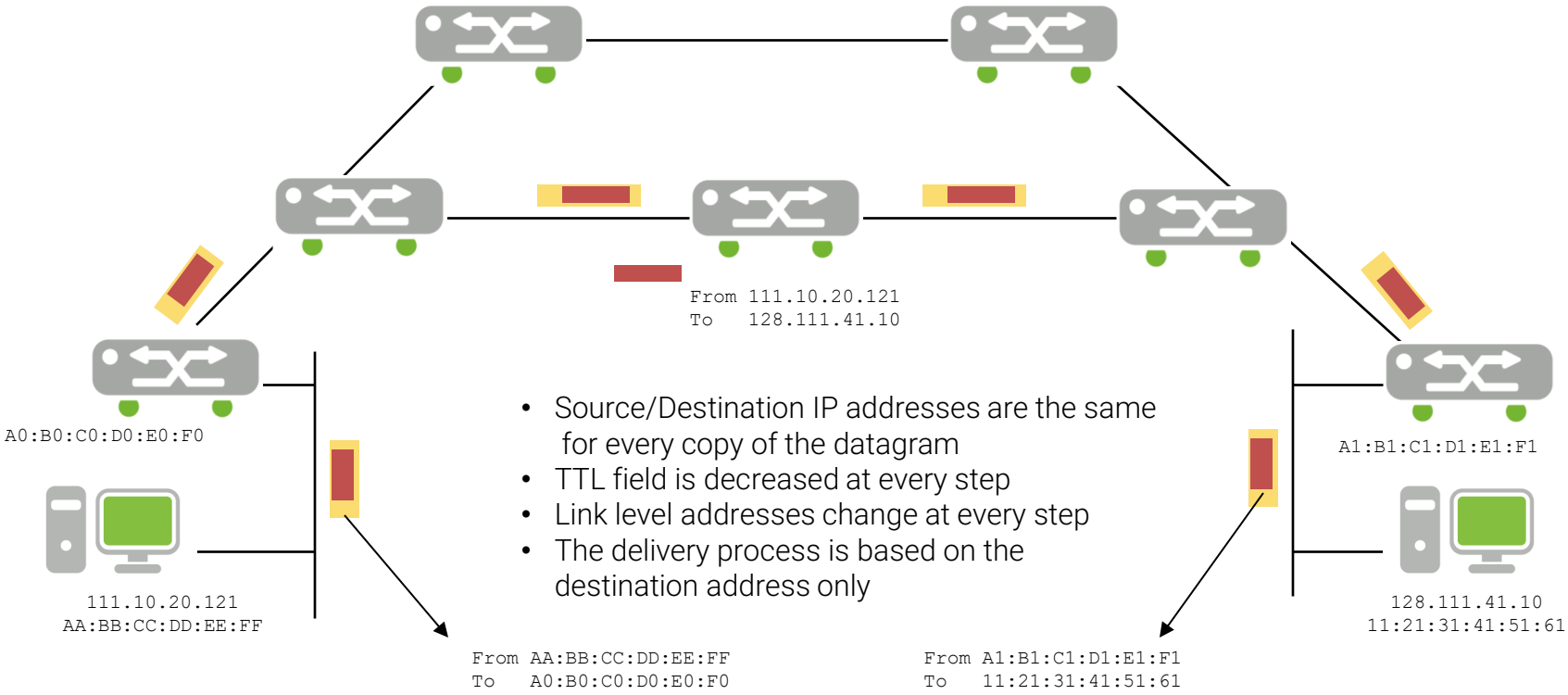
Hijacking

- Sniffing and spoofing are the basis for hijacking
- The attacker sniffs the network, waiting for a client request
- Races against legitimate host when producing a reply
- We will see ARP-, UDP-, and TCP-based variations of this attack

Routing: Indirect Delivery

- If two hosts are in different physical networks the IP datagram is encapsulated in a lower level protocol and delivered to the directly connected gateway
- The gateway decides which is the next step in the delivery process
- This step is repeated until a gateway that is in the same physical subnetwork of the destination host is reached
- Then direct delivery is used

Routing



- Source/Destination IP addresses are the same for every copy of the datagram
- TTL field is decreased at every step
- Link level addresses change at every step
- The delivery process is based on the destination address only

Types of Routing

- Hop-by-hop routing
  - The delivery route is determined by the gateways that participate in the delivery process
- Source routing
  - The originator of a datagram determines the route to follow independently before sending the datagram (IP source routing option)

Attacks Using Source Routing

- The IP source routing option can be used to specify the route to be used in the delivery process, independent of the “normal” delivery mechanisms
- Using source routing a host can force the traffic through specific routes that allow one to access the traffic (to perform sniffing or man-in-the-middle attacks)
- If the reverse route is used to reply to traffic, a host can easily impersonate another host that has some kind of privileged relationship with the host that is the destination of the datagram (a trust relationship)
- For these reasons, source routing is not honored by most routers

Hop-by-hop Routing: The Routing Table

- The information about delivery is maintained in the routing table

```
% route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Iface
192.168.1.24    0.0.0.0         255.255.255.255 UH    eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     lo
0.0.0.0         192.168.1.1     0.0.0.0         UG    eth0
```

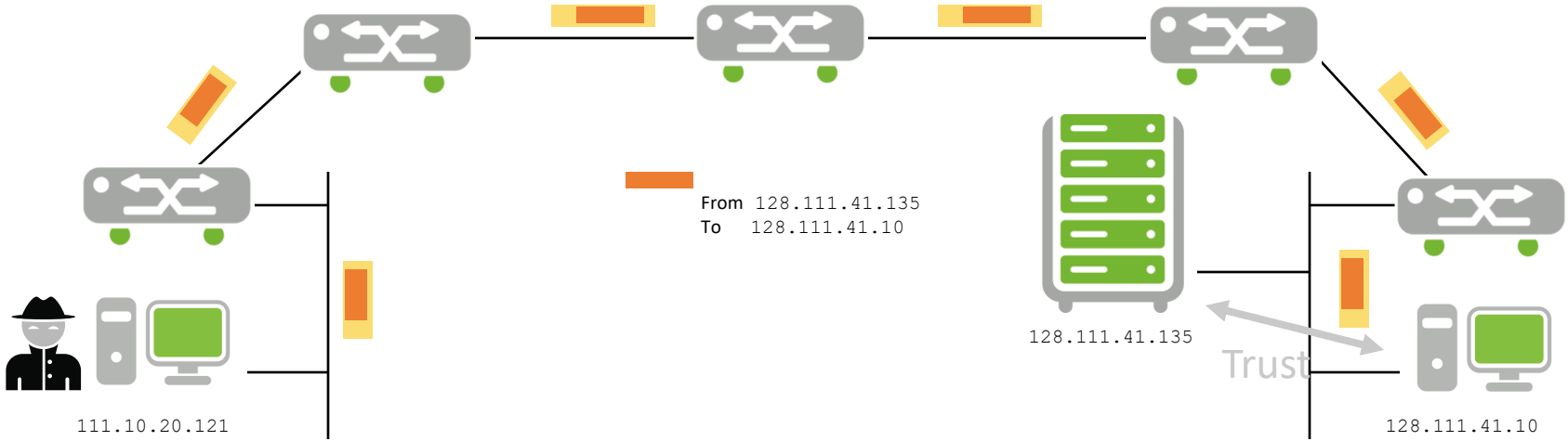
- Flags
  - U: the route is up
  - G: the destination is a gateway
  - H: the route is to a host (if not set, the route is to a network)
  - D: the route was created by a redirect message
  - M: the route was modified by a redirect message

Routing Mechanism

- Search for a matching host address
- Search for a matching network address
- Search for a default entry
- If a match is not found a message of “host unreachable” or “network unreachable” is returned (by the kernel or by a remote gateway using ICMP)
- Routing tables can be set
  - Statically (at startup, or by using the “route” command)
  - Dynamically (using routing protocols)

Blind IP Spoofing

- A host (111.10.20.121) sends an IP datagram with the address of some other host as the source address (128.111.41.135)
- The attacked host replies to the impersonated host
- Usually the attacker does not have access to the reply traffic



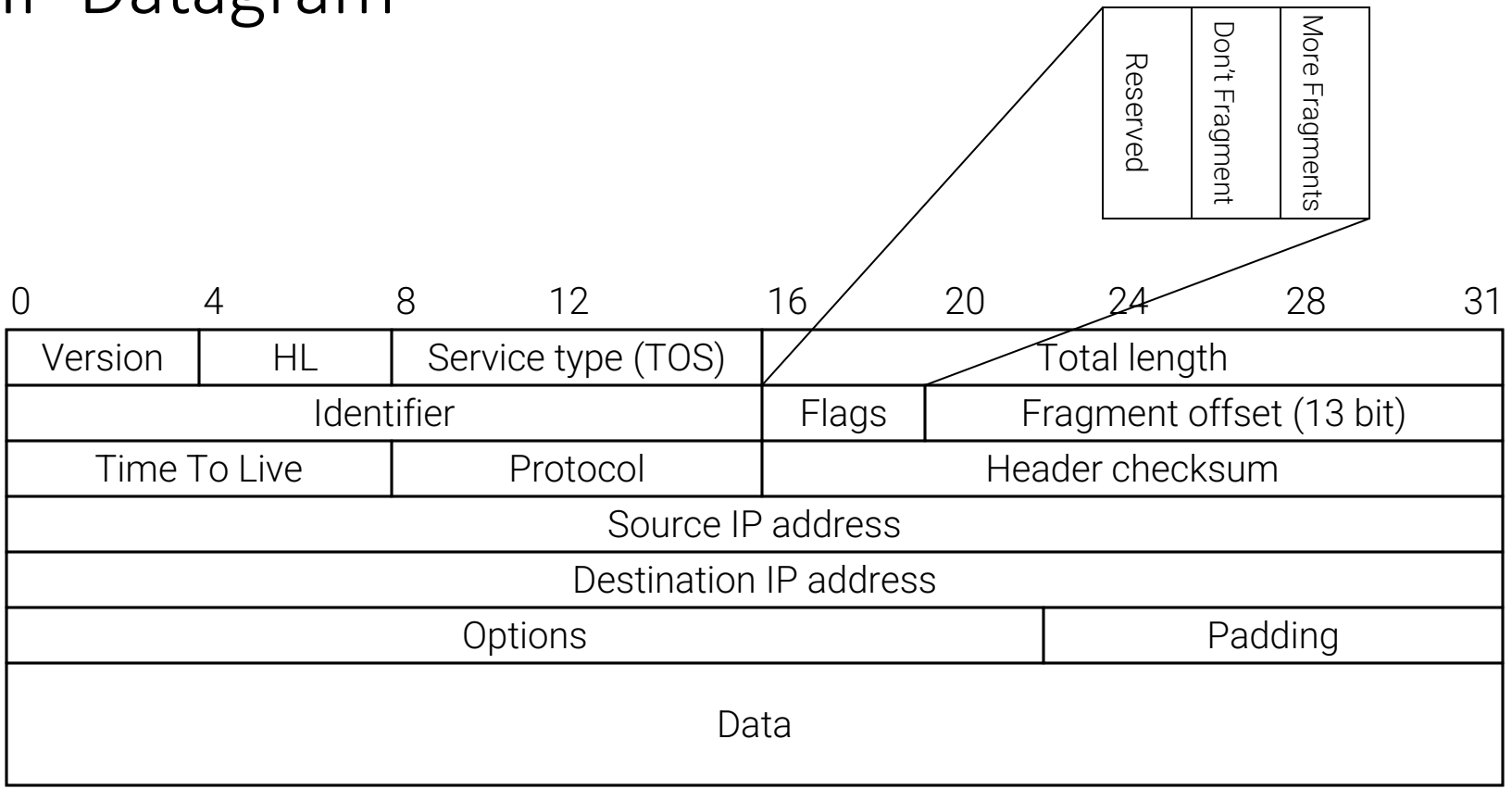
Man-in-the-middle Attacks

- An attacker that has control of a gateway used in the delivery process can
  - Sniff the traffic
  - Intercept/block traffic
  - Modify traffic
- Perform a full man-in-the-middle attack

Fragmentation

- When a datagram is encapsulated in lower level protocols (e.g., Ethernet) it may be necessary to split the datagram in smaller portions
- This happens when the datagram size is bigger than the data link layer MTU (Maximum Transmission Unit)
- Fragmentation can be performed at the source host or at an intermediate step in datagram delivery
- If the datagram has the “do not fragment” flag set, an ICMP error message is sent back to the originator

IP Datagram


Reserved
Don't Fragment
More Fragments
0
4
8
12
16
20
24
28
31
Version
HL
Service type (TOS)
Total length
Identifier
Flags
Fragment offset (13 bit)
Time To Live
Protocol
Header checksum
Source IP address
Destination IP address
Options
Padding
Data

Fragmentation

- If the datagram can be fragmented:
  - The header is copied in each fragment
    - In particular, the “datagram id” is copied in each fragment
  - If the “more fragments” flag is set with the exception of the last fragment
  - The “fragmentation offset” field contains the position of the fragment with respect to the original datagram expressed in 8-byte units
  - The “total length field” is changed to match the size of the fragment
- Each fragment is then delivered as a separate datagram
- If one fragment is lost the entire datagram is discarded after a timeout

Fragmentation

```
09:52:32.150083 < 128.111.48.69 > 128.111.48.70: (frag 36542:928@31080)
09:52:32.151231 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@29600+)
09:52:32.152483 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@28120+)
09:52:32.153703 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@26640+)
09:52:32.154896 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@25160+)
09:52:32.156208 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@23680+)
09:52:32.157401 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@22200+)
09:52:32.158632 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@20720+)
09:52:32.160441 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@17760+)
09:52:32.161640 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@16280+)
09:52:32.162951 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@14800+)
09:52:32.164133 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@13320+)
09:52:32.165379 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@11840+)
09:52:32.166559 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@10360+)
09:52:32.167797 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@8880+)
09:52:32.169107 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@7400+)
09:52:32.170884 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@4440+)
09:52:32.172114 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@2960+)
09:52:32.173296 < 128.111.48.69 > 128.111.48.70: (frag 36542:1480@1480+)
09:52:32.174527 < 128.111.48.69 > 128.111.48.70: icmp: echo request (frag 36542:1480@0+)
```

Fragmentation Attacks: Ping of Death

- The offset of the last segment is such that the total size of the reassembled datagram is bigger than the maximum allowed size
  - A kernel static buffer is overflowed, causing a kernel panic

Ping of Death

```
23:01:06.266646 < 128.111.48.69 > 128.111.48.70: icmp: echo request (frag 4321:1480@0+)
23:01:06.421261 < 128.111.48.69 > 128.111.48.70: (frag 4321:1480@1480+)
23:01:06.575953 < 128.111.48.69 > 128.111.48.70: (frag 4321:1480@2960+)
23:01:06.730065 < 128.111.48.69 > 128.111.48.70: (frag 4321:1480@4440+)
23:01:06.884625 < 128.111.48.69 > 128.111.48.70: (frag 4321:1480@5920+)
23:01:07.038801 < 128.111.48.69 > 128.111.48.70: (frag 4321:1480@7400+)
23:01:07.193403 < 128.111.48.69 > 128.111.48.70: (frag 4321:1480@8880+)
23:01:07.348185 < 128.111.48.69 > 128.111.48.70: (frag 4321:1480@10360+)
23:01:07.502326 < 128.111.48.69 > 128.111.48.70: (frag 4321:1480@11840+)
[...]
23:01:12.451121 < 128.111.48.69 > 128.111.48.70: (frag 4321:1480@59200+)
23:01:12.605235 < 128.111.48.69 > 128.111.48.70: (frag 4321:1480@60680+)
23:01:12.759927 < 128.111.48.69 > 128.111.48.70: (frag 4321:1480@62160+)
23:01:12.917811 < 128.111.48.69 > 128.111.48.70: (frag 4321:1480@63640+)
23:01:13.090936 < 128.111.48.69 > 128.111.48.70: (frag 4321:398@65120)
```

Total 65120 + 398 = 65518 + 20 bytes of header = 65538 > 65535!

Fragmentation Attacks: Evasion and Denial-Of-Service

- Firewalls and intrusion detection systems analyze incoming datagrams using the information contained in both the datagram header and the datagram payload (TCP ports, UDP ports, SYN and ACK flags in the TCP header)
- An attacker may use fragmentation to avoid filtering
  - Some firewalls make a decision on the first fragment and let the other fragments through by keeping track of the datagram ID
  - The first fragment of a TCP-over-IP may contain only 8 bytes (source and destination ports for both UDP and TCP)
    - Setup flags (SYN/ACK) can be “postponed” so that incoming SYNs can go through
    - Setup flags can be overwritten by using overlapping fragments
    - In some cases, even the original src/dst port can be rewritten

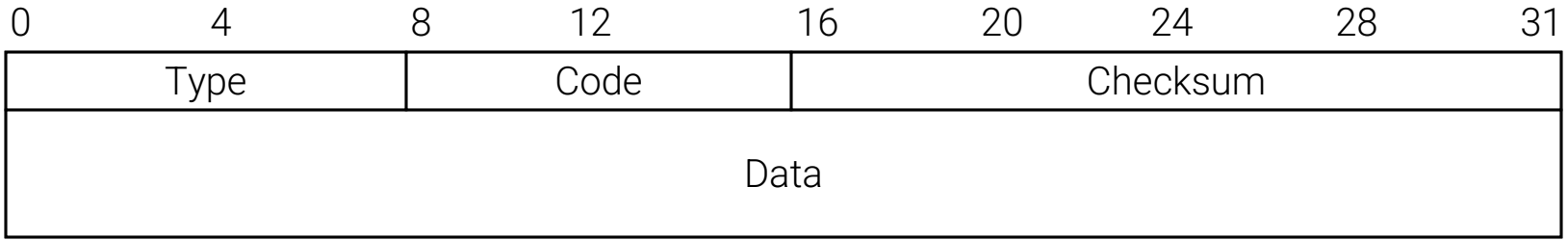
Fragmentation Attacks: Evasion and Denial-Of-Service

- An attacker may use fragmentation to avoid detection
  - Some network-based IDSs do not reassemble datagrams
  - An IDS may obtain a reassembled datagram different from the one obtained at the receiving host
- An attacker may use fragmentation to build a DOS attack
  - An attacker may exploit problems in the reassembling code (teardrop, ping of death)
  - If firewalls and IDSs reassemble a datagram before analyzing it, it is possible to force the system to use a large amount of memory

Internet Control Message Protocol

- ICMP is used to exchange control/error messages about the delivery of IP datagrams
- ICMP messages are encapsulated inside IP datagrams
- ICMP messages can be:
  - Requests
  - Responses
  - Error messages
    - An ICMP error message includes the header and a portion of the payload (usually the first 8 bytes) of the offending IP datagram

Message Format



ICMP Messages

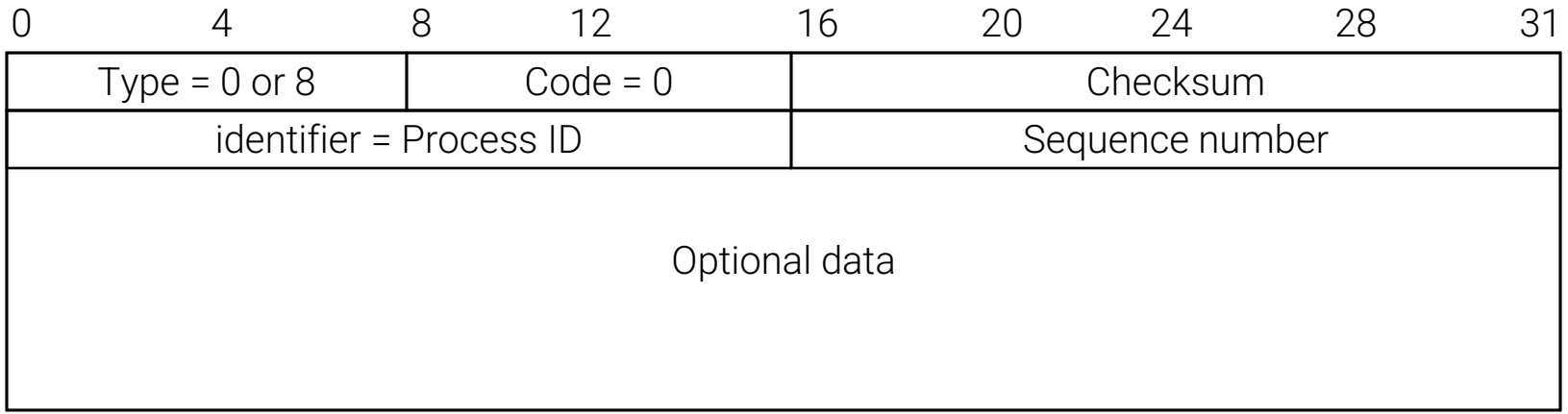
- Address mask request/reply: used by diskless systems to obtain the network mask at boot time
- Timestamp request/reply: used to synchronize clocks
- Source quench: used to inform about traffic overloads
- Parameter problem: used to inform about errors in the IP datagram fields

ICMP Messages

- Echo request/reply: used to test connectivity (ping)
- Time exceeded: used to report expired datagrams (TTL = 0)
- Redirect: used to inform hosts about better routes (gateways)
- Destination unreachable: used to inform a host of the impossibility to deliver traffic to a specific destination

ICMP Echo Request/Reply

- Used by the ping program



Ping

```
# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) from 192.168.1.100 : 56(84) bytes of
data.
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=1.049 msec
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=660 usec
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=597 usec
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=548 usec
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=601 usec
64 bytes from 192.168.1.1: icmp_seq=5 ttl=64 time=592 usec
64 bytes from 192.168.1.1: icmp_seq=6 ttl=64 time=547 usec

--- 192.168.1.1 ping statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.547/0.656/1.049/0.165 ms
```

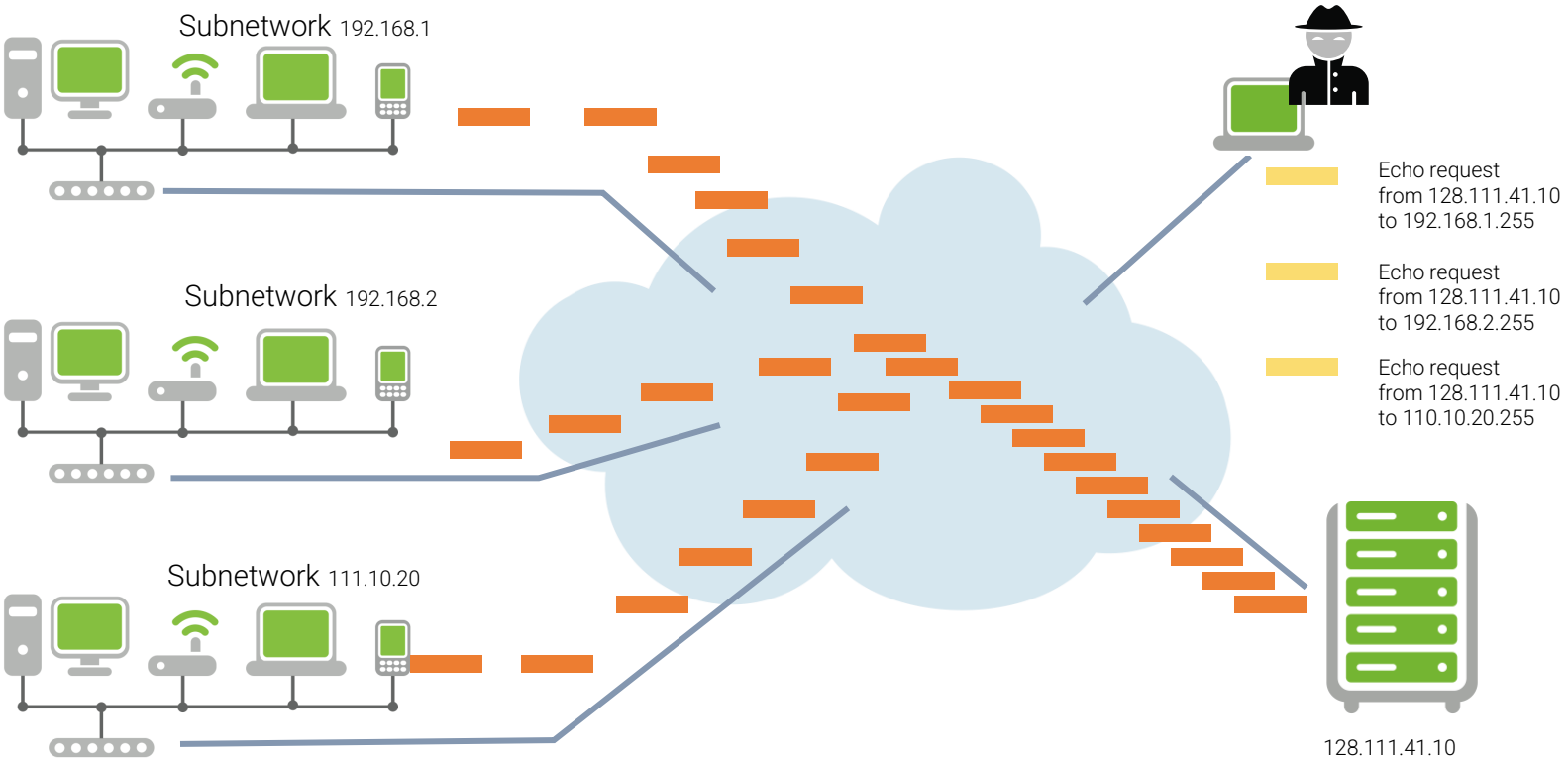
ICMP Echo Attacks

- ICMP Echo Request messages can be used to map the hosts of a network (pingscan or ipsweep)
  - ICMP echo datagrams are sent to all the hosts in a subnetwork
  - The attacker collects the replies and determines which hosts are actually alive

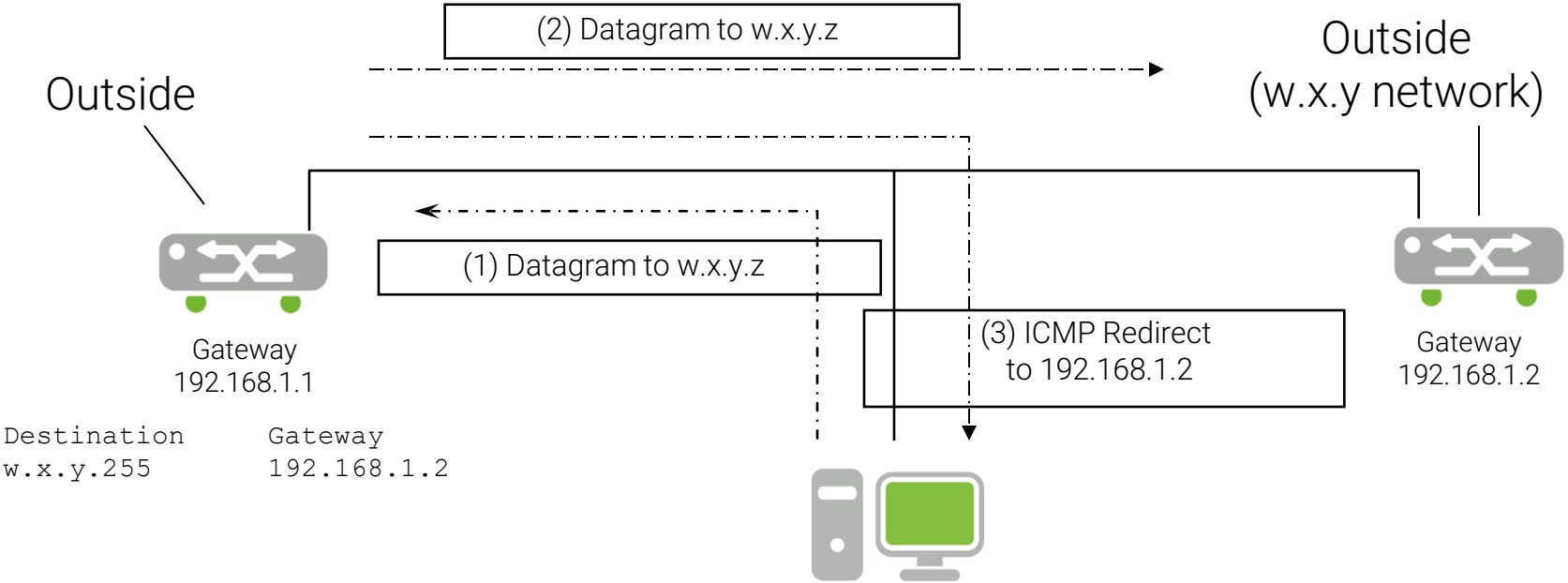
```
Starting nmap by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Host cisco-sales.ns.com (192.168.31.11) appears to be up.
Host sales1.ns.com (192.168.31.19) appears to be up.
Host sales4.ns.com (192.168.31.22) appears to be up.
Host sales2.ns.com (192.168.31.43) appears to be up.
Host sales3.ns.com (192.168.31.181) appears to be up.

Nmap run completed -- 256 IP addresses (5 hosts up) scanned in 1 second
```

- ICMP Echo Request can be used to perform a denial of service attack (smurf)

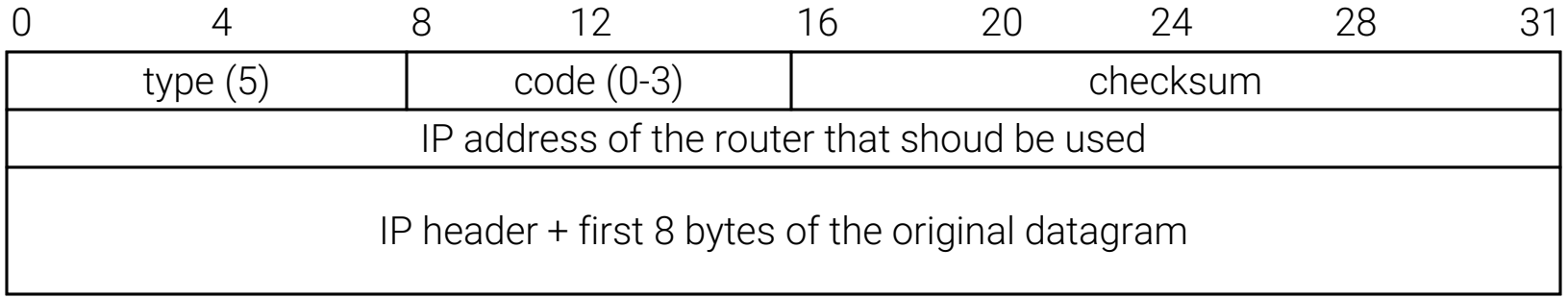
Smurf


Subnetwork 192.168.1
Subnetwork 192.168.2
Subnetwork 111.10.20
Echo request
from 128.111.41.10
to 192.168.1.255
Echo request
from 128.111.41.10
to 192.168.2.255
Echo request
from 128.111.41.10
to 110.10.20.255
128.111.41.10

ICMP Redirect



ICMP Redirect



- Code
  - 0: redirect for networks (deprecated and usually treated as host)
  - 1: redirect for hosts

ICMP Redirect

- A host that receives an ICMP redirect message performs the following checks:
  - The new router must be on a directly connected network
  - The redirect must be from the current router for that destination
  - The redirect cannot tell the host to use itself as the router
  - The route that is being modified must be an indirect route

ICMP Redirect Attacks

- ICMP redirect messages can be used to re-route traffic on specific routes or to a specific host that is not a router at all
- The attack is performed sending to a host a spoofed ICMP redirect message that appears to come from the host's default gateway
- The attack can be used to
  - Hijack traffic
  - Perform a denial-of-service attack

ICMP Redirect Attacks

```
# arp -n
  Address                  HWtype  HWaddress
  192.168.1.1              ether   00:20:78:CA:7E:AE
  192.168.1.10             ether   00:01:03:1D:98:B8
  192.168.1.100            ether   08:00:46:07:04:A3

C:\WINDOWS>route PRINT

Active Routes:

  Network Address          Netmask  Gateway Address        Interface  Metric
          0.0.0.0          0.0.0.0      192.168.1.1     192.168.1.10       1
        127.0.0.0        255.0.0.0        127.0.0.1        127.0.0.1       1
      192.168.1.0    255.255.255.0     192.168.1.10     192.168.1.10       1
     192.168.1.10  255.255.255.255        127.0.0.1        127.0.0.1       1
    192.168.1.255  255.255.255.255     192.168.1.10     192.168.1.10       1

# tcpdump -n

8:0:46:7:4:a3 0:1:3:1d:98:b8 0800 70: 192.168.1.1 > 192.168.1.10:
  icmp: redirect 128.111.48.69 to host 192.168.1.100
```

ICMP Redirect Attacks

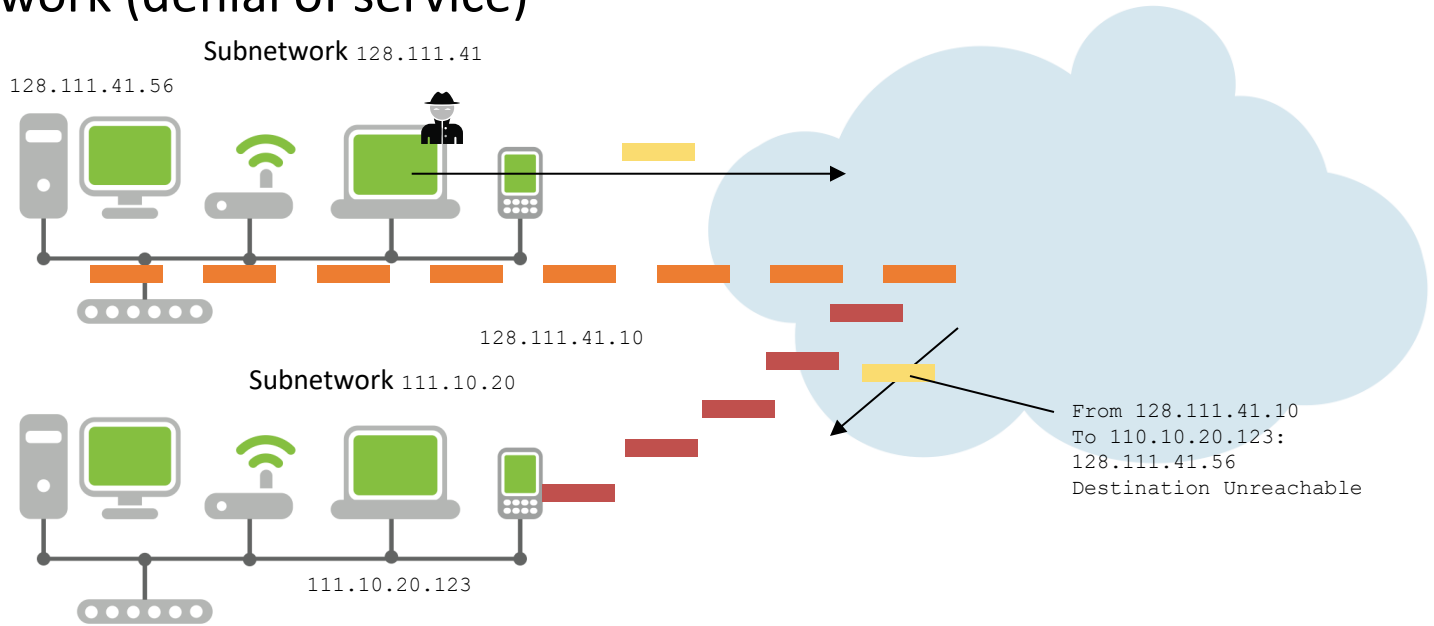
```
C:\WINDOWS>route PRINT
Active Routes:
  Network Address          Netmask  Gateway Address        Interface  Metric
          0.0.0.0          0.0.0.0      192.168.1.1     192.168.1.10       1
        127.0.0.0        255.0.0.0        127.0.0.1        127.0.0.1       1
    128.111.48.69  255.255.255.255    192.168.1.100     192.168.1.10       1
      192.168.1.0    255.255.255.0     192.168.1.10     192.168.1.10       1
     192.168.1.10  255.255.255.255        127.0.0.1        127.0.0.1       1
    192.168.1.255  255.255.255.255     192.168.1.10     192.168.1.10       1


C:\WINDOWS>ping 128.111.48.69
0:1:3:1d:98:b8 8:0:46:7:4:a3 0800 74: 192.168.1.10 > 128.111.48.69:
   icmp: echo request
0:1:3:1d:98:b8 8:0:46:7:4:a3 0800 74: 192.168.1.10 > 128.111.48.69:
   icmp: echo request
...
```

ICMP Destination Unreachable

- ICMP message used by gateways to state that the datagram cannot be delivered
- Many subtypes
  - Network unreachable
  - Host unreachable
  - Protocol unreachable
  - Port unreachable
  - Fragmentation needed but don’t fragment bit set
  - Destination host unknown
  - Destination network unknown
  - …

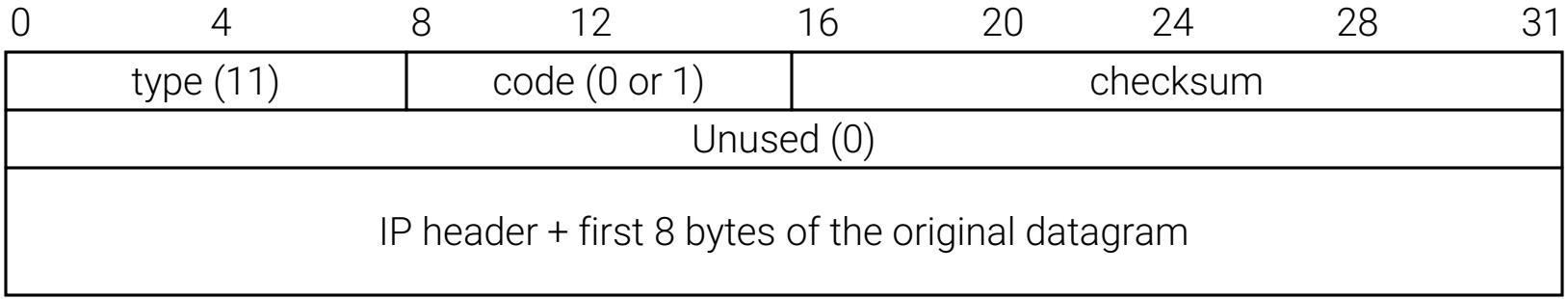
Destination Unreachable Attacks

- Forged destination unreachable messages can cut out nodes from the network (denial of service)



ICMP Time Exceeded

- Used when
  - TTL becomes zero (code = 0)
  - The reassembling of a fragmented datagram times out (code =1)



Traceroute

- ICMP Time Exceeded messages are used by the traceroute program to determine the path used to deliver a datagram
- A series of IP datagrams are sent to the destination node
- Each datagram has an increasing TTL field (starting at 1)
- From the ICMP Time exceeded messages returned by the intermediate gateways it is possible to reconstruct the route from the source to the destination
- Note: traceroute allows one to specify loose source routing (-g option)
- Useful for network analysis, topology mapping

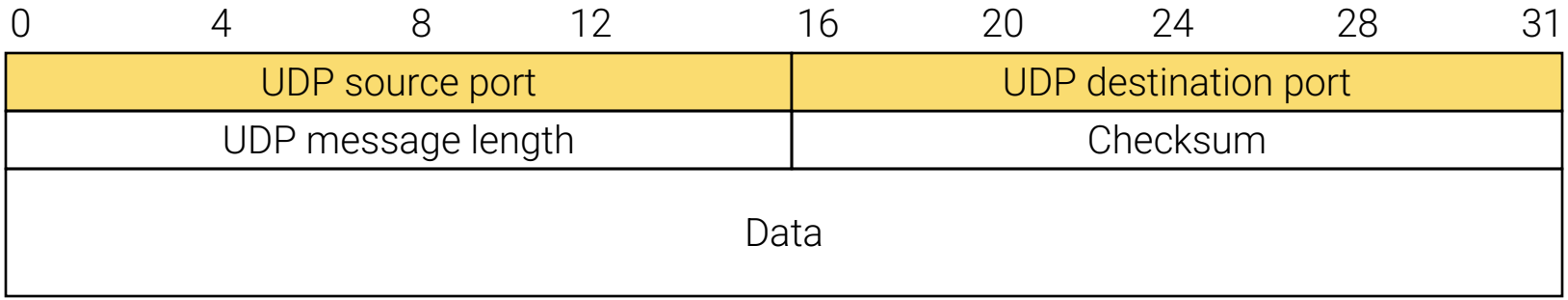
Traceroute

```
traceroute to pos4-1-155M.cr2.SNV.gblx.net (206.132.150.233), 30 hops max, 38 byte packets 1

csworld48 (128.111.48.2)  1.077 ms  0.827 ms  1.051 ms 2
engr-gw-lo.ucsb.edu (128.111.51.1)  1.479 ms  0.855 ms  1.222 ms 3
border1.ucsb.edu (128.111.1.83)  1.224 ms  1.375 ms  1.222 ms 4
gsr-g-1-0.commserv.ucsb.edu (128.111.252.150)  1.357 ms  1.383 ms  1.642 ms 5
USC--ucsb.ATM.calren2.net (198.32.248.73)  3.876 ms  4.493 ms  3.913 ms 6
ISI--USC.POS.calren2.net (198.32.248.26)  4.401 ms  4.533 ms  4.261 ms 7
UCLA--ISI.POS.calren2.net (198.32.248.30)  4.933 ms  4.897 ms  5.002 ms 8
UCLA-7507--UCLA.POS.calren2.net (198.32.248.118)  5.429 ms  5.530 ms  5.384 ms 9
corerouter2-serial6-0-0.Bloomington.cw.net (166.63.131.129)  8.562 ms  8.244 ms  7.857 ms 10
corerouter1.SanFrancisco.cw.net (204.70.9.131)  17.563 ms  17.861 ms  17.941 ms11
bordercore1.SanFrancisco.cw.net (166.48.12.1)  18.108 ms  18.269 ms  17.945 ms12
frontier-comm.SanFrancisco.cw.net (166.48.13.242)  19.164 ms  18.749 ms  20.472 ms 13
pos4-1-155M.cr2.SNV.gblx.net (206.132.150.233)  19.664 ms  18.666 ms  18.503 ms 14
```

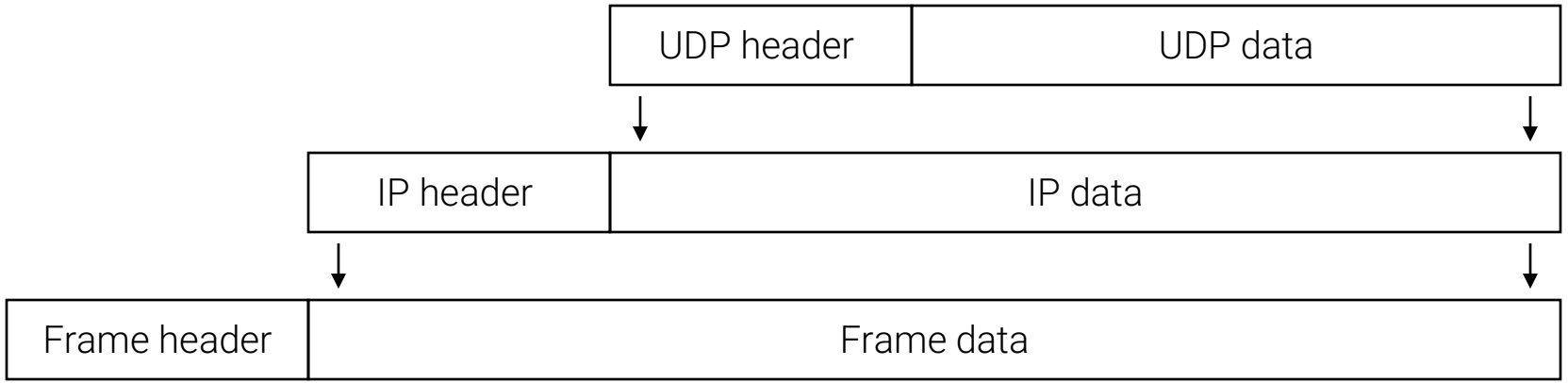
User Datagram Protocol (UDP)

- The UDP protocol relies on IP to provide a connectionless, unreliable, best-effort datagram delivery service (delivery, integrity, non-duplication, ordering, and bandwidth is not guaranteed)
- Introduces the port abstraction that allows one to address different message destinations for the same IP address
- Often used for multimedia (more efficient than TCP) and for services based on request/reply schema (DNS, RPC)

UDP Message

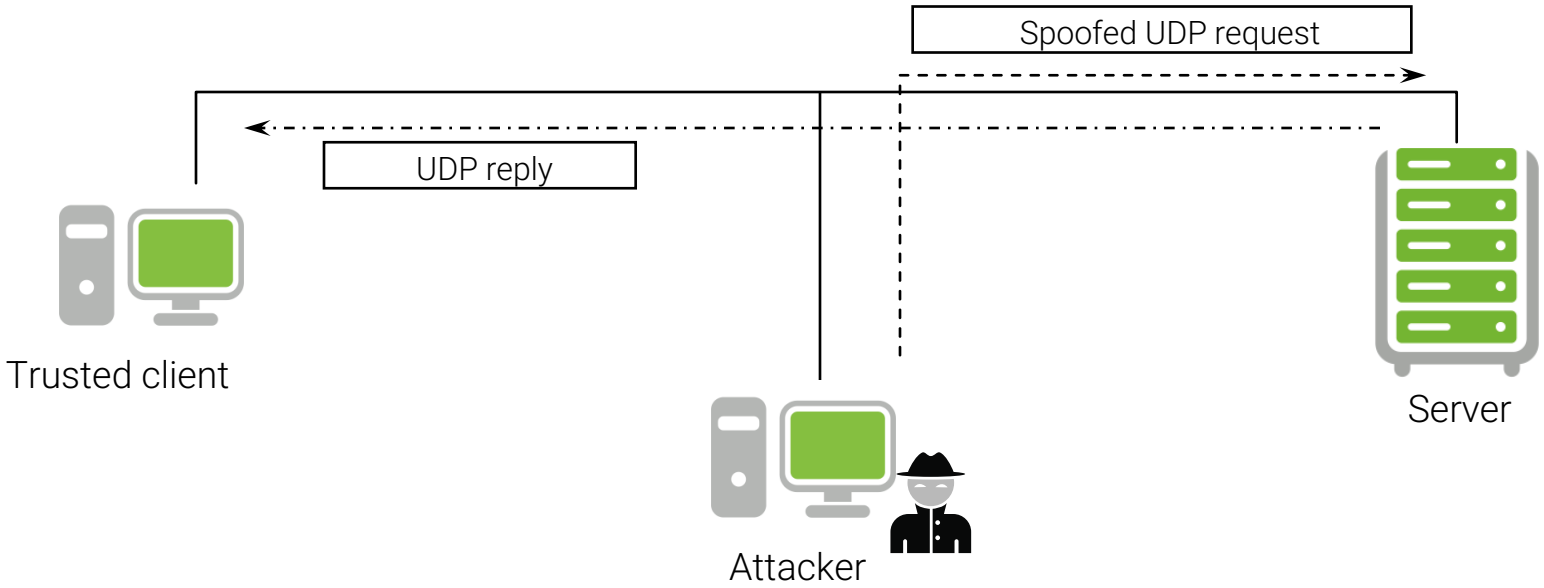

0
4
8
12
16
20
24
28
31
UDP source port
UDP destination port
UDP message length
Checksum
Data

UDP Encapsulation



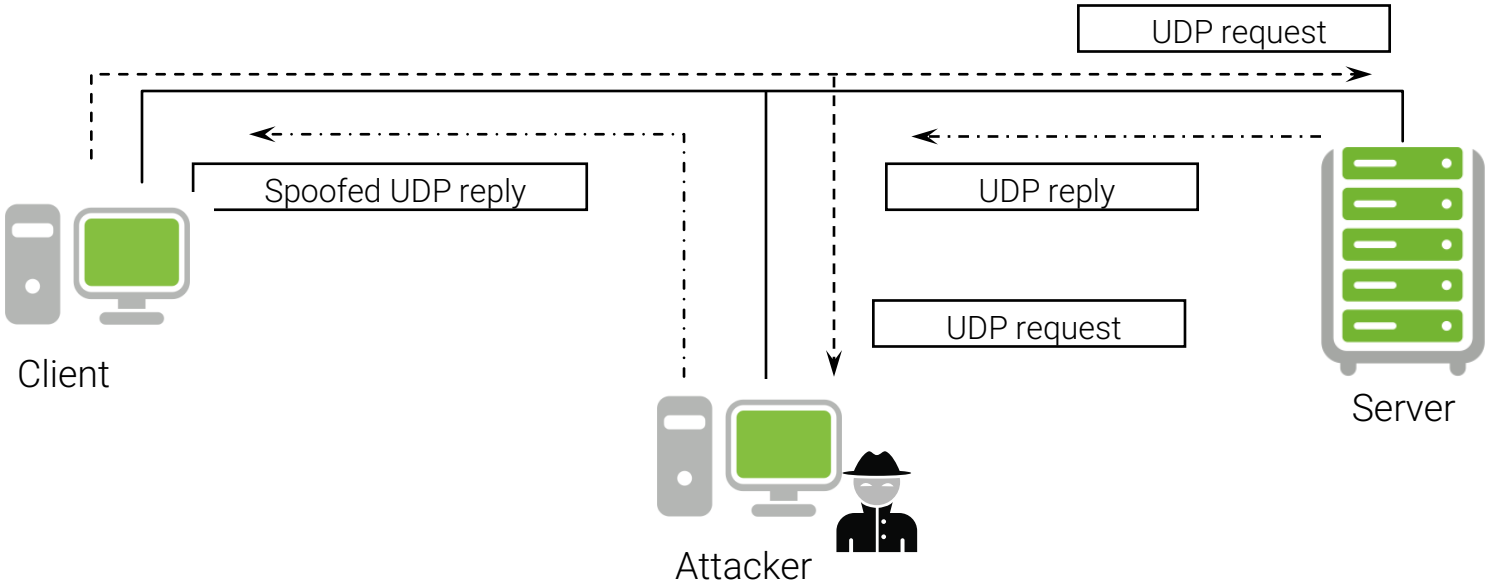
UDP Spoofing

- Basically IP spoofing



UDP Hijacking

- Variation of the UDP spoofing attack



UDP Portscan

- Used to determine which UDP services are available
- A zero-length UDP packet is sent to each port
- If an ICMP error message “port unreachable” is received the service is assumed to be unavailable
- Many TCP/IP stack implementations (not Windows’!) implement a limit on the error message rate, therefore this type of scan can be slow (e.g., Linux’ limit is 80 messages every 4 seconds)

UDP Portscan

```
% nmap -sU 192.168.1.10

Starting Nmap
Interesting ports on  (192.168.1.10):
(The 1445 ports scanned but not shown below are in state: closed)
Port       State       Service
137/udp    open        netbios-ns
138/udp    open        netbios-dgm

Nmap run completed -- 1 IP address (1 host up) scanned in 4 seconds
```

UDP Portscan

```
19:37:31.305674 192.168.1.100.41481 > 192.168.1.10.138: udp 0 (ttl 46, id 61284)
19:37:31.305706 192.168.1.100.41481 > 192.168.1.10.134: udp 0 (ttl 46, id 31166)
19:37:31.305730 192.168.1.100.41481 > 192.168.1.10.137: udp 0 (ttl 46, id 31406)
19:37:31.305734 192.168.1.100.41481 > 192.168.1.10.140: udp 0 (ttl 46, id 50734)
19:37:31.305770 192.168.1.100.41481 > 192.168.1.10.131: udp 0 (ttl 46, id 33361)
19:37:31.305775 192.168.1.100.41481 > 192.168.1.10.132: udp 0 (ttl 46, id 14242)
19:37:31.305804 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 134 unreachable
19:37:31.305809 192.168.1.100.41481 > 192.168.1.10.135: udp 0 (ttl 46, id 17622)
19:37:31.305815 192.168.1.100.41481 > 192.168.1.10.139: udp 0 (ttl 46, id 52452)
19:37:31.305871 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 140 unreachable
19:37:31.305875 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 131 unreachable
19:37:31.305881 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 132 unreachable
19:37:31.305887 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 135 unreachable
19:37:31.305892 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 139 unreachable
19:37:31.305927 192.168.1.100.41481 > 192.168.1.10.133: udp 0 (ttl 46, id 38693)
19:37:31.305932 192.168.1.100.41481 > 192.168.1.10.130: udp 0 (ttl 46, id 60943)
19:37:31.305974 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 133 unreachable
19:37:31.305979 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 130 unreachable
19:37:31.617611 192.168.1.100.41482 > 192.168.1.10.138: udp 0 (ttl 46, id 21936)
19:37:31.617641 192.168.1.100.41482 > 192.168.1.10.137: udp 0 (ttl 46, id 17647)
19:37:31.617663 192.168.1.100.41481 > 192.168.1.10.136: udp 0 (ttl 46, id 55)
19:37:31.617737 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 136 unreachable
```

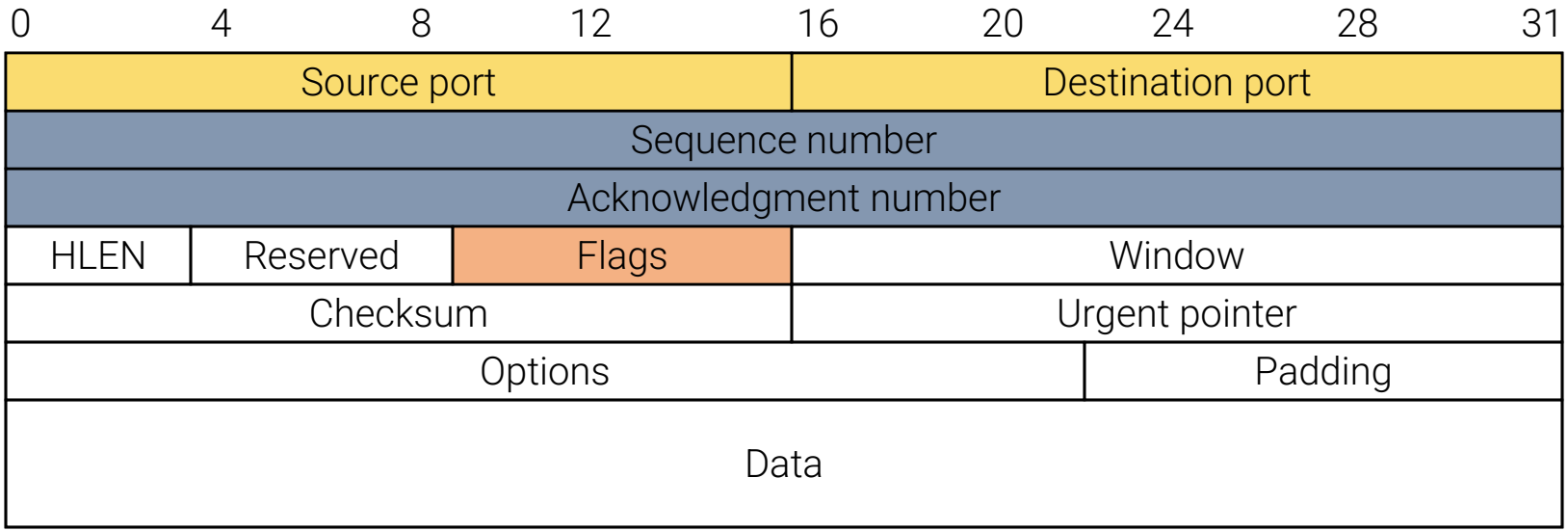
UDP and DOS

- UDP can be used as a basis for powerful DDOS attacks, as the lack of a session concept prevents the receiver from meaningfully block the packets in most cases
- Example: The Fraggle attack, similarly to Smurf, used net-directed broadcast

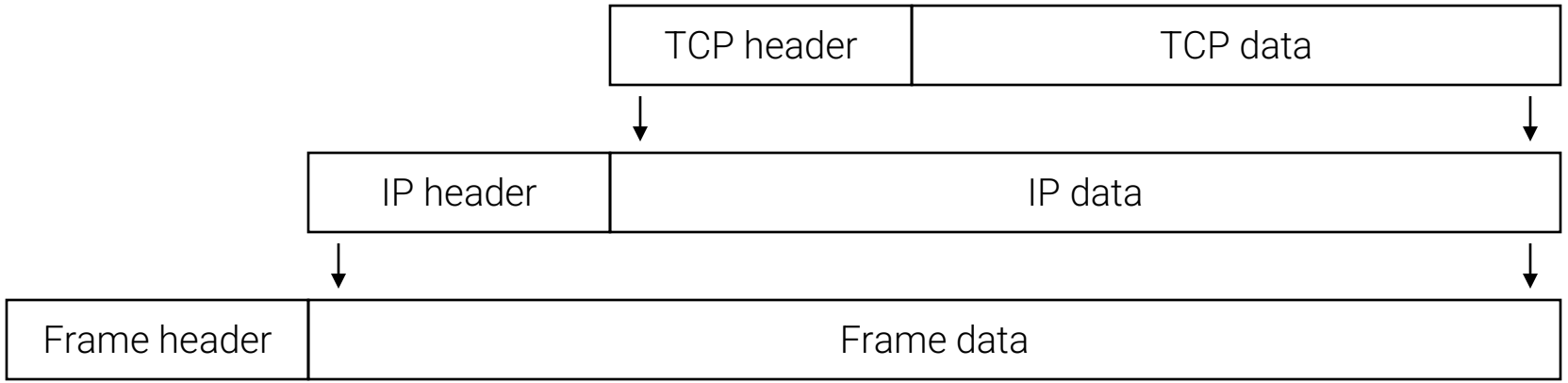
Transmission Control Protocol (TCP)

- The TCP protocol relies on IP to provide a connection-oriented, reliable stream-delivery service (no loss, no duplication, no transmission errors, correct ordering)
- TCP, as UDP, provides the port abstraction
- TCP allows two nodes to establish a virtual circuit, identified by source IP address, destination IP address, source TCP port, destination TCP port
- The virtual circuit is composed of two streams (full-duplex connection)
- The couple IP address/port number is sometimes called a socket (and the two streams are called a socket pair)

TCP Segment


0
4
8
12
16
20
24
28
31
Source port
Destination port
Sequence number
Acknowledgment number
HLEN
Reserved
Flags
Window
Checksum
Urgent pointer
Options
Padding
Data

TCP Encapsulation



TCP Seq/Ack Numbers

- The sequence number specifies the position of the segment data in the communication stream
  (SYN=13423 means: the payload of this segment contains the data from byte 13423 to byte 13458)
- The acknowledgment number specifies the position of the next byte expected from the communication partner
  (ACK = 16754 means: I have received correctly up to byte 16753 in the stream, I expect the next byte to be 16754)
- These numbers are used to manage retransmission of lost segments, duplication, flow control

TCP Window

- The TCP window is used to perform flow control
- Segment will be accepted only if their sequence numbers are inside the window that starts with the current acknowledgment number: ack number < sequence number < ack number + window
- The window size can change dynamically to adjust the amount of information sent by the sender

TCP Flags

- Flags are used to manage the establishment and shutdown of a virtual circuit
  - SYN: request for the synchronization of syn/ack numbers (used in connection setup)
  - ACK: states the acknowledgment number is valid (all segment in a virtual circuit have this flag set, except for the first one)
  - FIN: request to shutdown one stream
  - RST: request to immediately reset the virtual circuit
  - URG: states that the Urgent Pointer is valid
  - PSH: request a “push” operation on the stream (that is, the stream data should be passed to the user application as soon as possible)

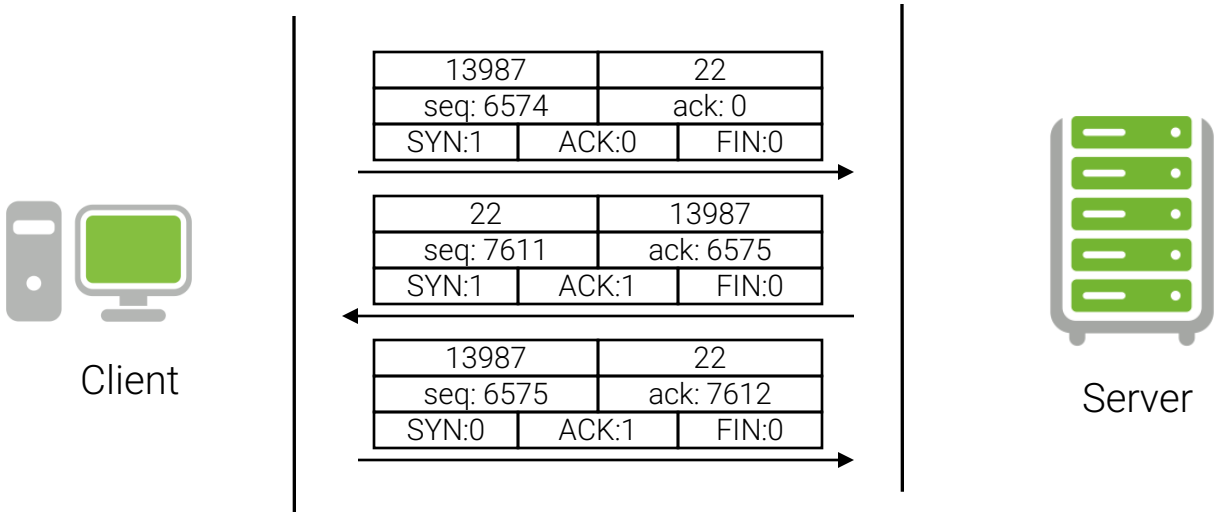
TCP Virtual Circuit: Setup

- A server, listening to a specific port, receives a connection request from a client: The segment containing the request is marked with the SYN flag and contains a random initial sequence number Sc
- The server answers with a segment marked with both the SYN and ACK flags and containing
  - an initial random sequence number Ss
  - Sc+ 1 as the acknowledgment number
- The client sends a segment with the ACK flag set and with sequence number Sc+ 1 and acknowledgment number Ss+ 1

What Initial Sequence Number?

- The TCP standard (RFC 793) specifies that the sequence number should be incremented every 4 microseconds
- BSD UNIX systems initially used a number that is incremented by 64,000 every half second (8 microseconds increments) and by 64,000 each time a connection is established

TCP: Three-way Handshake



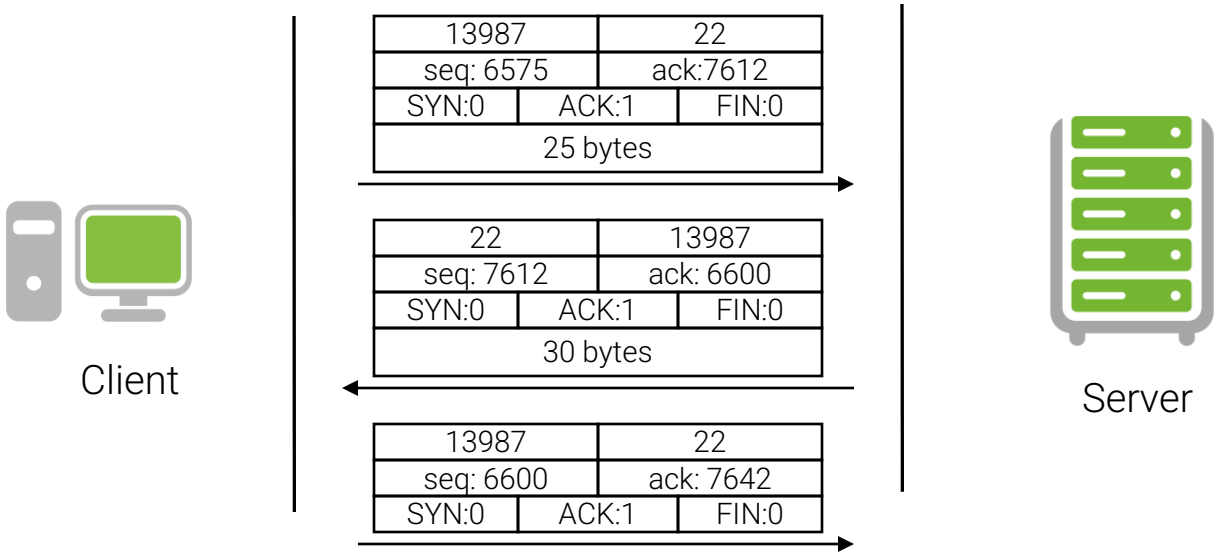
TCP: Three-way Handshake

```
arp who-has 192.168.1.20 tell 192.168.1.10
arp reply 192.168.1.20 is-at 0:10:4b:e2:f6:4c
192.168.1.10.1026 > 192.168.1.20.22: S 1015043:1015043(0)
192.168.1.20.22 > 192.168.1.10.1026: S 4056577943:4056577943(0) ack 1015044
192.168.1.10.1026 > 192.168.1.20.22: . ack 4056577944
```

TCP Virtual Circuit: Data Exchange

- A partner sends in each packet the acknowledgment of the previous segment and its own sequence number increased of the number of transmitted bytes
- A partner accepts a segment of the other partner only if the numbers are inside the transmission window
- An empty segment may be used to acknowledge the received data

TCP Virtual Circuit: Data Exchange



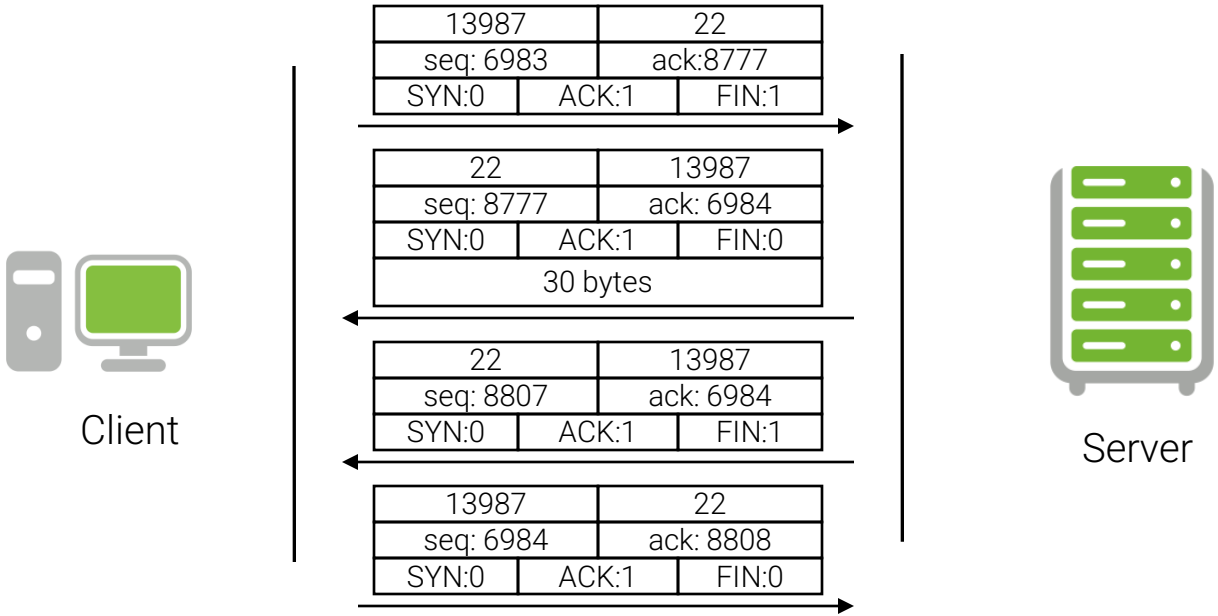
TCP Virtual Circuit: Data Exchange

```
192.168.1.20.22 > 192.168.1.10.1026: P 4056577944:4056577956(12) ack 1015044
192.168.1.10.1026 > 192.168.1.20.22: P 1015044:1015047(3) ack 4056577956
192.168.1.20.22 > 192.168.1.10.1026: . ack 1015047
192.168.1.10.1026 > 192.168.1.20.22: P 1015047:1015056(9) ack 4056577956
192.168.1.20.22 > 192.168.1.10.1026: P 4056577956:4056577962(6) ack 1015056
192.168.1.10.1026 > 192.168.1.20.22: P 1015056:1015066(10) ack 4056577962
192.168.1.20.22 > 192.168.1.10.1026: . ack 1015066
192.168.1.20.22 > 192.168.1.10.1026: P 4056577962:4056577977(15) ack 1015066
192.168.1.10.1026 > 192.168.1.20.22: P 1015066:1015069(3) ack 4056577977
192.168.1.20.22 > 192.168.1.10.1026: . ack 1015069
```

TCP Virtual Circuit: Shutdown

- One of the partners, say A, can terminate its stream by sending a segment with the FIN flag set
- The other partner, say B, answers with an ACK segment
- From that point on, A will not send any data to B: it will just acknowledge data sent by B
- When B shutdowns its stream the virtual circuit is considered closed

TCP Virtual Circuit: Shutdown



TCP Virtual Circuit: Shutdown

```
192.168.1.20.22 > 192.168.1.10.1026: F 4056579200:4056579200(0) ack 1016070
192.168.1.10.1026 > 192.168.1.20.22: . ack 4056579201
192.168.1.10.1026 > 192.168.1.20.22: F 1016070:1016070(0) ack 4056579201
192.168.1.20.22 > 192.168.1.10.1026: . ack 1016071
```

TCP Portscan

- Used to determine the TCP services available on a victim host
- Most services are statically associated with port numbers (see /etc/services in UNIX systems)
- In its simplest form (connect() scanning), the attacker tries to open a TCP connection to all the 65535 ports of the victim host
- If the handshake is successful then the service is available
- Advantage: no need to be root
- Disadvantage: very noisy

connect() Scan

```
root@localhost/home/vigna: nmap -sT 192.168.1.20
Starting Nmap
Interesting ports on  (192.168.1.20):
(The 1500 ports scanned but not shown below are in state: closed)
Port       State       Service
7/tcp      open        echo
9/tcp      open        discard
11/tcp     open        systat
13/tcp     open        daytime
15/tcp     open        netstat
19/tcp     open        chargen
21/tcp     open        ftp
22/tcp     open        ssh
23/tcp     open        telnet
512/tcp    open        exec
513/tcp    open        login
514/tcp    open        shell
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

TCP SYN Scanning

- AKA “half-open” scanning
- The attacker sends a SYN packet
- The server answers with a SYN/ACK packet if the port is open or (usually) with a RST packet if the port is closed
- The attacker sends a RST packet instead of the final ACK
- The connection is never open and the event is not logged by the operating system/application

TCP SYN Scanning

```
# nmap -sS 128.111.38.78
Port       State       Service
80/tcp     open        http

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second

11:27:32.249220 128.111.48.69.47146 > 128.111.41.38.78: S 3886663922:3886663922(0) win 2048
11:27:32.266910 128.111.48.69.47146 > 128.111.41.38.78: S 3886663922:3886663922(0) win 2048
11:27:32.266914 128.111.48.69.47146 > 128.111.41.38.81: S 3886663922:3886663922(0) win 2048
11:27:32.266918 128.111.48.69.47146 > 128.111.41.38.82: S 3886663922:3886663922(0) win 2048
11:27:32.266923 128.111.48.69.47146 > 128.111.41.38.80: S 3886663922:3886663922(0) win 2048
11:27:32.266925 128.111.48.69.47146 > 128.111.41.38.79: S 3886663922:3886663922(0) win 2048
11:27:32.267904 128.111.41.38.78 > 128.111.48.69.47146: R 0:0(0) ack 3886663923 win 0 (DF)
11:27:32.267970 128.111.41.38.81 > 128.111.48.69.47146: R 0:0(0) ack 3886663923 win 0 (DF)
11:27:32.268038 128.111.41.38.82 > 128.111.48.69.47146: R 0:0(0) ack 3886663923 win 0 (DF)
11:27:32.268106 128.111.41.38.80 > 128.111.48.69.47146: S 1441896698:1441896698(0) ack 3886663923 win 5840 <mss 1460> (DF)
11:27:32.268121 128.111.48.69.47146 > 128.111.41.38.80: R 3886663923:3886663923(0) win 0 (DF)
11:27:32.268174 128.111.41.38.79 > 128.111.48.69.47146: R 0:0(0) ack 3886663923 win 0 (DF)
```

TCP FIN Scanning

- The attacker sends a FIN-marked packet
- In most TCP/IP implementations
  - If the port is closed a RST packet is sent back
  - If the port is open the FIN packet is ignored (timeout)
- In Windows a RST is sent back in any case, so that all ports appear to be closed
- Variation of this type of scanning technique
  - Xmas: FIN, PSH, URG set
  - Null: no flags set

TCP FIN Scanning

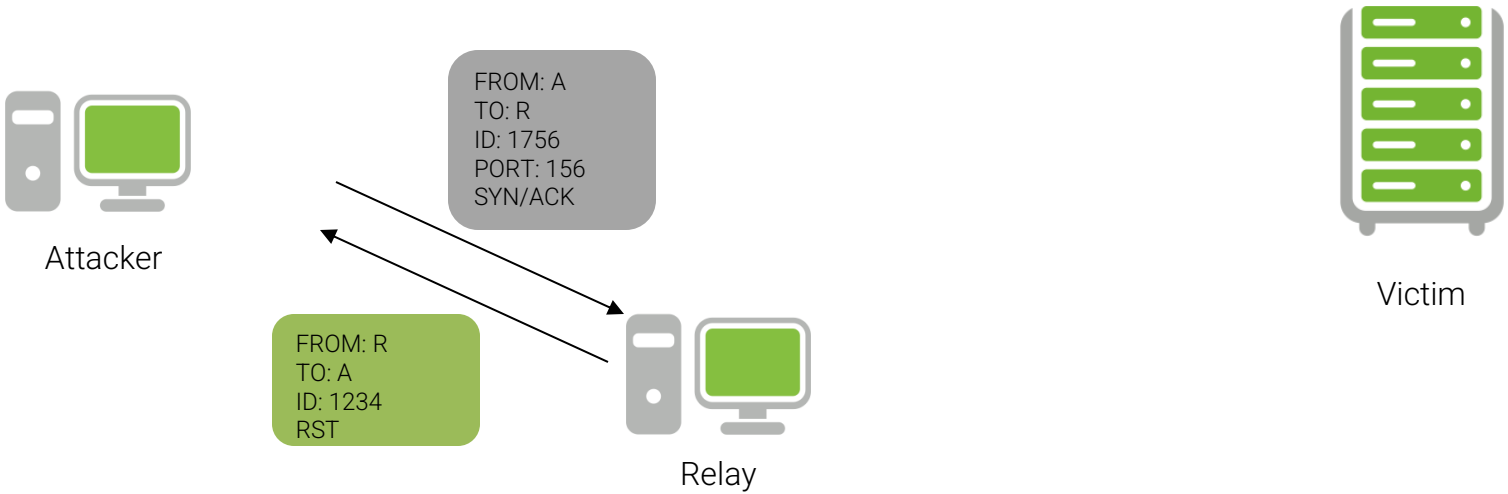
```
# nmap -sF 128.111.41.38

Starting nmap ( www.insecure.org/nmap/ )
Port       State       Service
80/tcp     open        http

11:39:07.356917 128.111.48.69.38772 > 128.111.41.38.79: F 0:0(0) win 1024
11:39:07.356921 128.111.48.69.38772 > 128.111.41.38.82: F 0:0(0) win 1024
11:39:07.356925 128.111.48.69.38772 > 128.111.41.38.81: F 0:0(0) win 1024
11:39:07.356927 128.111.48.69.38772 > 128.111.41.38.80: F 0:0(0) win 1024
11:39:07.356931 128.111.48.69.38772 > 128.111.41.38.78: F 0:0(0) win 1024
11:39:07.357918 128.111.41.38.79 > 128.111.48.69.38772: R 0:0(0) ack 1 win 0 (DF)
11:39:07.357983 128.111.41.38.82 > 128.111.48.69.38772: R 0:0(0) ack 1 win 0 (DF)
11:39:07.358051 128.111.41.38.81 > 128.111.48.69.38772: R 0:0(0) ack 1 win 0 (DF)
11:39:07.358326 128.111.41.38.78 > 128.111.48.69.38772: R 0:0(0) ack 1 win 0 (DF)
11:39:07.666939 128.111.48.69.38773 > 128.111.41.38.80: F 0:0(0) win 1024
11:39:07.976951 128.111.48.69.38772 > 128.111.41.38.80: F 0:0(0) win 1024
11:39:08.286929 128.111.48.69.38773 > 128.111.41.38.80: F 0:0(0) win 1024
```

Idle Scanning

- Idle scanning leverages a victim host to “relay” the scan
- The attacker sends spoofed TCP SYN packets to the target
- The packets appear to come from the victim
- The target replies to the victim
  - If the target replies with a SYN+ACK packet (open port) then the victim will send out a RST
  - If the target replies with a RST (closed port) then the victim will not send out any packet
- The attacker checks the IP datagram ID of the victim before and after each port probe
  - If it has increased: port on target was open
  - If it has not increased: port on target was closed

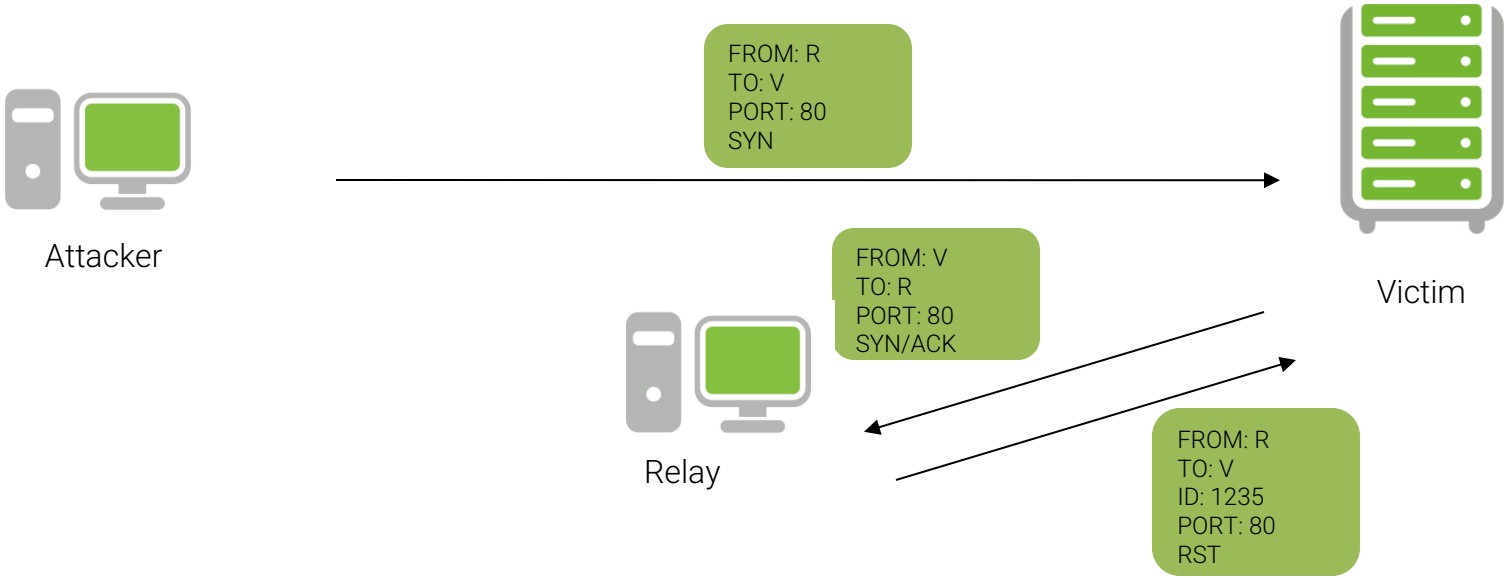
Idle Scanning

- Step 1: Determine the relay's initial IP sequence number



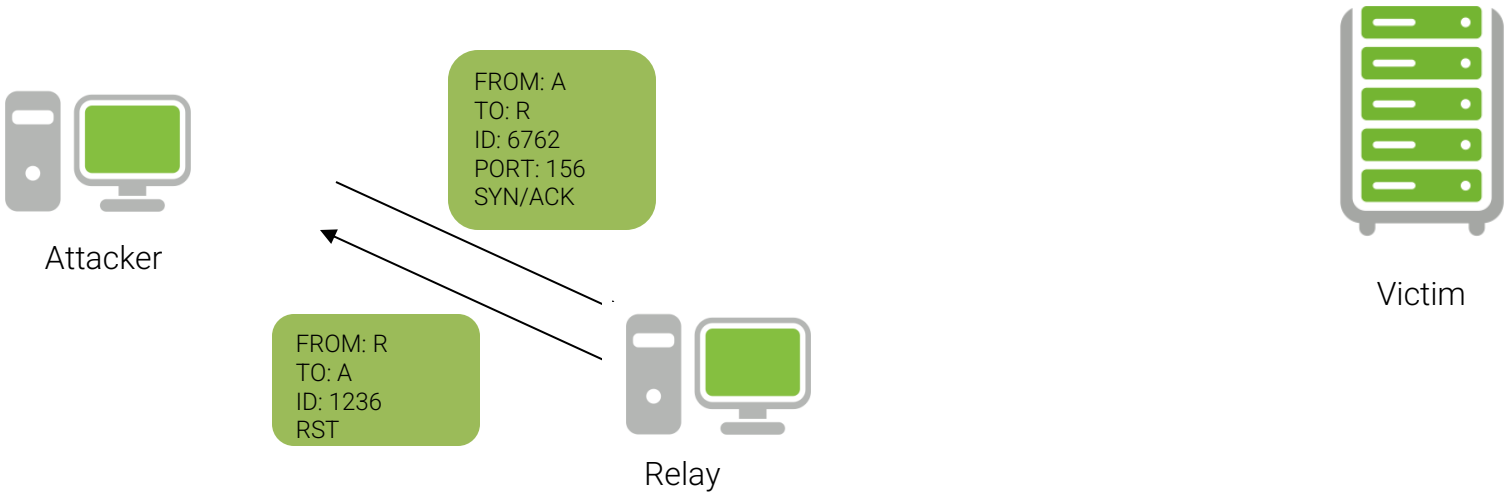
Idle Scanning

- Step 2: Send a spoofed connection request



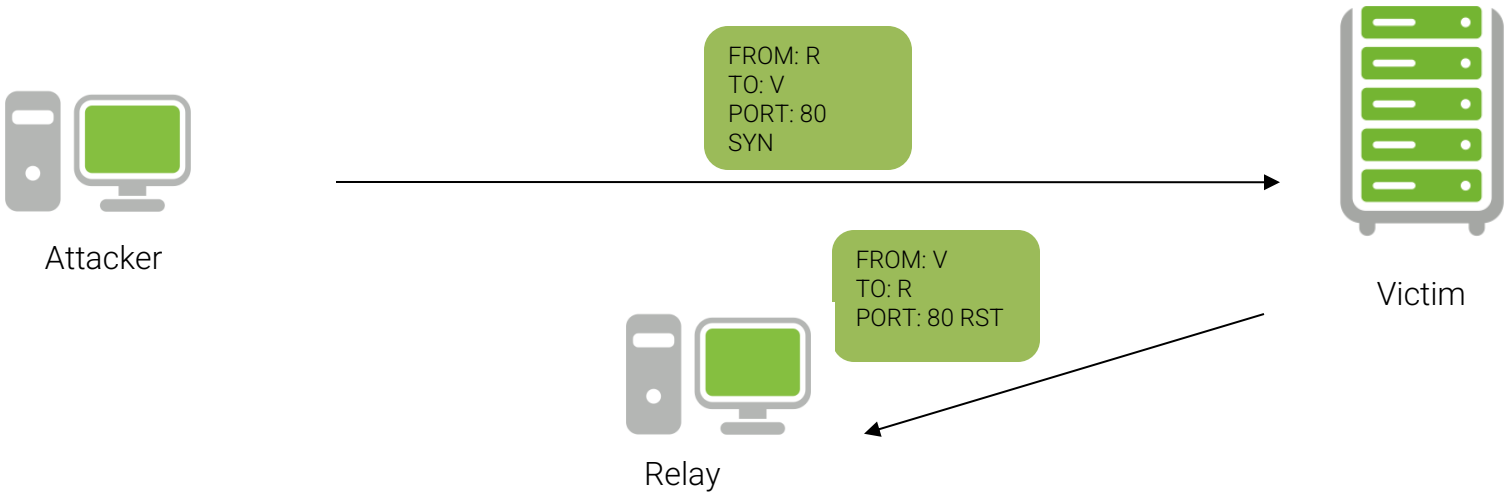
Idle Scanning

- Step 3: Determine the relay's final IP sequence number



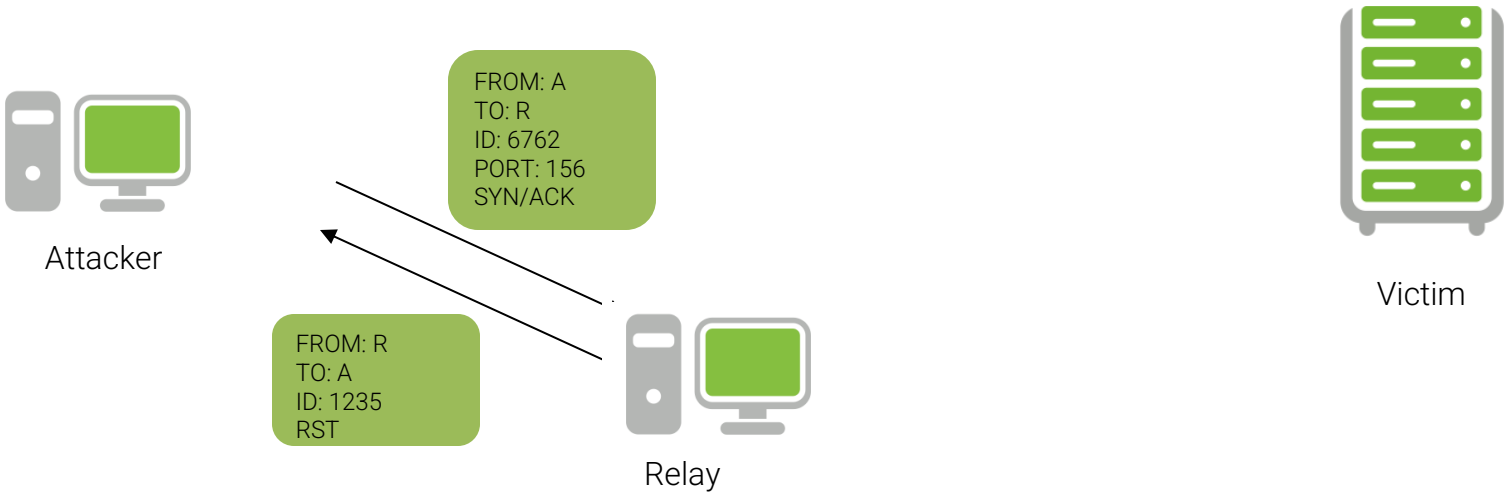
Idle Scanning

- Step 2b: Send a spoofed connection request to a closed port



Idle Scanning

- Step 3b: Determine the relay's final IP sequence number



OS Fingerprinting

- OS fingerprinting allows one to determine the operating system of a host by examining the reaction to carefully crafted packets
  - Wrong answers to FIN TCP packets
  - “Undefined” flags in the TCP header of a request are copied verbatim in the reply
  - Weird combinations of flags in the TCP header
  - Selection of TCP initial sequence numbers
  - Selection of initial TCP window size
  - Analysis of the use of ICMP messages
    - Error rate
    - Amount of offending datagram included
  - TCP options
- OS fingerprinting also can be performed in a passive way using tools such as p0f, ettercap

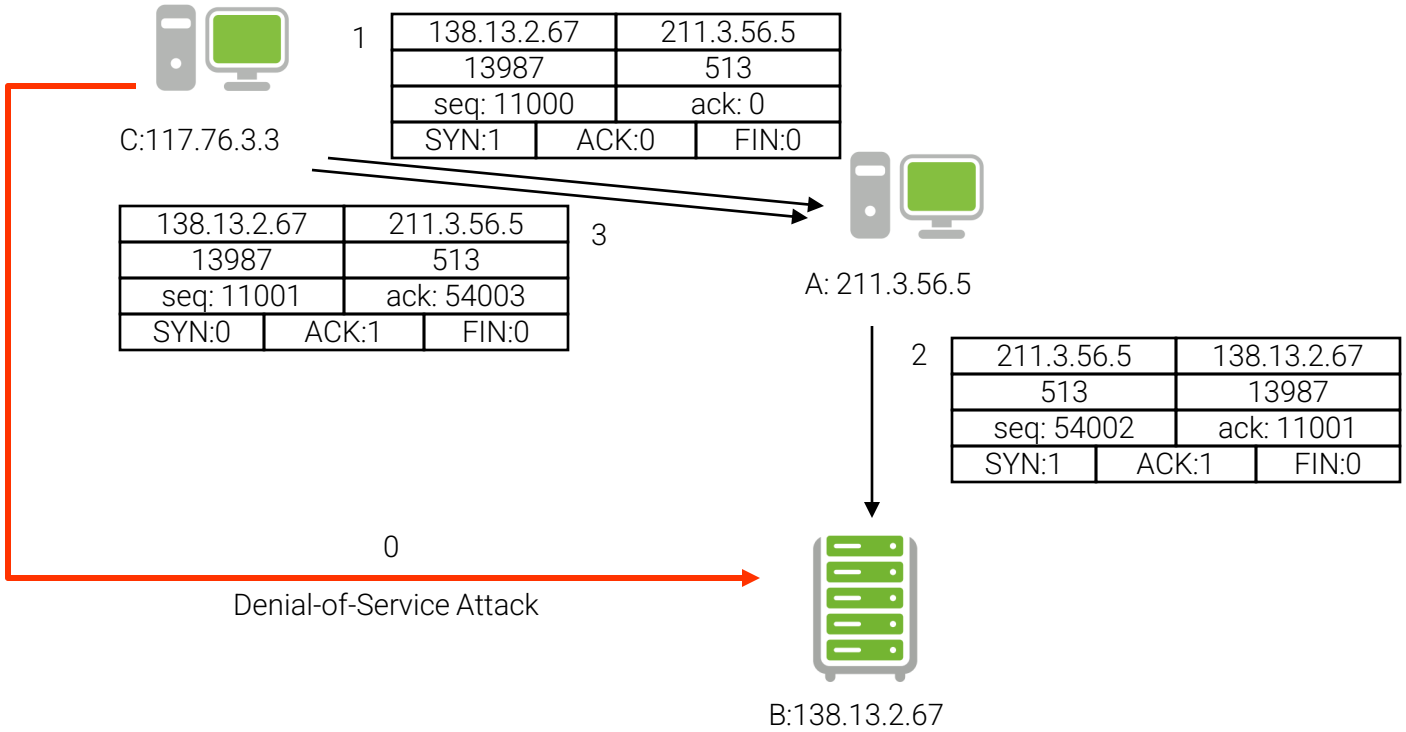
TCP Spoofing

- Attack aimed at impersonating another host when establishing a TCP connection
- First discussed by R.T. Morris in “A Weakness in the 4.2BSD Unix TCP/IP Software” in 1985
- Used by Mitnick in his attack against SDSC in 1994

TCP Spoofing

- Node A trusts node B (e.g., login with no password if the TCP connection comes from a specific IP)
- Node C wants to impersonate B with respect to A in opening a TCP connection
- C kills B (flooding, crashing, redirecting) so that B does not send annoying RST segments
- C sends A a TCP SYN segment in a spoofed IP packet with B's address as the source IP and Sc as the sequence number
- A replies with a TCP SYN/ACK segment to B with Ss as the sequence number. B ignores the segment: dead or too busy
- C does not receive this segment but to finish the handshake it has to send an ACK segment with Ss + 1 as the acknowledgment number
  - C eavesdrop the SYN/ACK segment
  - C guesses the correct sequence number

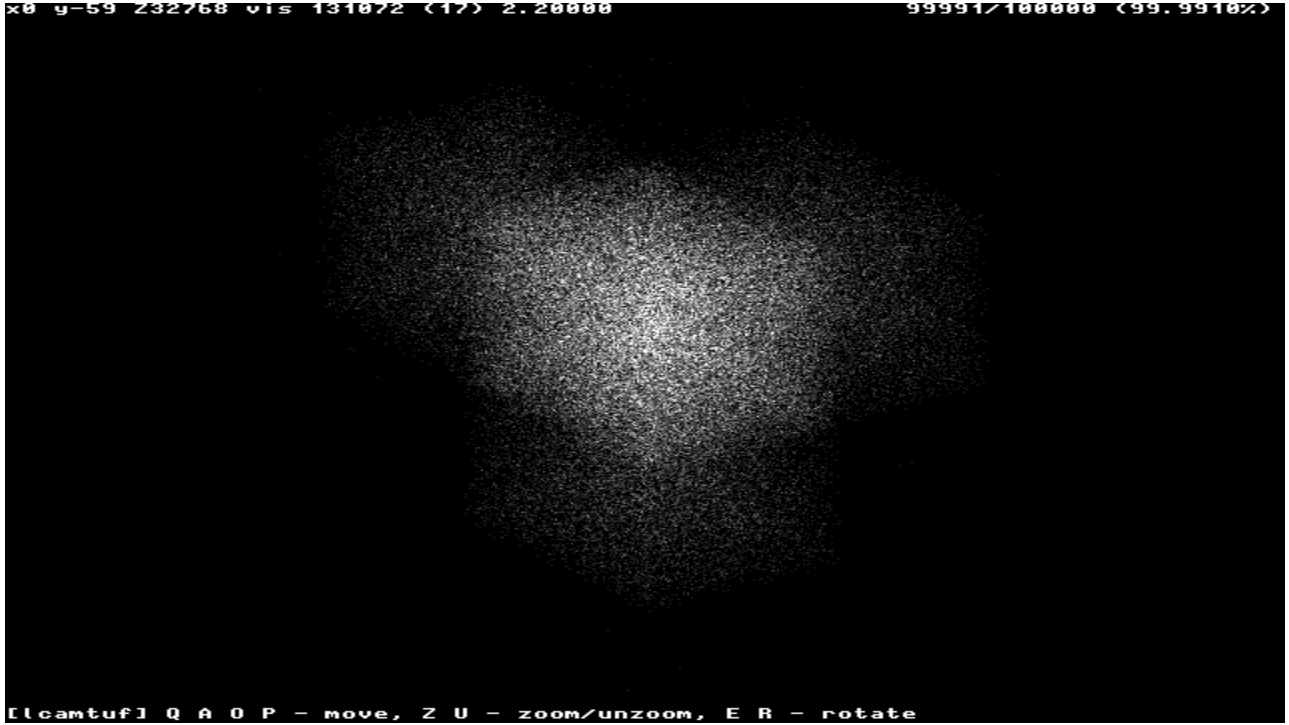
TCP Spoofing



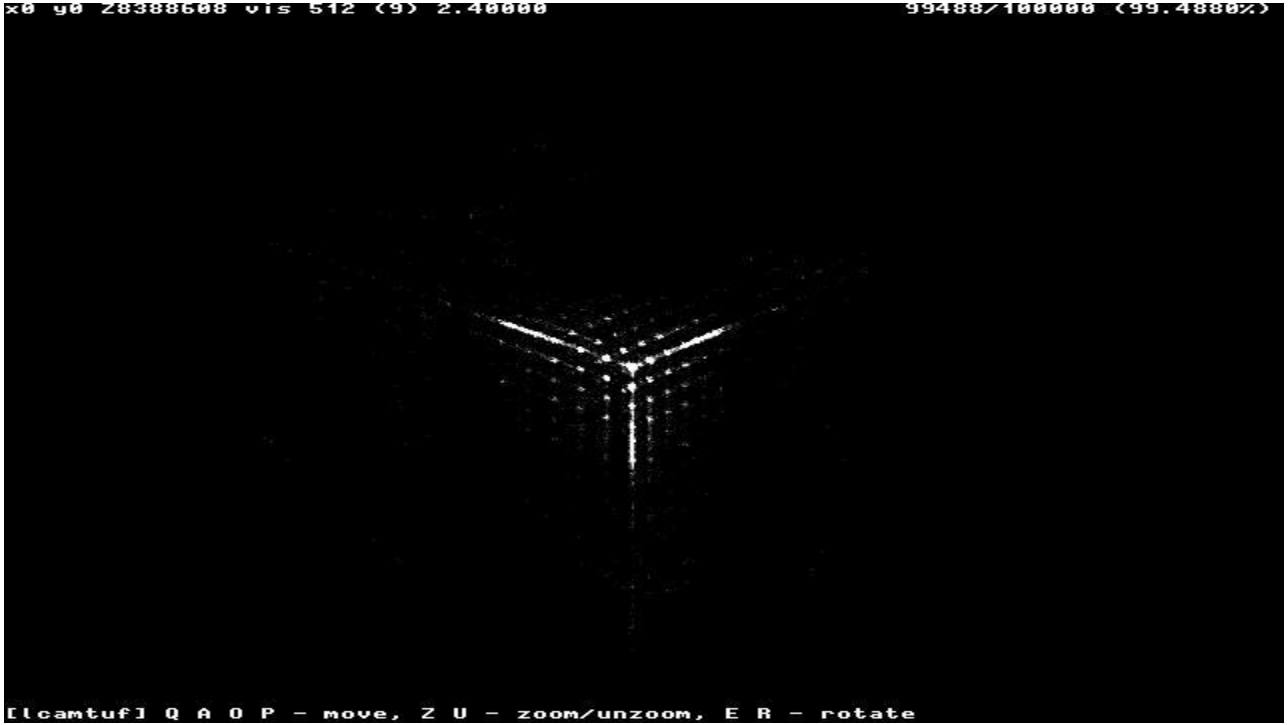
Choosing The Right Sequence Number

- RFC 1948 defines ways to improve sequence number generation
- Some implementations still don’t get it
- See Michal Zalewski’s paper “Strange Attractors and TCP/IP Sequence Number Analysis” and its update “One Year Later”
- Builds a graph using a composition of the values seen recently in a series of sequence numbers:
  - x[n] = s[n-2] - s[n-3]
  - y[n] = s[n-1] - s[n-2]
  - z[n] = s[n] - s [n-1]

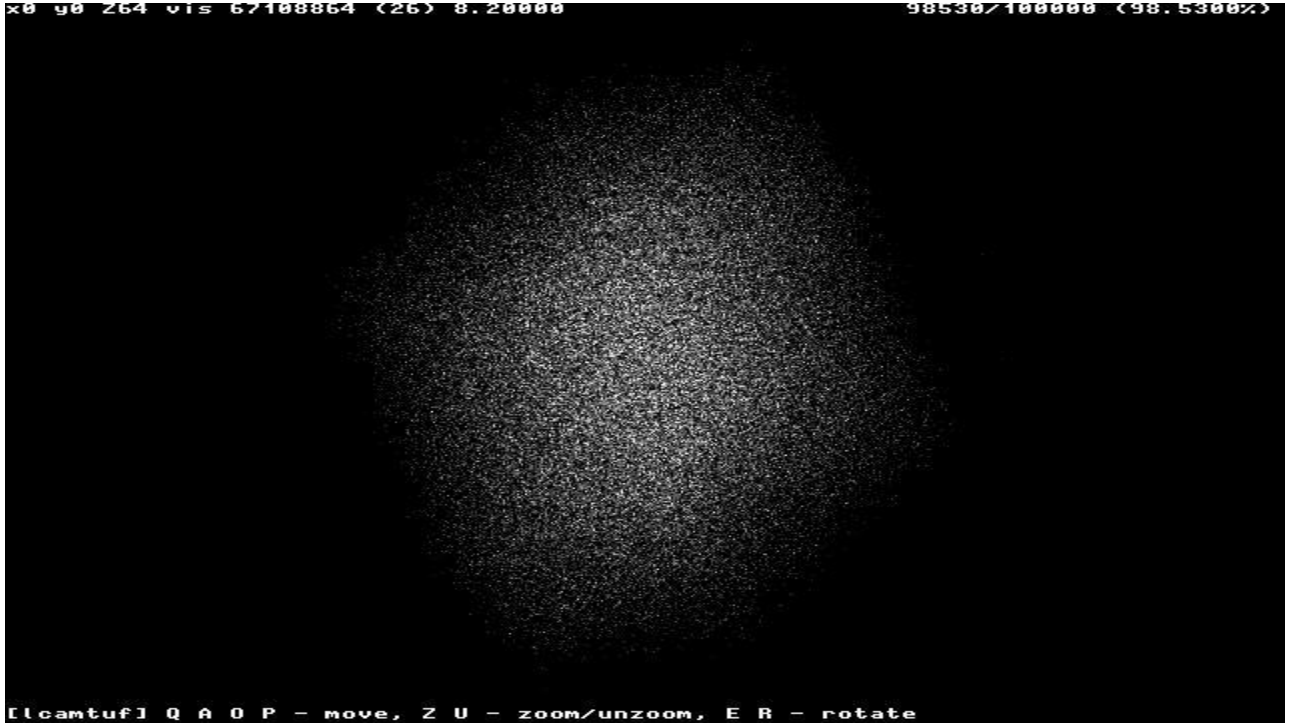
Windows 2000/XP



Windows 95/98



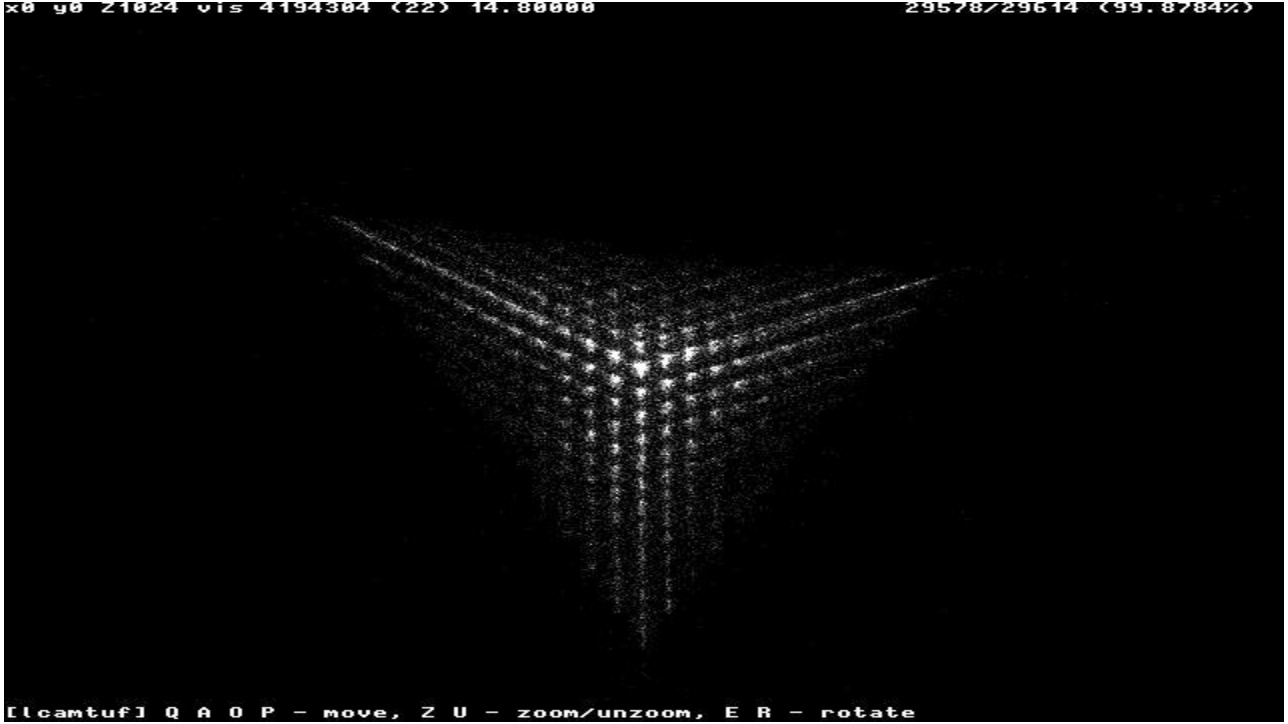
Linux


x0 y0 z64 vis 67108864 (26) 8.20000
98530/100000 (98.5300%)
[lcamtuf] Q A O P - move, Z U - zoom/unzoom, E R - rotate

Free BSD

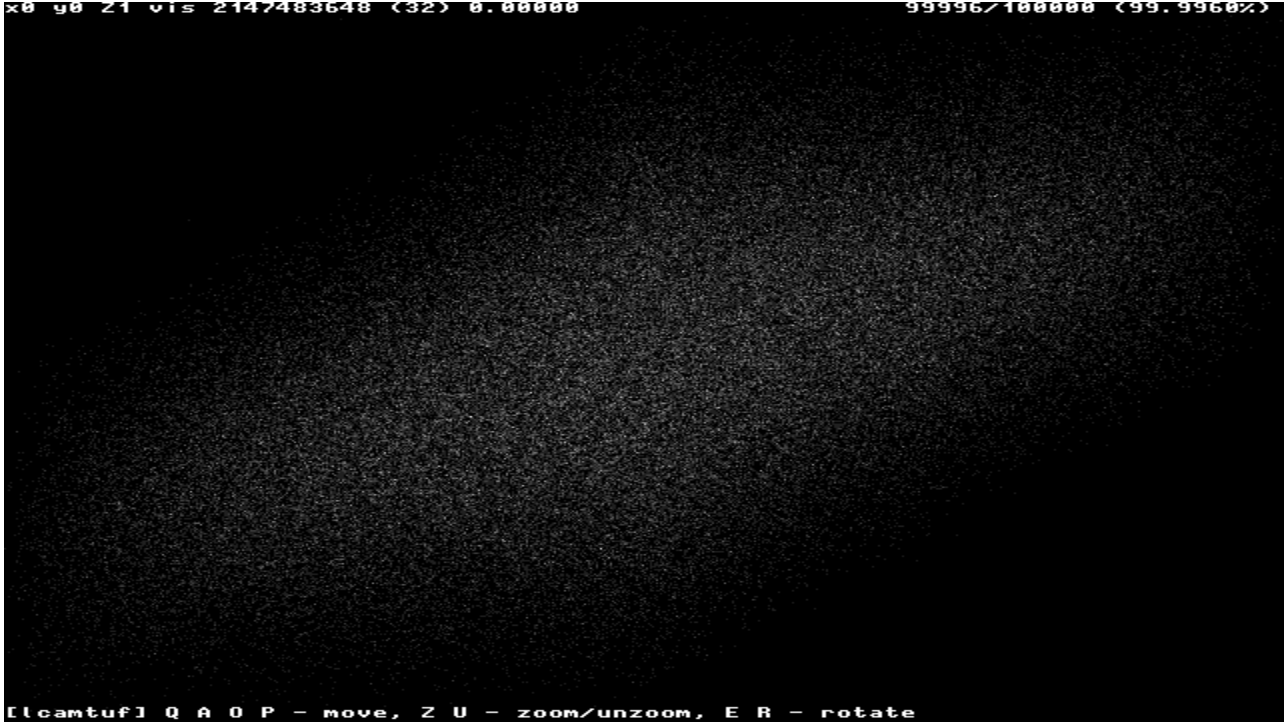

;)0 g0 20 vis 2147483648 (33) 0.00000
100000/100000 (100.0000%
[lcamtuf] Q A O P - move, Z U - zoom/unzoom, E R - rotate

Cisco IOS Before The Cure

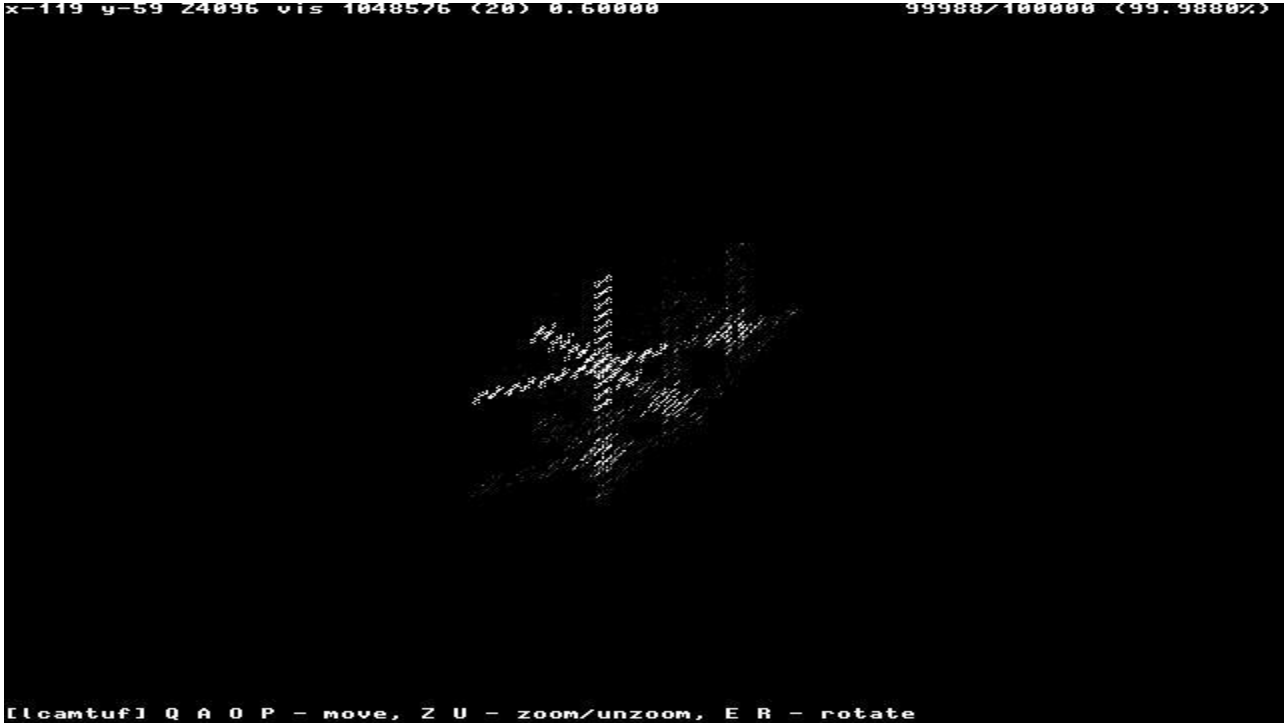


Cisco IOS After The Cure

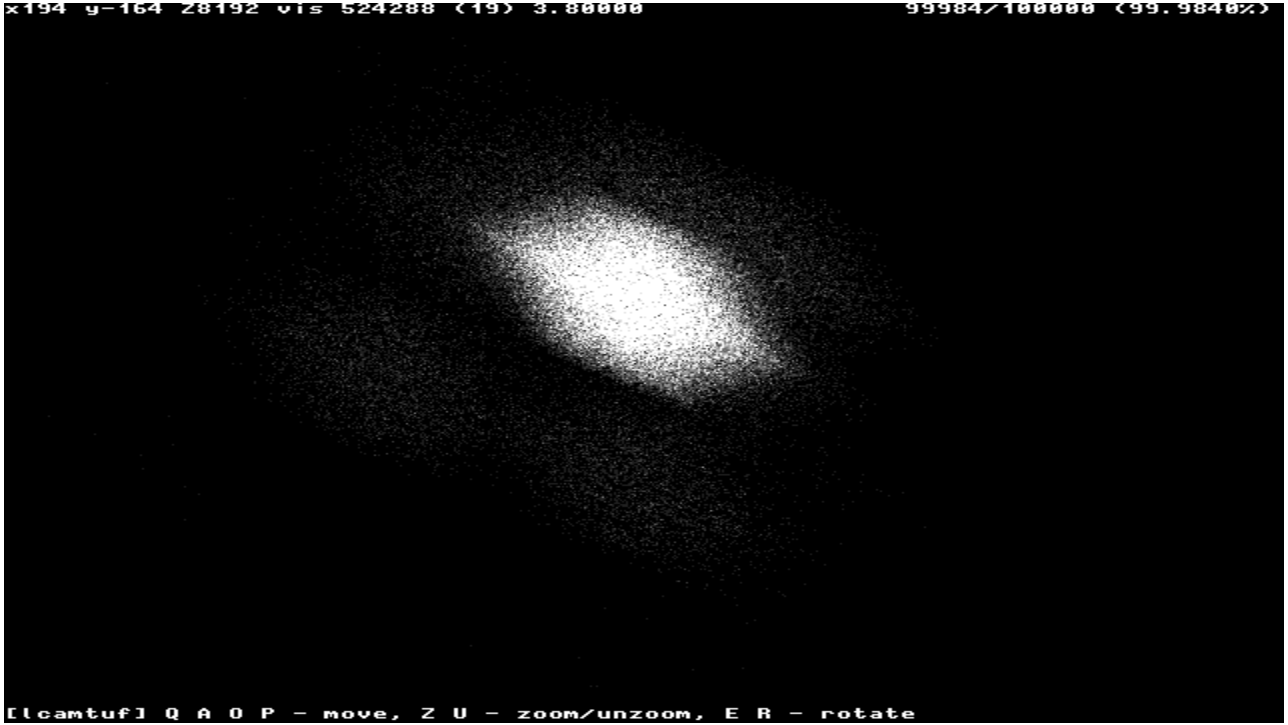
MacOS X


x0 y0 z1 vis 2147483648 (32) 0.00000
99996/100000 (99.9960%)
[lcamtuf] Q A O P - move, Z U - zoom/unzoom, E R - rotate

HP-UX Before The Cure



HP-UX After The Cure



IRIX (w/out MD5 generation)

TCP Hijacking

- Powerful technique to take control of an existing TCP connection
- The attacker uses spoofed TCP segments to
  - Insert data in the streams
  - Reset an existing connection (denial of service)
- The correct sequence/acknowledgment numbers must be used
  - The attacker can eavesdrop the traffic between client and server
  - The attacker can guess the correct seq/ack numbers
- Described in “A Simple Active Attack Against TCP” by L. Joncheray

TCP Hijacking

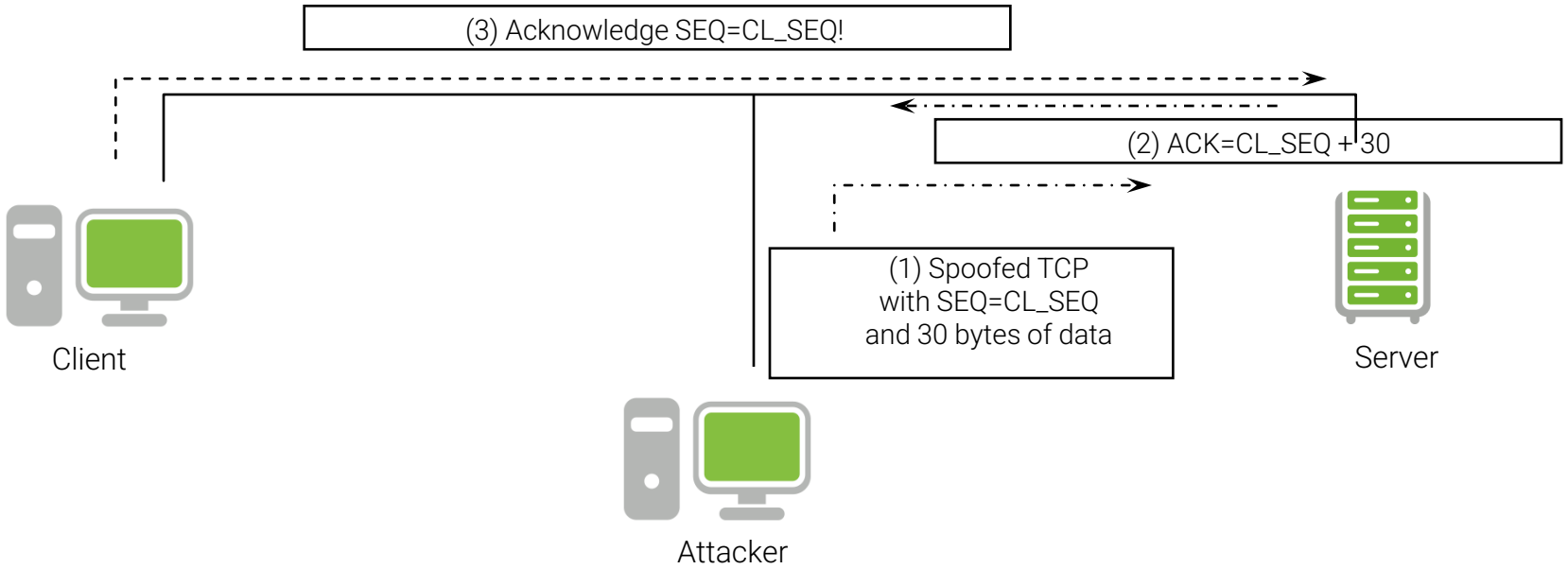
- The attacker waits until the connection is “quiet”
  - All the transmitted data have been acknowledged (by both endpoints)
- The attacker injects the data in the stream
  - “Desynchronizes” the connection
- The receiver of the injected data sends an acknowledgment to the apparent sender
- The apparent sender replies with an acknowledgement with the “expected” sequence number
- The receiver considers this as out-of-sync and sends an an acknowledgement with the “expected” sequence number
- ....

TCP Hijacking

- ACK messages with no data are not retransmitted in case of loss
- The “ACK storm” continues until one message is lost
- Any subsequent attempt to communicate will generate an ACK storm
- ACK storms can be blocked by the attacker using ACK packets with the right numbers

TCP Hijacking

CL_SEQ = SVR_ACK
SVR_SEQ = CL_ACK



TCP Hijacking

- This technique can be used against both client and server to completely hijack the communication channel (man-in-the-middle attack)

ACK Storm

```
192.168.1.10.1026 > 192.168.1.20.23: P 1015112:1015133(21) ack 4056578923
192.168.1.20.23 > 192.168.1.10.1026: . seq 4056578923 ack 1015133
192.168.1.10.1026 > 192.168.1.20.23: . seq 1015112 ack 4056578923
192.168.1.20.23 > 192.168.1.10.1026: . seq 4056578923 ack 1015133
192.168.1.10.1026 > 192.168.1.20.23: . seq 1015112 ack 4056578923
192.168.1.20.23 > 192.168.1.10.1026: . seq 4056578923 ack 1015133
192.168.1.10.1026 > 192.168.1.20.23: . seq 1015112 ack 4056578923
192.168.1.20.23 > 192.168.1.10.1026: . seq 4056578923 ack 1015133
192.168.1.10.1026 > 192.168.1.20.23: . seq 1015112 ack 4056578923
192.168.1.20.23 > 192.168.1.10.1026: . seq 4056578923 ack 1015133
192.168.1.10.1026 > 192.168.1.20.23: . seq 1015112 ack 4056578923
192.168.1.20.23 > 192.168.1.10.1026: . seq 4056578923 ack 1015133
192.168.1.10.1026 > 192.168.1.20.23: . seq 1015112 ack 4056578923
192.168.1.20.23 > 192.168.1.10.1026: . seq 4056578923 ack 1015133
192.168.1.10.1026 > 192.168.1.20.23: . seq 1015112 ack 4056578923
192.168.1.20.23 > 192.168.1.10.1026: . seq 4056578923 ack 1015133
192.168.1.10.1026 > 192.168.1.20.23: . seq 1015112 ack 4056578923
```

TCP Hijacking Variants

- “Early desynchronization” can be achieved by the attacker by resetting existing connections and immediately opening new ones (between the same ports) with different initial sequence numbers
- In a “veto attack” the attacker predicts the size of the next packet that will be sent by the victim: by sending a packet with exactly the same size, it is possible to avoid the ack storm

SYN-flooding Attack

- Very common denial-of-service attack, aka Neptune
- Attacker starts handshake with SYN-marked segment
- Victim replies with SYN-ACK segment
- Attacker… stays silent
  - Note that the source IP of the attacker can be spoofed, since no final ACK is required
- A host can keep a limited number of TCP connections in half-open state. After that limit, it cannot accept any more connections

SYN-flooding Attack

- Current solutions
  - Filtering
  - Increase the length of the half-open connection queue
  - Reduce the SYN-received timeout
  - Drop half-open connections when the limit has been reached and new requests for connection arrive
  - Limit the number of half-open connections from a specific source
  - Use SYN cookies
- See TCP SYN Flooding Attacks and Common Mitigations, RFC 4987

SYN Cookies

- Special algorithm used for determining the initial sequence number of the server
- The number is
  - Top 5 bits: t mod 32, where t is a 32-bit time counter that increases every 64 seconds
  - Following 3 bits: the encoding of the Maximum Segment Size (MSS) chosen by the server in response to the client's MSS
  - A keyed hash of:
    - Counter t
    - Source/Destination IP addresses and ports

SYN Cookies

- A server that uses SYN cookies sends back a SYN+ACK, exactly as if the SYN queue had been larger
- When the server receives an ACK, it checks that the secret function works for a recent value of t, and then rebuilds the SYN queue entry (using the encoded MSS info)
- Drawbacks:
  - The server sequence number grows faster than normal
  - The MSS value is limited by the encoding procedure (only 8 possible values)
  - No data can be included in the initial SYN

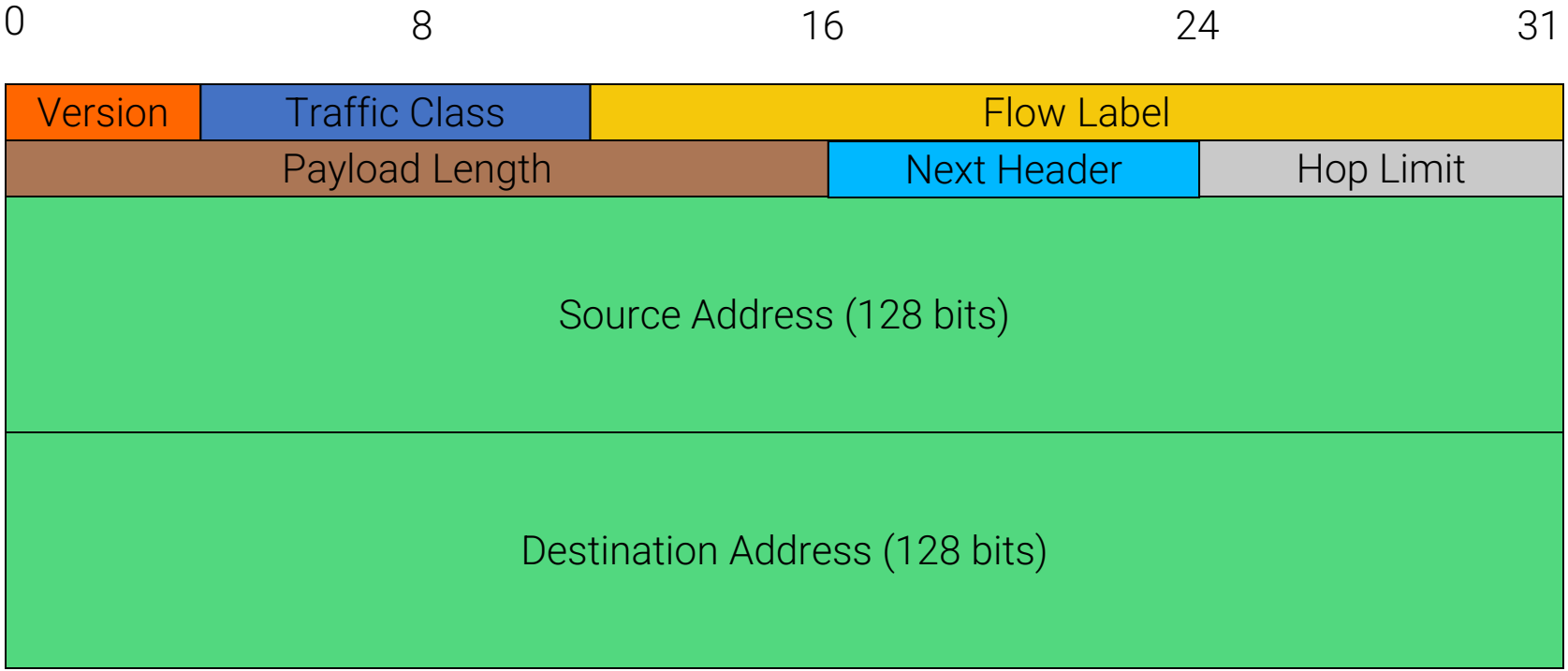
State Attacks

- There are other attacks that exploit the fact that the server has to maintain a certain amount of memory/resources associated with each open TCP connection
  - Memory for the socket descriptor
  - Process or thread to manage the connection
  - Memory associated with the data in the TCP stream that has not yet been acknowledged
  - …

IPv6

- New version of the IP protocol
- Introduced to solve major problems with IPv4
  - Shortage of IP addresses
  - Increase of routing information
  - End-to-end communication disruption caused by NAT-ing (which is often used to deal with the IP address shortage)
- Uses 128-bit addresses
- Supports better routing, QoS, extended options, security

IPv6 Header


0
8
16
24
31
Version
Traffic Class
Flow Label
Payload Length
Next Header
Hop Limit
Source Address (128 bits)
Destination Address (128 bits)

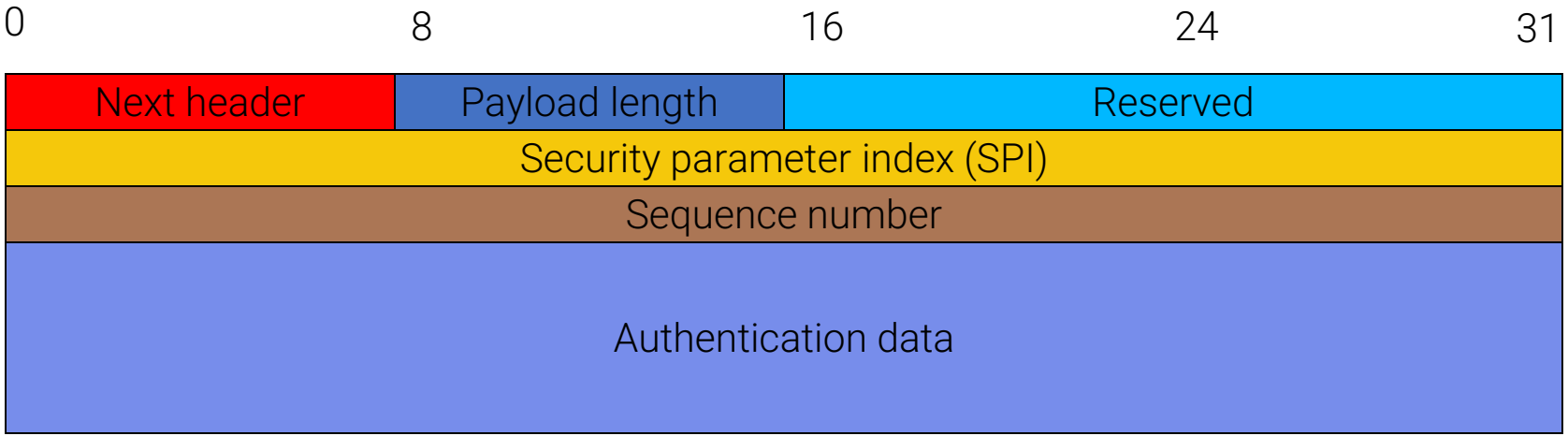
IPSec

- Security built into the IP layer
- Part of the IPv6 standard
- Can be used as an extension to IPv4
- Supports gradual transition from IPv4 to IPv6
- Provides authentication and confidentiality
- Provides support for Virtual Private Networks (VPNs)
- Defines infrastructure for key management

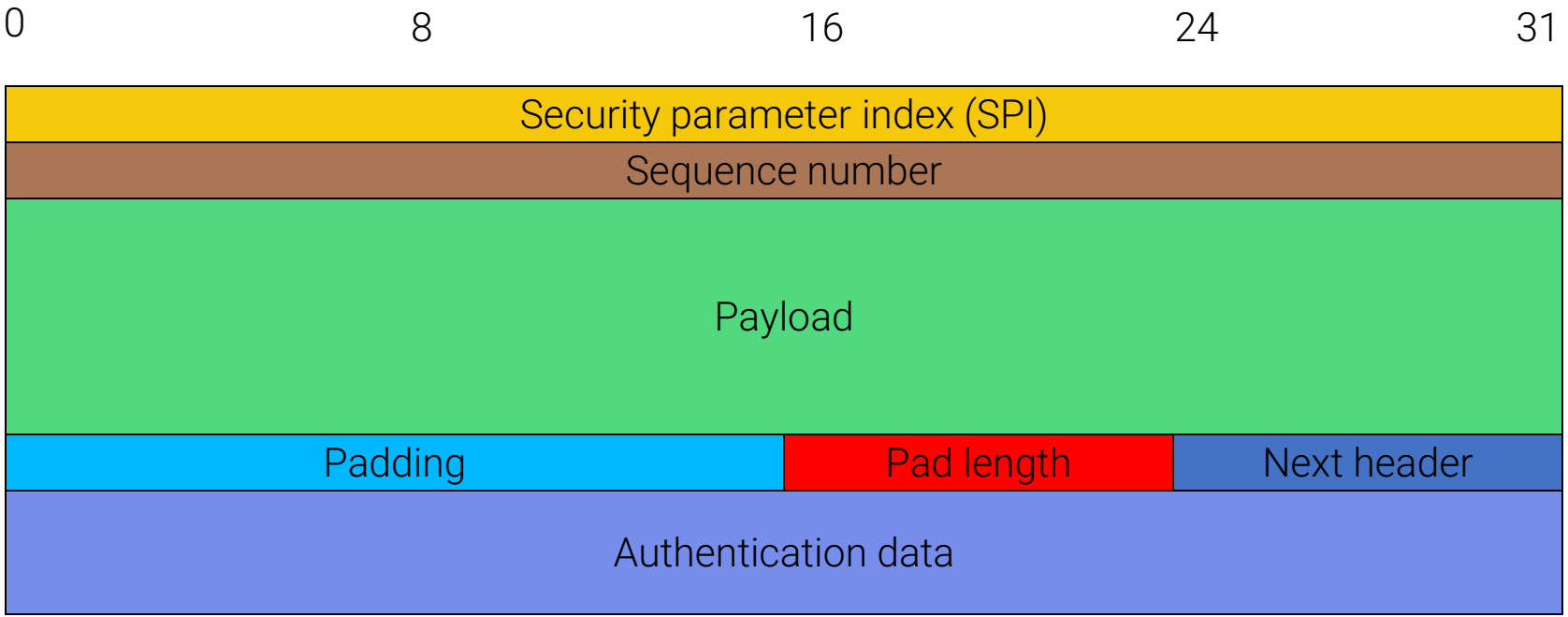
Basic Concepts

- Security association (SA)
  - A security association is defined before starting to communicate
  - The security association specifies the security parameters for a particular communication channel
    - Algorithms used for authentication
    - Algorithms used for bulk encryption
    - Validity of the association
  - Each IP datagram contains a reference (Security Parameter Index) to the relevant SA
- Authentication Header (AH)
  - Extension that provides authentication and integrity with HMAC
- Encapsulating Security Payload (ESP)
  - Extension that provides confidentiality

Authentication Header



Encapsulating Security Payload



Some Confusion…

- Initially:
  - AH only for authentication and integrity checking
  - ESP only for confidentiality
  - AH + ESP for all of the above
- Some flaws were found when ESP alone is used
  - Protocol extended
- Currently:
  - AH if only authentication and integrity checking are required
  - ESP for confidentiality, authentication, and integrity

Modes

- Transport mode
  - Use in host-to-host communication
  - AH authenticates parts of the IP header and the whole payload
  - ESP used to encrypt the IP payload
- Tunnel mode
  - The original datagram is encapsulated in another datagram
  - AH authenticates the encapsulated IP datagram plus parts of the header of the encapsulating datagram
  - ESP used to encrypt the encapsulated datagram

Internet Key Exchange

- Prior to an IP datagram being exchanged between two hosts, a Security Association must be in place
- SAs can be created manually or dynamically using the IKE
- The IKE is based on the Internet Security Association and Key Management Protocol (ISAKMP)
- Usually IKE is performed by a daemon

IPv6 Security Risks

- IPv6 provides better security than IPv4
- Some network security tools that are not able to handle IPv6
  - IPv6 traffic can be used to avoid detection

- Suggested reading: “IPv6 Security: Attacks and Countermeasures in a Nutshell”, in Proceedings of WOOT, 2014

Wireless Networks

- Wireless networks allow to exchange network packets over a radio link
- Protocols are defined in the 802.11 standard series
- 802.11 supports bit rates up to 2 Mbps
- 802.11b-g provides bit rates up to 11 Mbps (2.4 GHz band)
- 802.11a provides bit rates up to 54 Mbps (5 GHz band)
- 802.11n uses both 2.4GHz and 5GHz bands for a bit rate of 600 Mb/s
- 802.11ac provides bandwidth up to 3.5 Gb/s
- 802.11ad provides bit rates up to 6.75 Gb/s (60 GHz band)

802.11 Data Link Layer

- Logical Link Control (LLC) layer shared with other 802 lans (48-bit addresses)
- Different Media Access Control (MAC)
  - Wired Ethernet uses Carrier Sense Multiple Access with Collision Detection (CSMA/CD)
  - Near/far problem radio transmission does not allow a station to listen for collisions
    - Stronger signals make weaker signals become noise
  - WLAN: Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)

CSMA/CA

- Station A waits for “no activity”
- Station A waits for a random amount of time
- Station A starts sending
- Station B sends an ACK
- If station A does not receive a valid ACK
  - It assumes that the packet was lost or the ACK was lost
  - The packet is retransmitted after a random amount of time
- Robustness features
  - Ready To Send/Clear To Send (RTC/CTS)
  - CRC checksums

Modes of Operation

- Infrastructure mode
  - Wireless access point (AP) connected to wired network
  - Mobile stations with wireless cards
- Ad hoc mode
  - Mobile stations with wireless cards
  - Traffic not directed to in-range hosts must be routed by a mobile station

Type of Frames

- Management frames
  - Authentication frames
    - Open networks require a simple request-response
    - Networks protected by shared-key authentication require a number of steps in order to authenticate the client
  - De-authentication frames
  - Association request/response frames
  - Disassociation frames
  - Beacon frames
  - Probe request/response frames

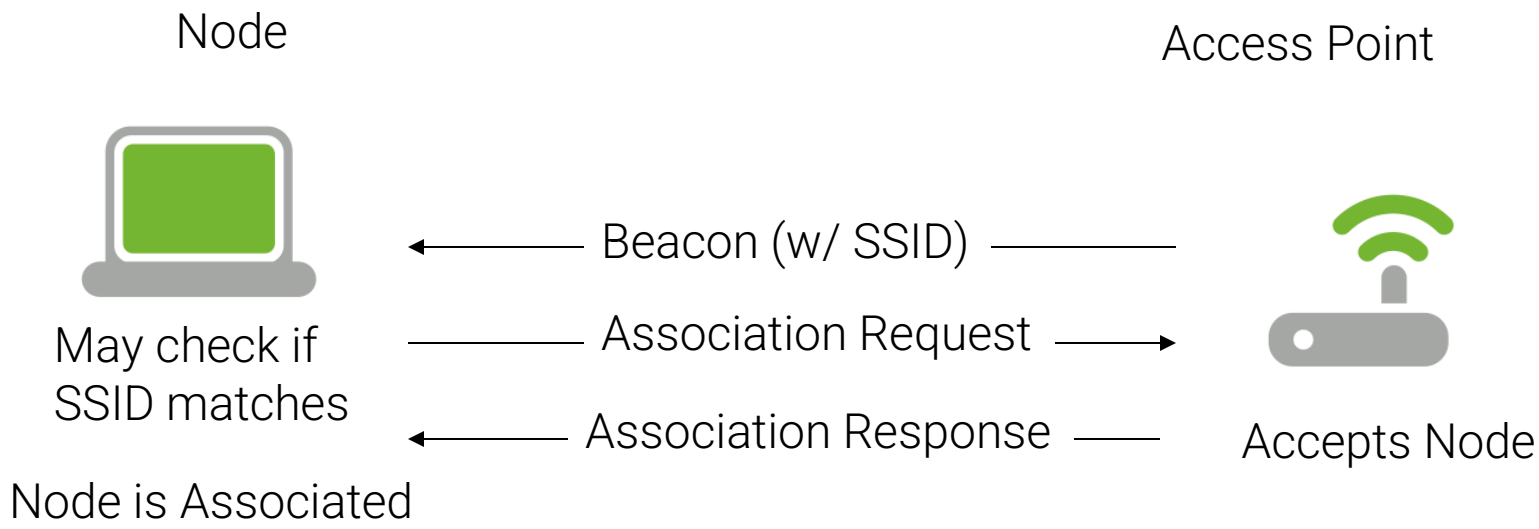
Type of Frames

- Control frames
  - RTS/CTS frames
  - ACK frames
- Data frames

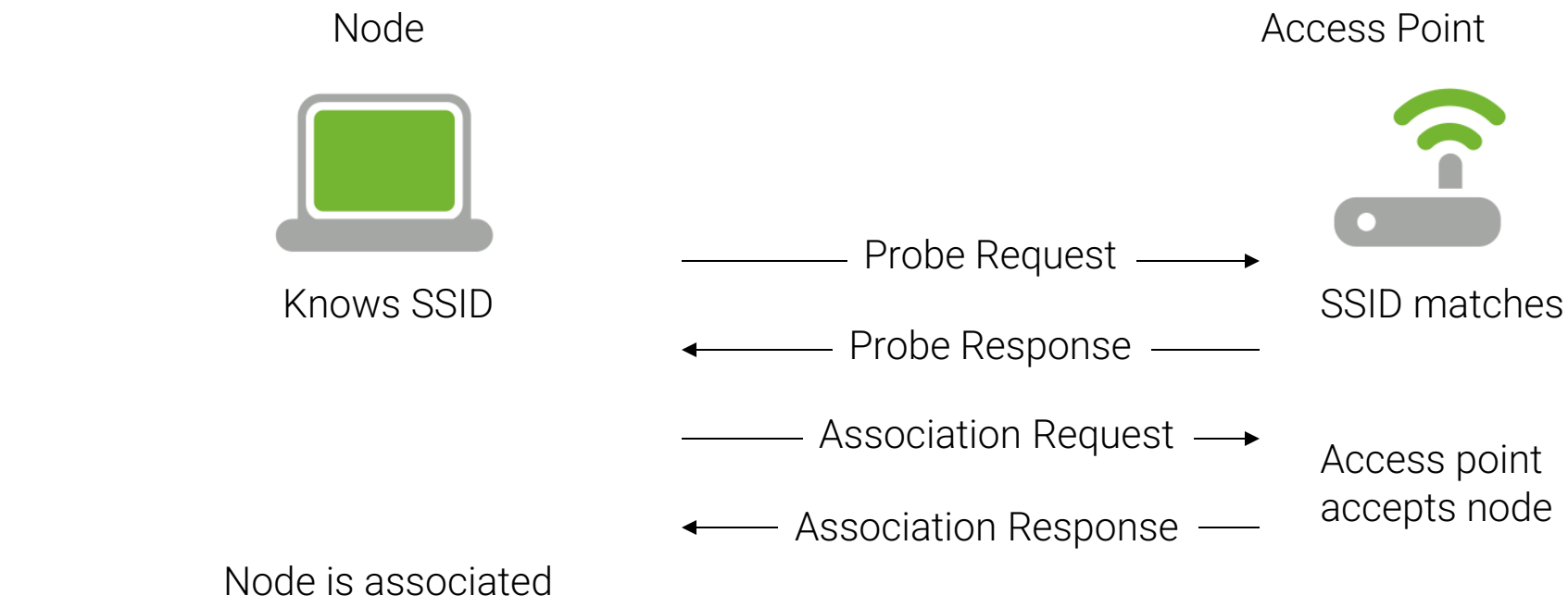
Association

- Clients (also called “stations”) associate with an access point (AP)
- Access points are identified by a service set identifier (SSID)
- When a mobile station enters the transmission range of one or more Aps, it will connect to the station with the strongest signal (and lowest observed error rate)
- Periodically it will scan the network for better APs
- MAC Access Control Lists can be used to regulate which stations can associate with an AP

Discovery - Open Network



Discovery - Closed Network



Wired Equivalent Privacy

- Wired Equivalent Privacy (WEP) is a series of mechanisms to provide security
- Encryption with shared-key can be used to encrypt traffic
  - RC4 with 40-bit or 104-bit static key, 24-bit IV in the clear
- This protocol is broken

WPA

- Wi-Fi Protected Access (WPA) was introduced to solve the security problems associated with WEP
  - Implements a subset of the 802.11i standard
  - Relies on TKIP (Temporal Key Integrity Protocol), which uses a per-packet key
    - Secret key and initialization vector for RC4 are composed in a complex way and not simply juxtaposed
    - Keys are routinely changed
    - A sequence number is added to prevent replay attacks
    - Introduces a message integrity check (Michael) that is better than WEP's CRC
- WPA supports authentication mechanisms, such as 802.1X - EAP/Radius
- WPA does not rely on additional hardware and therefore can be used on hardware that supports WEP

WPA2

- A number of vulnerabilities have been found in WPA
  - See for example: M. Vanhoef and F. Piessens “Practical Verification of WPA-TKIP Vulnerabilities,” in Proceedings of ASIACCS, 2013
- WPA/TKIP was deprecated by IEEE in 2009
- WPA2 is the evolution of WPA implements the full 802.11i standard
- Requires new hardware
- Uses AES instead of TKIP
- Supports both pre-shared key (PSK) and extended authorization (802.1X - EAP/RADIUS)

Attacks Against Wireless Networks

- Wireless traffic sniffing
- Wireless network detection
  - Closed networks
- Injection attacks
  - Denial of service
  - Man-in-the-middle
- WEP attacks
  - Stream reuse attacks
  - WEP traffic generation
  - Brute force attacks
  - FMS attack

Wireless Traffic Sniffing

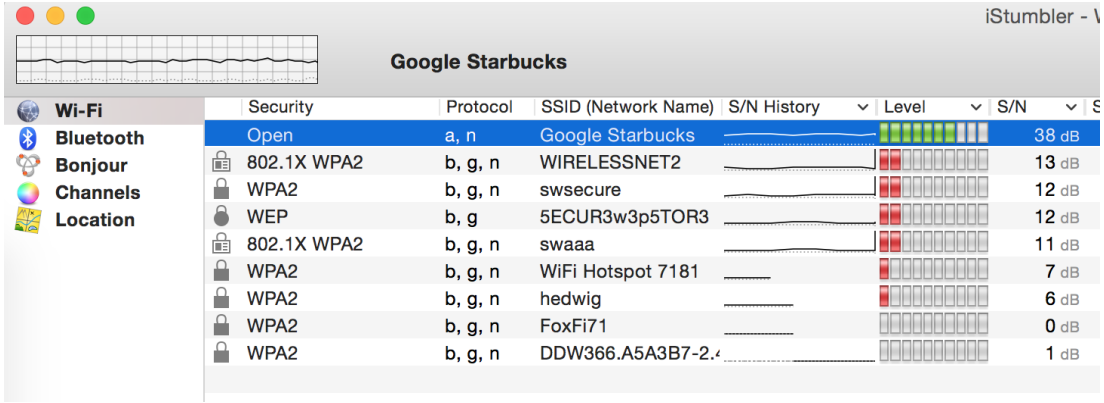
- Traffic in an open wireless network is particularly vulnerable to eavesdropping
  - No physical access is necessary (e.g., the “parking lot attack”)
- Even when using encryption, source and destination MAC addresses are accessible
  - May be used to bypass MAC access control lists at another AP, configured differently (e.g., no encryption)
  - May be used to perform traffic flow analysis

Monitor vs. Promiscuous

- In promiscuous mode, the 802.11 headers are removed and only the packets transmitted by the AP with whom the client is associated are passed on by the driver
- In order to observe all traffic (including management frames) the network card must be able to be in monitor mode (RFMON)
  - When in monitor mode, the card (usually) cannot send traffic

Wireless Network Detection

- Monitor Mode Protocol Analysis
  - Card set to monitor mode
  - Sniffs beacons and probes
  - Allows one to detect closed APs
  - Allows one to detect wireless nodes



Detecting Closed Networks

- Analyze probe responses from APs that are replies to probe requests from valid nodes
  - Probe requests are sent out roughly every 10 minutes
- Probe requests can be forced using a disassociation attack
  - Attacker sends a spoofed disassociation frame pretending to be the AP
  - The disassociated station tries to connect again and sends a probe message (containing the SSID)

Denial of Service

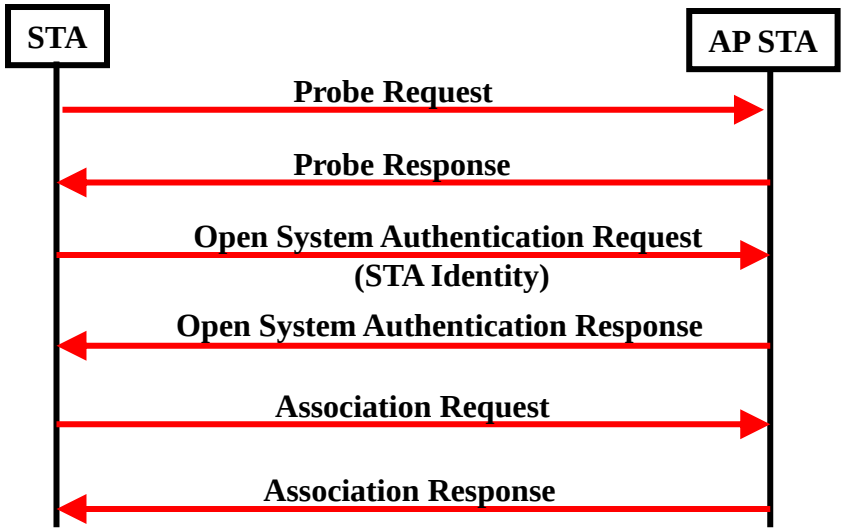
- Wireless networks are particularly vulnerable to denial-of-service attacks
- One can send a disassociation request to nodes on a wireless network and continue to send disassociation messages whenever they re-associate
- One can use radio interference to make the wireless network unusable

Man-in-the-Middle Attacks

- There are two main ways to perform a man-in-the-middle attacks:
  - By performing ARP spoofing once associated with an AP
  - By de-authorizing a victim host
    - Attacker becomes an AP with the same SSID on a different channel
    - Attacker de-authorizes victim
    - Victim reconnects to fake AP
    - Attacker relays frames back and forth

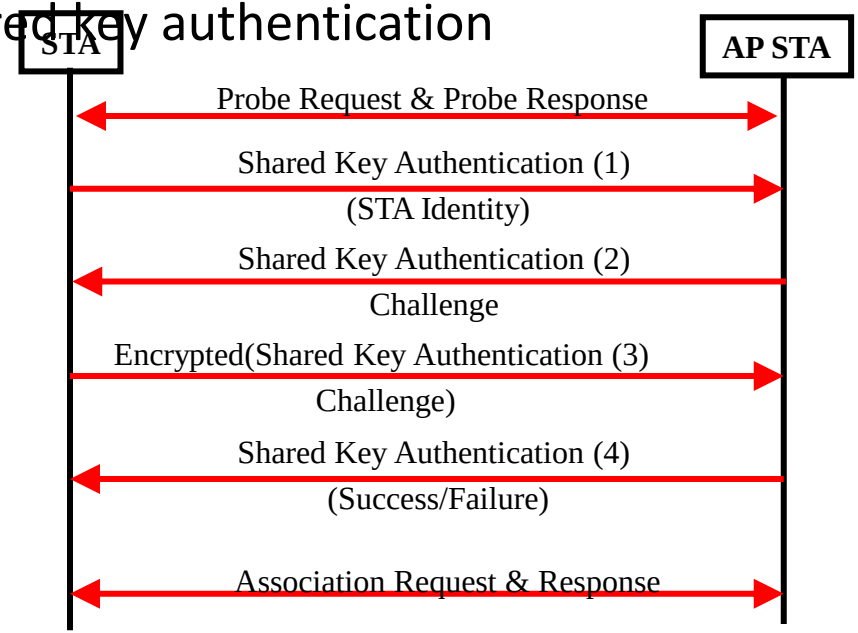
1.Open System Authentication

- Establishing the IEEE 802.11 association with no authentication



2. Wired Equivalent Privacy (WEP)

- WEP uses shared key authentication

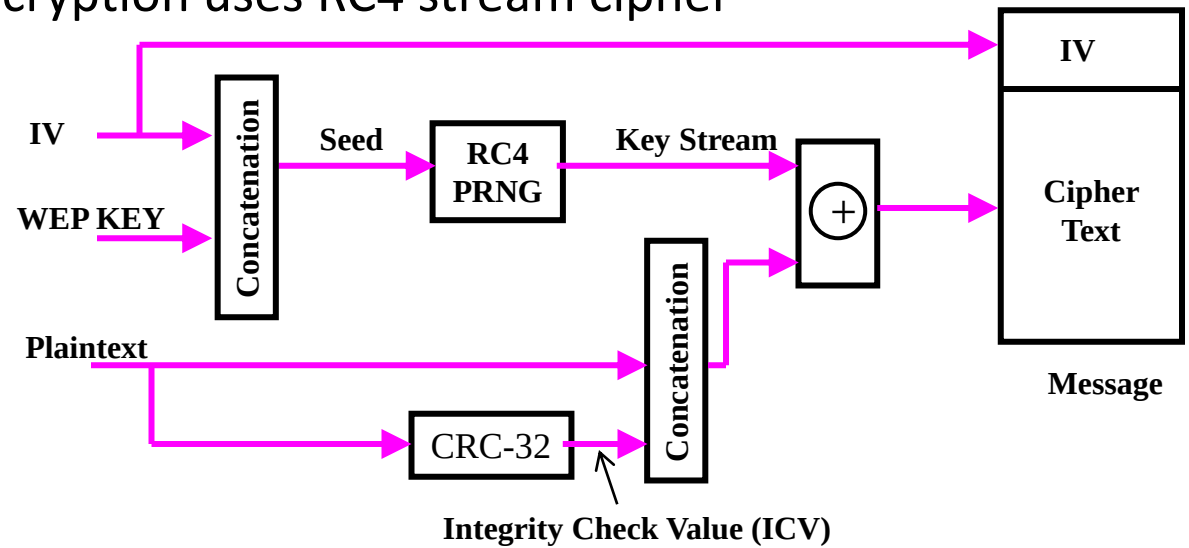


WEP Encryption

- A shared secret key K (40 or 104 bits) and a public IV (24 bits) are used to generate a stream of pseudo-random bits using RC4
- The message includes a CRC code computed on the plain text part of the message
- The stream is XORed with the message
- The message and the IV are sent to the receiver
  - A different IV is used every time
- At the receiving end, the IV and the shared key are used to generate the same bit stream, which is then XORed with the encrypted message

2. Wired Equivalent Privacy (WEP)

- WEP Encryption uses RC4 stream cipher



2. Wired Equivalent Privacy (WEP)

- Several major problems in WEP security
  - The IV used to produce the RC4 stream is only 24-bit long
    - The short IV field means that the same RC4 stream will be used to encrypt different texts – IV collision
    - Statistical attacks can be used to recover the plaintexts due to IV collision
  - The CRC-32 checksum can be easily manipulated to produce a valid integrity check value (ICV) for a false message

WEP Attacks

- Brute Force
  - Brute forcing a 40-bit key is feasible
- Key stream reuse
  - If a key stream is reused, it is possible to obtain the XOR of two plaintext messages
    - (M1 ^ Si) ^ (M2 ^ Si) = M1 ^ M2 ^ Si ^ Si = M1 ^ M2
  - If one of the messages is known, it is possible to derive the other or the key stream for the IV,K used
    - The Shared Key Authentication mechanism worked by providing a plaintext challenge that the client had to encrypt to prove that it knew the password
    - This allows one to collect some bits of that particular cypher stream
- Weak integrity check
  - Because the CRC is computed on the plaintext it is possible to modify the cipher text without breaking the CRC

WEP Attack Countermeasures

- Closed Mode
- MAC Filtering
- Use Weak IV Filtering Hardware
- Keys Rotation

Closed Mode

- Strengths
  - Doesn’t send out any beacon frames (more difficult to detect)
  - Requires connecting nodes to supply the correct SSID in order to associate
- Weaknesses
  - AP can still be detected through sniffing probe requests and responses

MAC Filtering

- Strengths
  - Allows one to control which MAC addresses can associate with an access point
- Weaknesses
  - MAC addresses can be easily spoofed
  - One can easily determine which MAC addresses are allowed by monitoring activity on the network
  - Attackers can still monitor communications
  - One can easily brute force MAC addresses

Weak IV Filtering Hardware

- Strengths
  - It requires much more time for an attacker to crack the WEP key
- Weaknesses
  - An attacker can still crack the WEP key given enough time

Change Keys Frequently

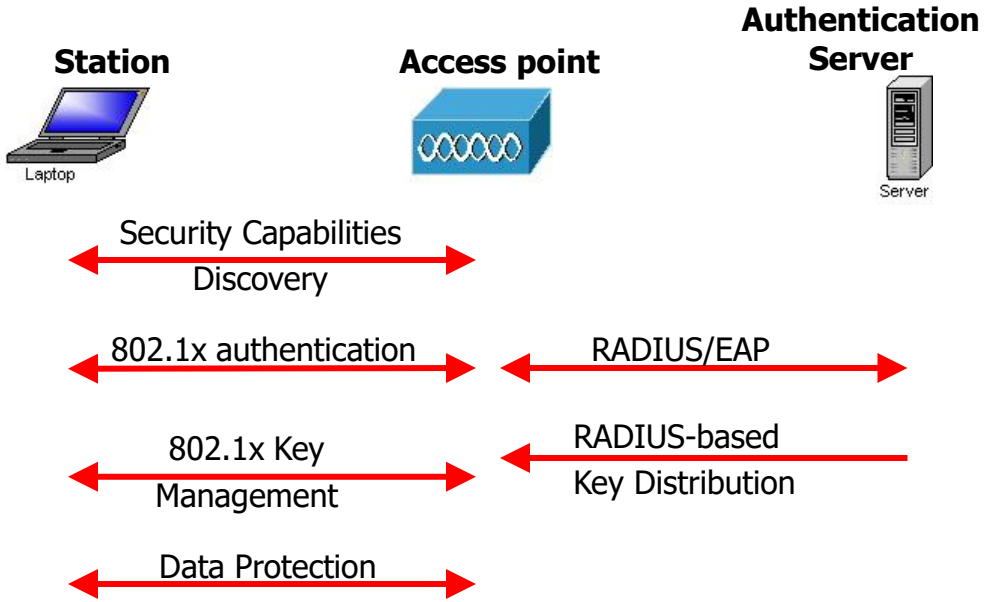
- Strengths
  - Makes it more difficult for attackers to crack WEP
  - Even if an attacker cracks a key, they will only have access for a limited amount of time
- Weaknesses
  - With traffic generation, an attacker can crack a key in a reasonable amount of time
  - Attackers can still decrypt collected traffic using cracked keys

3. Robust Security Network (RSN)

- 802.11i defines a set of features to establish a RSN association (RSNA) between stations (STAs)
  - Enhanced data encapsulation mechanism
    - CCMP
    - Optional: TKIP
  - Key management and establishment
    - Four-way handshake and group-key handshake
  - Enhanced authentication mechanism for STAs
    - Pre-shared key (PSK); IEEE 802.1x/EAP methods

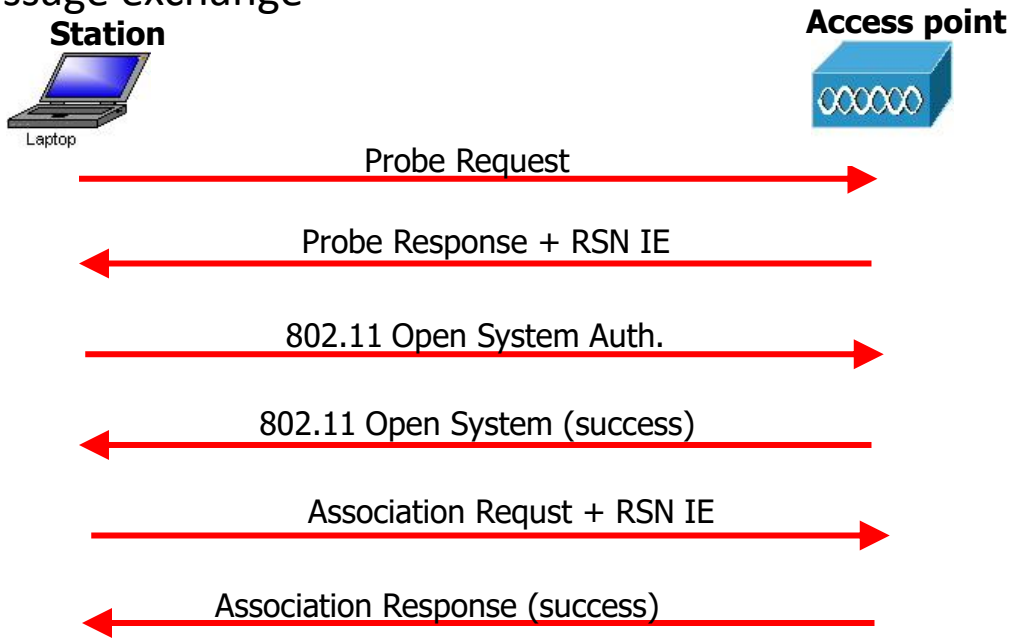
3. Robust Security Network (RSN)

- Operational phases



3. Robust Security Network (RSN)

- Discovery message exchange



3. Robust Security Network (RSN)

- Authentication
  - Mutual authentication
  - The AS and station derive a Master Key (MK)
  - A Pairwise Master Key (PMK) is derived from MK
  - The AS distributed PMK to the AP
  - In PSK authentication, the authentication phase is skipped
    - PMK = PSK

3. Robust Security Network (RSN)

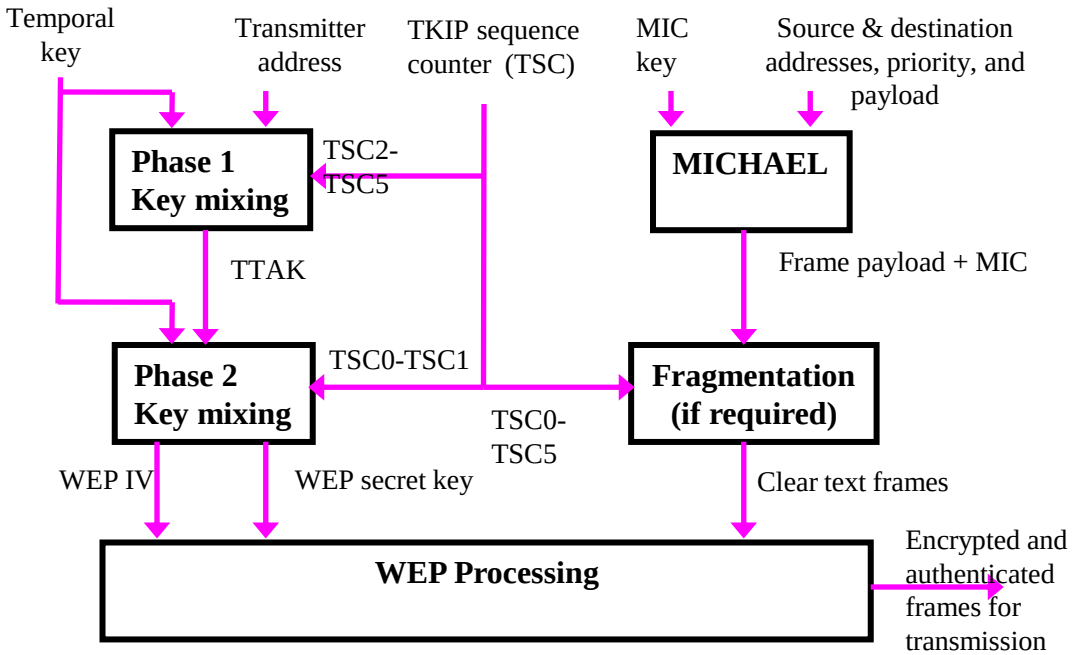
- Key management and establishment
  - PMK is sent to AP by AS
  - Key management is performed between AP and the peer – four-way handshake
    - The four-way handshake can also be used for mutual authentication between AP and the peer in PSK mode
  - A set of keys are derived from PMK to protect group key exchange and data
  - Group key exchange allows AP to distribute group key (for multicast) to the peer

4. Temporal Key Integrity Protocol (TKIP)

- Optional IEEE802.11i protocol for data confidentiality and integrity
  - TKIP is designed explicitly for implementation on WEP legacy hardware
- TKIP three new features:
  - A cryptographic message integrity code (MIC)
  - A new IV sequencing discipline
    - The transmitter increments the sequence number with each packet it sends
  - A per-packet key mixing function

4. Temporal Key Integrity Protocol (TKIP)

- TKIP frame processing



4. Temporal Key Integrity Protocol (TKIP)

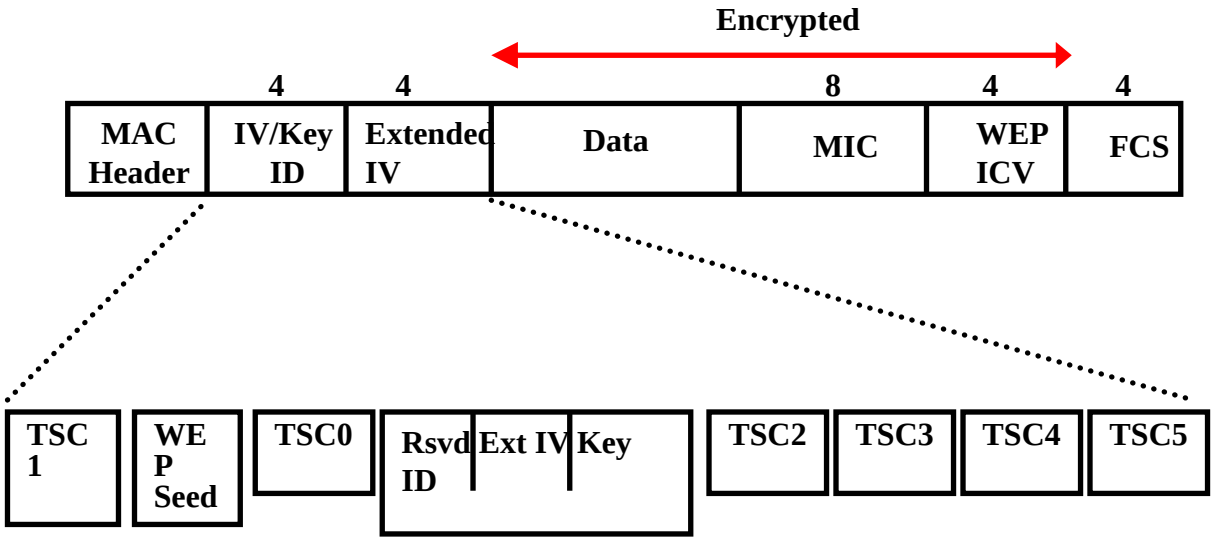
- Defeating weak key attacks: key mixing
  - Transforms a temporal key and packet sequence number into a per packet key and IV
  - The key mixing function operates in two phases
    - Phase 1: Different keys used by different links
      - Phase 1 needs to be recomputed only once every 2^{16} frames
    - Phase 2: Different WEP key and IV per packet
  - Phases 1 and 2 can be pre-computed

3. Temporal Key Integrity Protocol (TKIP)

- Defeating replays: IV sequence enforcement
  - TKIP uses the IV field as a packet sequence number
  - The transmitter increments the sequence number with each packet it send
  - A packet will be discarded if it arrives out of order
    - A packet is out-of-order if its IV is the same or smaller than a previous correctly received packet
- Defeating forgeries: New MIC (Michael)
  - MIC key is 64-bits
    - security level of 20 bits

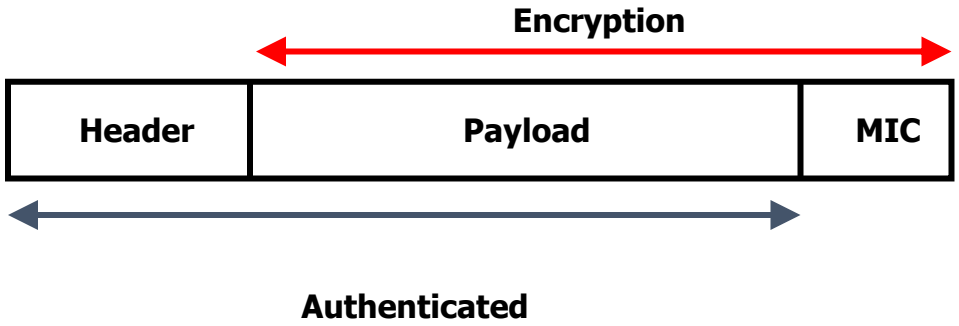
4. Temporal Key Integrity Protocol (TKIP)

- TKIP encapsulation



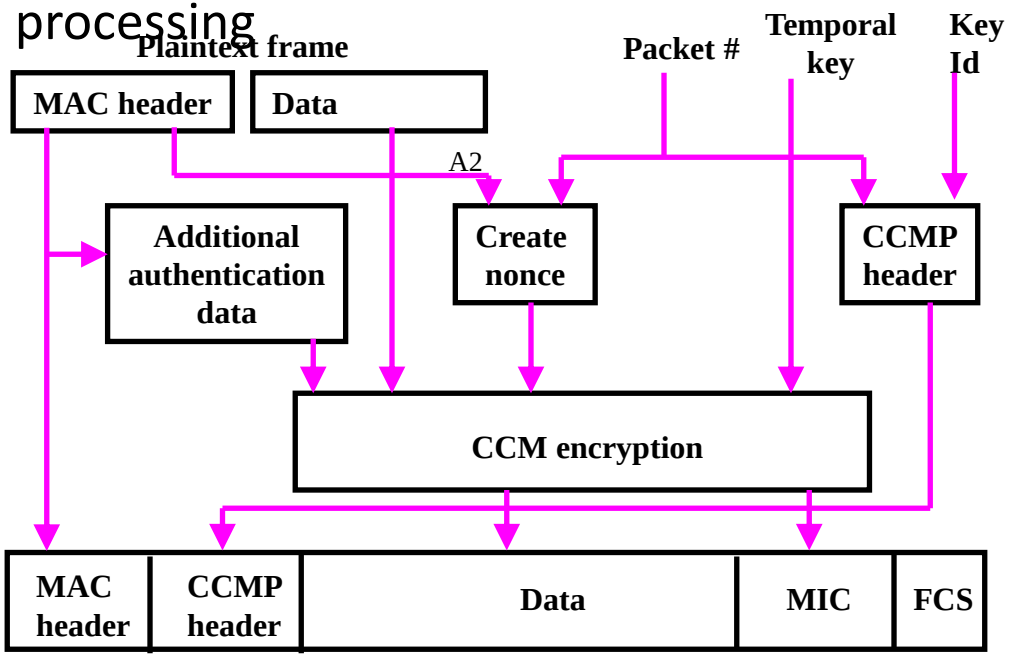
5. Counter Mode with CBC-MAC (CCMP)

- Both encryption and MIC use AES
  - Uses counter Mode (CTR) to encrypt the payload and MIC
  - Uses CBC-MAC to compute a MIC on the plaintext header and the payload
  - Both encryption and authentication use the same key



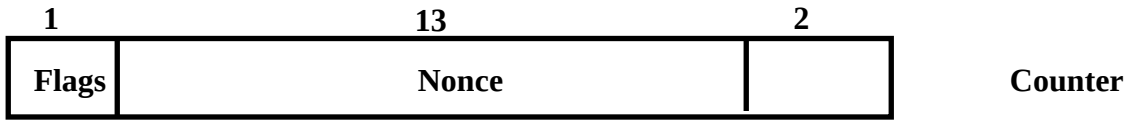
5. Counter Mode with CBC-MAC (CCMP)

- CCMP data processing

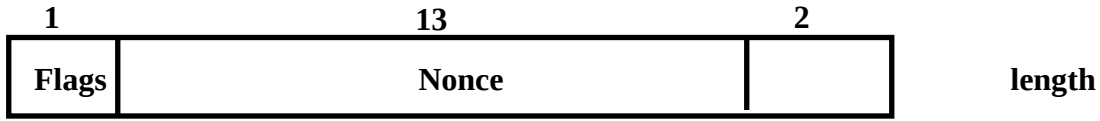


5. Counter Mode with CBC-MAC (CCMP)

- Each message block has the size of 16 octets
  - For CTR encryption, A_i has the following format (i is the value of the counter field):

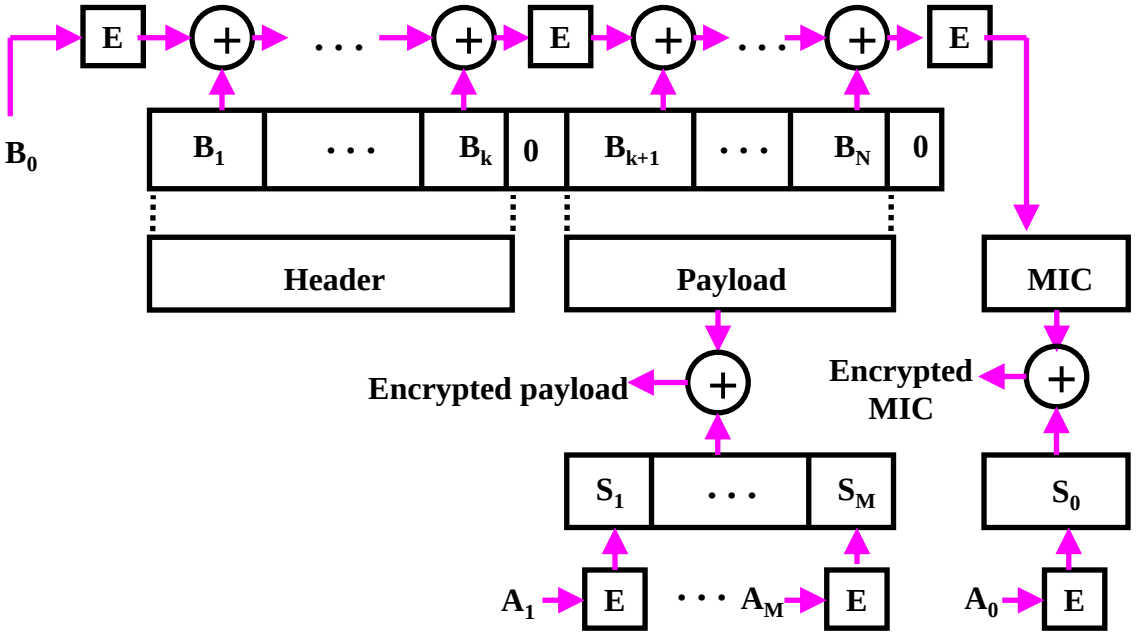


  - For the CBC-MAC authentication, B_0 has the following format (length := size of the payload):



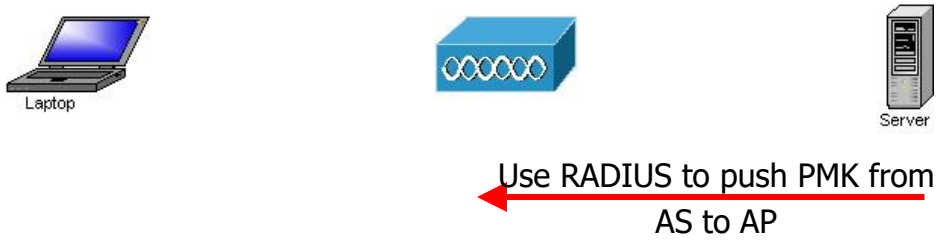
5. Counter Mode with CBC-MAC (CCMP)

- CCM encryption

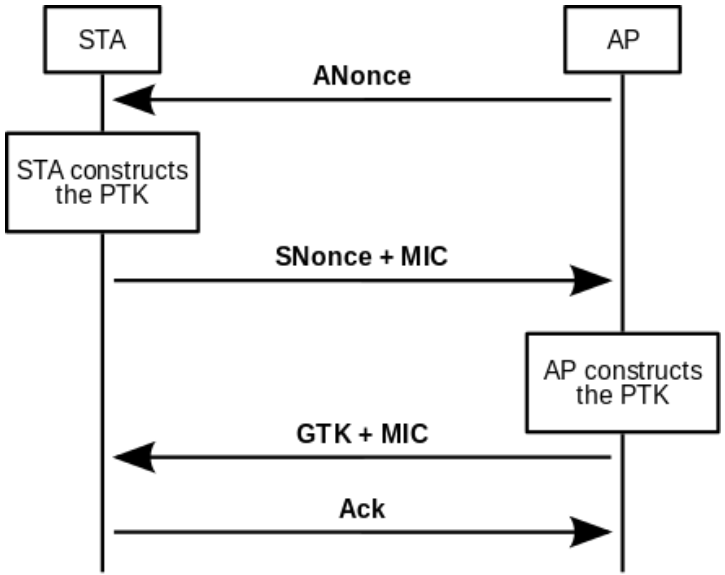


6. Key Management and Establishment

- 802.1x key management



WPA2 4-way Exchange

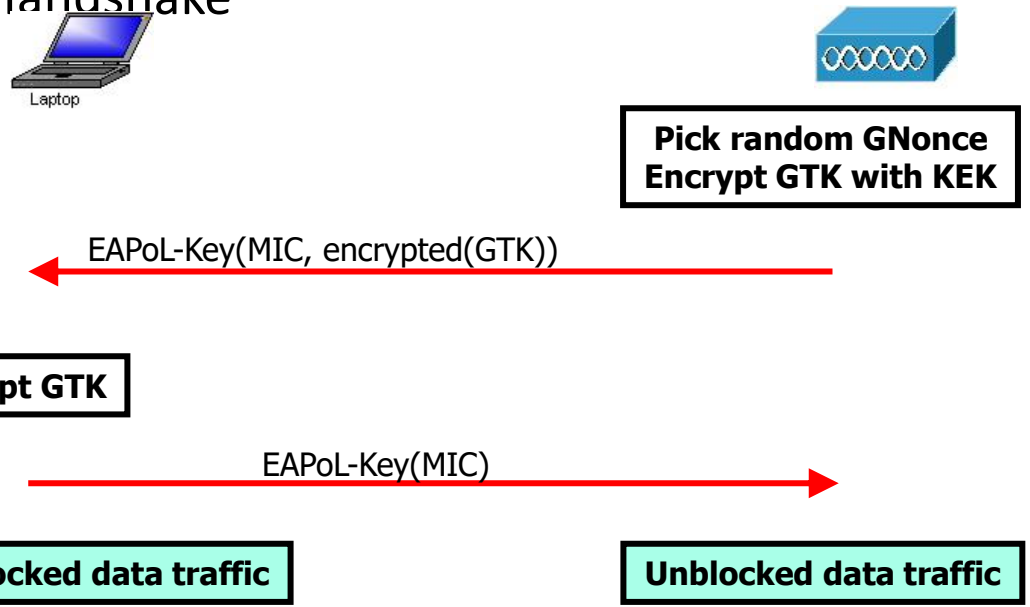


1. The AP sends to the client an Anonce
2. The Client derives the Pairwise Transient Key (PTK) from Anonce, Snonce, and PMK and sends the Snonce protected by a Message Integrity Code (MIC) calculated using the PTK
3. AP also calculates the PTK and verifies the MIC. Then the AP sends the Group Transient Key (GTK)
4. The Client receives the GTK and sends an acknowledgment

https://en.wikipedia.org/wiki/IEEE_802.11i-2004

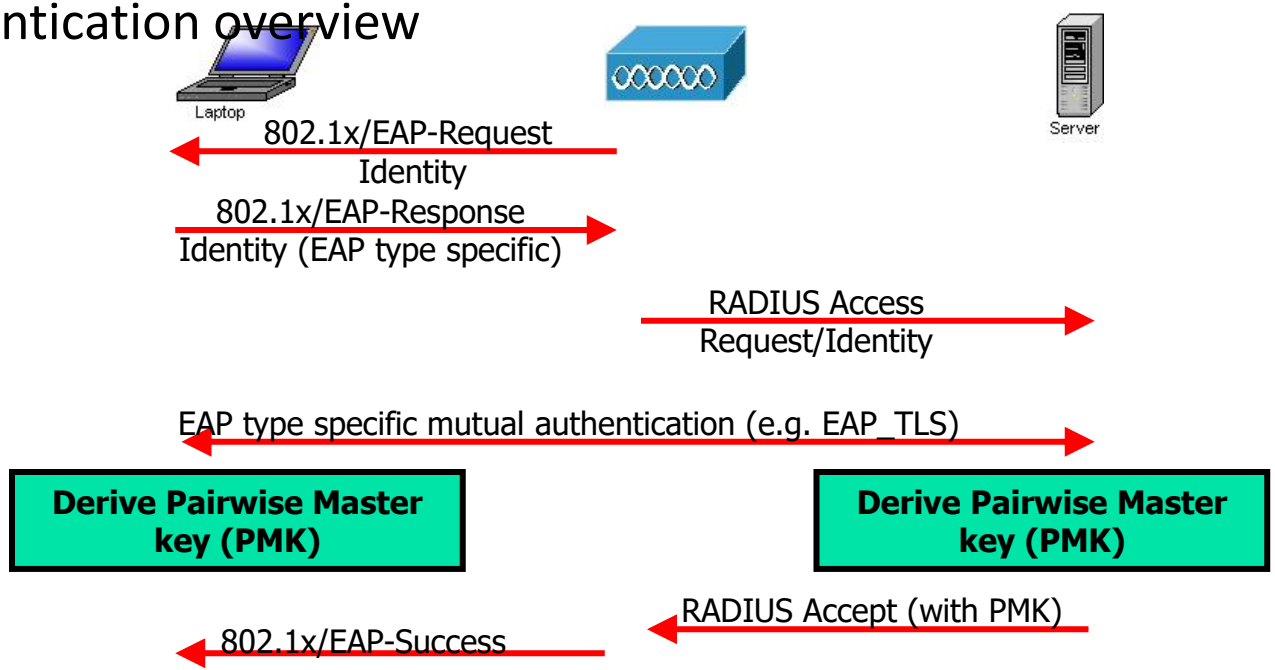
6. Key Management and Establishment

- Group Key Handshake



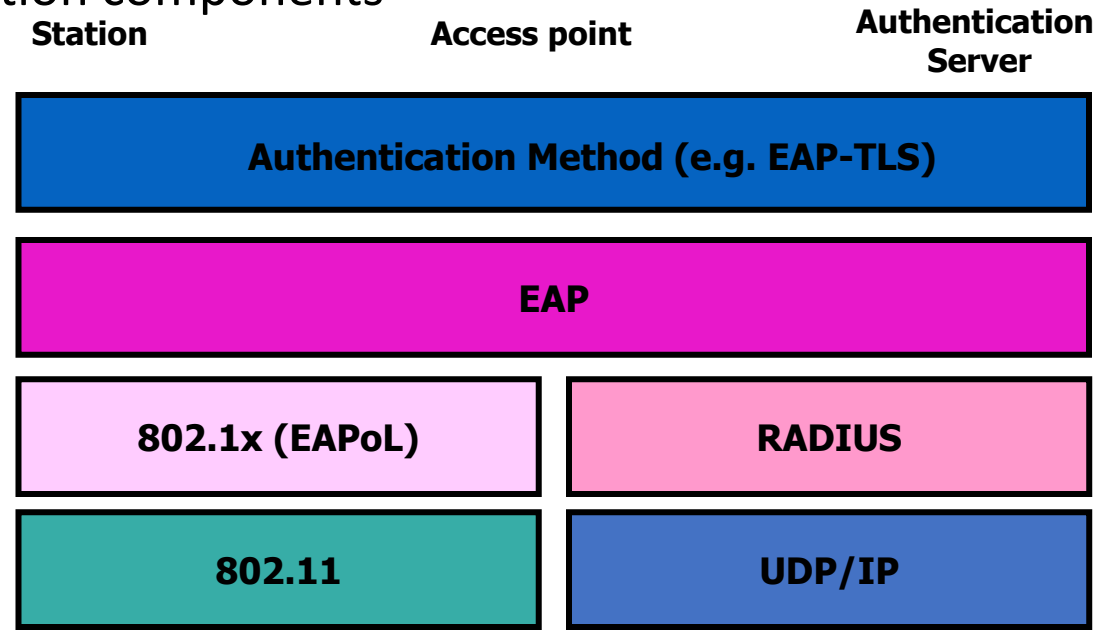
7. Authentication protocols

- Authentication overview



7. Authentication Protocols

- Authentication components

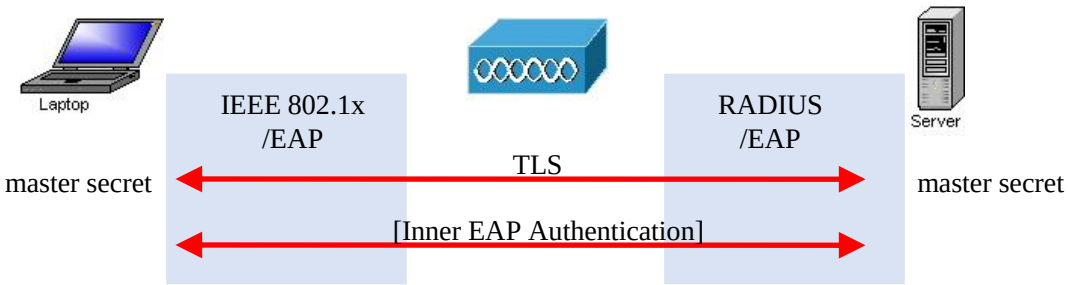


7. Authentication Protocols

- LEAP
  - Simple – neither server certificate or peer certificates is required
  - CHAP is used for mutual authentication
    - The user's password is the shared secret
  - Session key is derived from the shared secret , the challenges and the challenge responses
  - Susceptible to the dictionary attack

7. Authentication Protocols

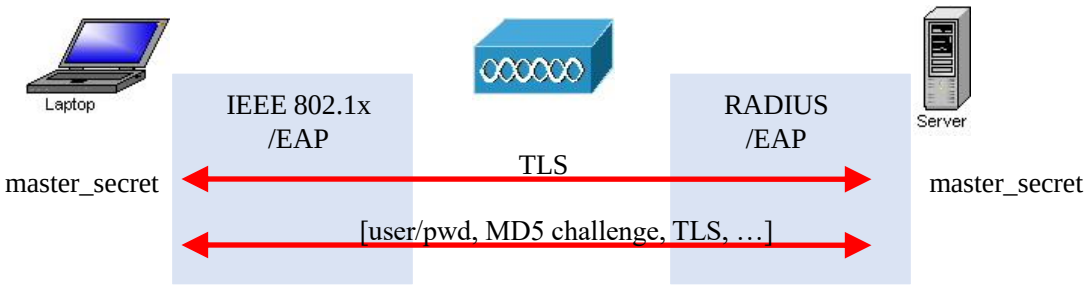
- EAP authentication: general approach
  - Used TLS to setup a secure tunnel
  - Inner authentication method is used for further authentication



PMK = function of (nonces, {DH secret/session key})

7. Authentication Protocols

- EAP-TLS
  - Both peer and AS authenticate each other using certificates in the TLS phase
  - Inner authentication may be used for user authentication



master_secret = PRF(pre_master_secret, " master secret", nonces)
PMK = PRF(master_secret, "client EAP encryption", nonces)

7. Authentication Protocols

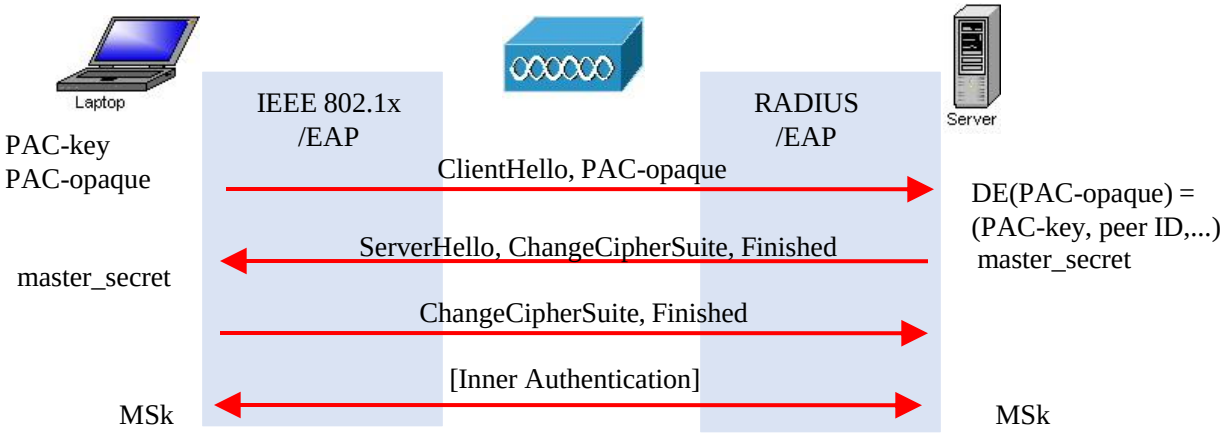
- PEAP
  - At the TLS phase, server is authenticated based on the server's certificate – no peer authentication
  - Peer authentication is done at the inner authentication
    - EAP-MS-CHAPV2 is the most popular inner authentication method – it provides mutual authentication plus key generation
  - The PMK generated is based on both the TLS master_secret and the master_session_key (MSK)

7. Authentication Protocols

- EAP-FAST
  - Two methods for setting up TLS tunnel
    - Server certificate
    - Protected Access Credential (PAC)
  - PAC components:
    - Shared secret – used to derive TLS master secret
    - opaque element – presented by the peer to the AS
      - Contains shared secret and peer identity
      - Protected with cryptographic keys and algorithm
    - other information – identity of the PAC issuer, secret lifetime …

7. Authentication Protocols

- TLS tunnel using PAC



master_secret = PRF(PAC-key, “PAC to master secret label hash”, nonces)
PMK = function of (master_secret, MSK)

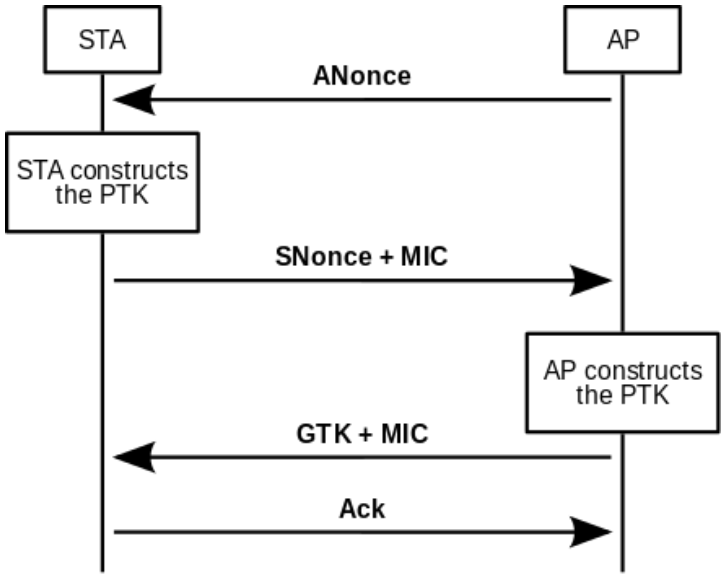
WPA2 Attacks

- WPA2 does not have known serious vulnerability
- It is still open to a password guessing attack
- The attacker de-authenticates a connected user
- The user re-authenticates using a four-step handshake containing the encrypted password
- The four handshake is recorded and used offline to try a number of passwords (dictionaries, random permutation, etc.)
- Under certain conditions, the protocol is vulnerable to a key reinstallation attack

WPA2 4-way Exchange

- Both Client and AP know a Pairwise Master Key (PMK), derived by the PSK (Pre-Shared Key)
- Note that if the network is using 802.1X, the PMK is based on the data associated with the user authentication process and the PSK, otherwise the PMK is the PSK
- The exchange is protected using the PMK
- If one knows the Pre-Shared Key and observes the 4-way handshake he/she will be able to decrypt the traffic between the client and the access point

WPA2 4-way Exchange



1. The AP sends to the client an Anonce
2. The Client derives the Pairwise Transient Key (PTK) from Anonce, Snonce, and PMK and sends the Snonce protected by a Message Integrity Code (MIC) calculated using the PTK
3. AP also calculates the PTK and verifies the MIC. Then the AP sends the Group Transient Key (GTK)
4. The Client receives the GTK and sends an acknowledgment

https://en.wikipedia.org/wiki/IEEE_802.11i-2004

The KRACK Attack

- An attacker “clones” the AP (MAC address need to be the same) on a different channel
- If the attacker’s signal is stronger than the original AP, the client will connect to the cloned AP
- The attacker acts as a man-in-the-middle forwarding the first three steps of the 4-step handshake, but blocking the 4th step
- At this point the client resets its nonce counter, and starts sending traffic

The KRACK Attack

- The original AP does not see the 4th message, and re-sends the third message of the protocol
- The attacker intercepts this message and forwards it to the client
- The client accepts this message and re-resets its counter, causing the reuse of specific nonces

- See: “Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2” in Proceeding of ACM CCS, 2017

Summary

- Networks are used to exchange data between nodes
- It is important to understand what can and cannot be trusted
- Attack building blocks
  - Sniffing
  - Spoofing
  - Hijacking
  - Denial-of-service
  - Brute-forcing
- Tools
- Countermeasures

Appendix: VxWorks

- https://www.armis.com/urgent11/