ECE 435 – Network Engineering
Lecture 21

Vince Weaver

https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

12:30pm, Barrows 125

2 April 2024

Announcements

- HW#9 (Ethernet) will be posted
- Don't forget project status update due April 12th (next Friday)

Project Status

- Summary of topic
- What HW/SW you'll be using
- On track to finish
- Say if you're willing to present early (Early Thursday, Tuesday vs Thursday)
- Monday office hours cancelled (eclipse)

XZ Vulnerability (Current Events)

- In the news. Computer security but relate to stuff in this class.
- Problem with open source software and trust. Can you trust random contributors? Can you trust random tools? Reflections on Trusting Trust.
- ssh vulnerable. Maybe only thing you let through incoming to your firewall
- What if someone pretened they had a bug fix but instead it introduced evil code?

- Changeset that is like if (!strncmp(passwd,"mypassword"))
- In theory easy to spot, lots of reviewers, benefit of open source everyone can see in the open, run tools
- What if try to be sneaky about it?
- if (uid=0) printf("Error!\n");
- Underhanded C contest
- U of Minnesota Linux kernel incident. IRB. No justice.
- XZ issue. Problem not with Linux, or ssh. Was with xz compression library.
- Maintainer volunteer, overworked, someone showed up offered to help, gradually gained trust. Eventually given

commit privileges.

- They started making seemingly innocent changes
- Broke some of the hardening tests in autuoconf, including adding a hard to see "." so the test for it would always fail
- No actual code added to C code, but part of build process it would take some of the files from the test suite and patch the binary
  - original calls `__get_cpuid` at library start to see if can use CLMUL instruction
  - adds a `_get_cpuid` with one underscore to do sneaky

stuff

- systemd on Linux links against this. The library loaded by systemd before launching sshd. Would override some symbols (complicated linker stuff) but would override RSA key checking
- When the certificate came in with the connection, if it decompressed with the key from attacker then treat it as a binary and run it
- This is bypassing everything, would be really hard to detect in audit.
- How was it noticed? postgres guy was benchmarking

using perf and noticed ssh connections taking 10x as long, tracked it down. Luck.

- Attacker playing long game. Years to do things. Apparently had other accounts that were doing things like patching fuzzer/security tools to try to ignore this, also bugging stable distributions to update to newest version
- Only really got to the point of testing in Linux distros (debian unstable)
- Who responsible? They were careful to make it look like from China, some analysis of timezones maybe it was

eastern-europe/middle east timezone

- Cuckoo's Egg by Cliff Stoll

ARP – address resolution protocol

- On local network, how do we find MAC address if we know IP?
- Hard-code mapping `/etc/ethers`?
- Can it be automatically determined somehow?
- ARP – address resolution protocol (IPv4)
- ND – Neighborhood Discovery (IPv6)

ARP (RFC826)

- Device first checks ARP cache to see if already knows
- Otherwise, broadcasts to ff:ff:ff:ff:ff:ff “who has this IP”
- Device reply with its IP and MAC (unicast)
- These are cached
- Timeout in case you reassign
- ARP announcement: can broadcast when your address changes so they can update (gratuitous ARP)
- Other optimizations(?)

IPV6: Neighborhood Discovery Protocol

- TODO, copy this over from IPv6 notes

ARP Security

- Can you spoof ARP responses to get frames meant for a different device?

RARP/BOOTP

- Some cases need to do RARP (Reverse ARP) (RFC 903) have own MAC, find IP (netbooting is common reason)
- ARP packets not forwarded, so extension called BOOTP that allowed network booting.
- BOOTP automated by DHCP.
- IPv6 has IND (Inverse Neighborhood Discovery Protocol)

Ethernet Transmission (Review)

- Break data into frame
- In half-duplex CSMDA/CD senses carrier. Waits until channel clear
- Wait for an inter-frame-gap (IFG) 96 bit times. Allows time for receiver to finish processing
- Start transmitting frame
- In half-duplex, transmitter should check for collision.
  Co-ax, higher voltage than normal
  For twisted pair, noticing signal on the receive while

transmitting

- If no collision, then done
- If collision detected, a *jam* signal is sent for 32-bits to ensure everyone knows. Pattern is unspecified (can continue w data, or send alternating 1s and 0s)
- Abort the transmission
- Try 16 times. If can't, give up
- Exponential backoff. Randomly choose time from 0 to $2^k - 1$ where k is number of tries (capping at 10). Time slot is 512 bits for 10/100, 4096 for 1Gbs
- Wait the backoff time then retry

Ethernet Receiving (Review)

- Physical layer receives it, recording bits until signal done. Truncated to nearest byte.
- If too short (less than 512 bits) treated as collision
- If destination is not the receiver, drop it
- If frame too long, dropped and error recorded
- If incorrect FCS, dropped and error recorded
- If frame not an integer number of octets dropped and error recorded
- If everything OK, de-capsulated and passed up

- Frame passed up (minus preamble, SFD, and often crc)
- Promiscuous mode?

Maximum Frame Rate

- 7+1 byte preamble 64-byte frame, IFG of 12 bytes between transmissions. equals 672 bits. In 100Mbps system 148,800 frames/second

Ethernet Flow Control

- Flow control is optional
- In half duplex a receiver can transmit a “false carrier” of 1010..10 until it can take more.
- Congested receiver can also force a collision, causing a backoff and resend. Sometimes called force collision
- Above schemes called “back pressure”
- For full duplex can send a PAUSE frame that specifies how much time to wait.

Full Duplex MAC (requires switch)

- Early Ethernet was coaxial in a bus
- Twisted pair has replaced this, usually in a hub/or switch star topology
- 10BASE-T and 100BASE-TX pair for transmit or receive
- inefficient. Since point to point, why do you need arbitration?
- Full-duplex introduced in 1997. Must be able to transmit/receive w/o interference, and be point to point.
- Full duplex effectively doubles how much bandwidth

between. Also it lifts the distance limit imposed by collision detection

Router vs Hub vs switch

- Hub all frames are broadcast to all others
  Bandwidth is shared (only say 100MB for all)
- Switch – direct connection, no broadcast. Has to be intelligent. Each point to point connection full bandwidth.
  no collisions. Internally either own network to handle collisions, or else just buffer RAM that can hold onto frames until the coast is clear.
- Multi-speed hubs?

When 10/100MB first came out, cheap hubs could only run at 10MB or 100MB. But switches *really* expensive. They had a compromise 10/100MB hub that internally had a hub for both then a mini-switch to bridge the gap.

- Router will move frames from one network to another
- Lights. How many ports? Uplink ports?

Direct Connection Ethernet

- Direct connect two machines with one cable
- Used to need special “crossover” cable to swap TX and RX lines
- Modern cards can detect direct connect and swap the wires for you

Ethernet Security

- Traditional hub, all machines saw all packets
- With tcpdump could monitor all packets on network, back in day all plain text. e-mail, web-browsing, chat, passwords, telnet
- tcpdump put card in “promiscuous” mode which let it intercept all packets instead of ignoring ones not to system
- Why so low security? Old day trust people at your work/office, also was probably expensive/difficult to get

an unauthorized UNIX workstation with ethernet card and root access on the local network

- Lights on hub

Power over Ethernet

- Method B: In 10/100 Base T, only 2 of the 4 pairs in Cat5 used. So send voltage down spare pairs
- Method A: send DC voltage down with the signals floating on top
- Standards
  - Original POE: 44 VDC, 15.4W
  - POE+ 25W

Power over Ethernet Switch

- Need special switch to send power, and device on other end has to support it.
- Takes time to negotiate power, so can take many seconds (or more) for full power available to device
- Raspberry Pi hat, used on new Pi4 cluster of mine

Classic 10Mbps is too slow!

- 10Mbps not fast enough! What can we do?
- FDDI and Fibrechannel (fast optic-ring), too expensive
- Can we just multiply all speeds by 10? Or else come up with some completely new better thing?
- IEEE decided to just keep everything same, just faster 802.3u 1995
- The other group went off and made 802.12 100BaseVG (which failed)

“Fast” Ethernet (100Mbps)

- 100BASE-TX most common
- BASE means baseband, that it's not modulated on a carrier
- Bit time from 100nsec to 10nsec
- Uses twisted pair/switches, no coax
    - To use cat3 100BASE-T4 wiring needed 4 twisted pair and complex encoding, no Manchester, ternary
    - To use cat5 wiring 100BASE-TX. Two twisted pair, one to hub, one from.

- The pairs are differential pairs, like USB

“Fast” Ethernet (100Mbps)

- Often split between MAC (media access controller) and PHY (physical interface).
- Card configures the PHY via the MII (media independent interface)
- Goal was you could have same card but interchangeable PHY (twisted pair, fiber, etc)

MII Interface

- 4bit bus
- Interface requires 18 signals, only two can be shared if multiple PHY
- So RMII (reduced) was designed. Clock doubled, only 2-bit bus. Fewer signal wires.

100BASE-TX

- 2 pairs inside cat5 cable
- One pair 100MB each direction, full duplex
- 100m distance
- Raw bits (4 bits wide at 25MHz at MII) go through 4B/5B encoding clocked at 125MHz
- 4B/5B means 4 bits encoded as 5, this allows at least two transitions per 5 bits, allows special patterns for end frame or error
- TX then goes through MLT-3 encoding

MLT-3 Encoding

- MLT-3 has three voltages, -1, 0, +1
- Cycles through them -1, 0, +1, 0, repeats
- For a 1, it transitions to next, for a 0, no transition
- 31.25MHz, much like copper version of FDDI (CDDI/TP-PMD)
- Needs encoding that limits consecutive zeros

```
MLT-3
  1  0  1  1  0  0  1
__    !        !  !   __
  |  !        !  !  |
  ---!--  ----!--!--
```

! |_| !   !

4B/5B Encoding

- At least two transitions per 5-bit code
- Provides DC equalization and spectrum shaping (what does that mean? This is cut-and-pasted a lot)

4B/5B Encoding (layout)

hex	4B	5B
0	0000	11110
1	0001	01001
2	0010	10100
3	0011	10101
4	0100	01010
5	0101	01011
6	0110	01110
7	0111	01111
8	1000	10010
9	1001	10011
A	1010	10110
B	1011	10111
C	1100	11010
D	1101	11011
E	1110	11100
F	1111	11101

Control Char	5B	meaning
H	00100	halt
I	11111	idle
J	11000	start #1
K	10001	Start #2
L	00110	Start #3
Q	00000	Quiet
R	00111	Reset
S	11001	Set
T	01101	End (terminate)

Gigabit Ethernet

- Two task forces working in 1998/1999
- 802.3z 1998 (fiber), 802.3ab 1999 (copper)
- Could still use hub, problem was the CSMA/CD restriction.
  - About 200m for 100Mbps.
  - For Gb would have been 20m which is not very far.
  - Carrier extension: hardware transparently pads frames to 512 bytes
    Wasteful, 512 bytes to send 64 bytes of data

    - Frame bursting: allow sender to sends sequence of multiple frames grouped together
- Better solution is just use full duplex
- 1000Base-SX (fiber)/LX (fiber)/CX (shielded)/T (cat 5), more

Gigabit Ethernet – Fiber

- No Manchester, 8B/10B encoding.
- Chosen so no more than four identical bits in row,
- No more than six 0s or six 1s
- need transitions to keep in sync

Gigabit Copper – 1000BASE-T

- PAM-5, 5 voltage levels, 00, 01, 10, 11, or control. So 8 bits per clock cycle per pair,
- 4 pairs running at 125MHz, 2 bits per clock(?), so 1GBps
- simultaneous transmission in both directions with adaptive equalization (using DSPs)
  - Actually can send and receive at same time on one pair
  - Basically, when you send the values add up
  - However if you send 1 1 0 1 an you notice on the line

0 2 0 2 you can figure out the other side sent -1 1 0 1

  ◦ Slightly tricky because voltage loss on line

- 5-level pulse-level modulation (PAM-5) [technically 100BASE-TX is PAM-3]. Diagram? looks sort of like a sine wave as cycle through the voltages.
- four-dimensional trellis coded modulation (TCM) 6dB coding gain across the four pairs
  Trellis coding also provides error correction
- Autonegotiation of speed. Only uses two pairs for this, can be trouble if pairs missing.

Gigabit Ethernet – Other

- Try to balance 0s and 1s? keep DC component low so can pass through transformers?
- Fast enough that computers at time had trouble saturating such a connection
- Jumbo Frames? 9000 byte?

Even Faster Ethernet

`http://www.theregister.co.uk/2017/02/06/decoding_25gb_ethernet_and_beyond/`

- Misquote: Not sure what the network will be like in 30 years, but they will call it Ethernet.

10Gb Ethernet

- 10Gb: 802.3ae-2002. Full duplex, switches only
- Need Cat6a or Cat7 for links up to 100m
- Expensive. Lots of kinds.
- 10GBASE-T, 802.3an-2006 100m over cat6a, 55m Cat6
- additional encoding overhead, higher latency
- Tomlinson-Harashin precoding (THP), PAM15 in two-dimensional checkerboard DSQ128 at 800Msymbol/s

2.5Gb Ethernet

- 2.5Gb: 802.3bz (Sep 2016?)
- Like 10Gb but slower. Can't run 10Gb over Cat5e

- Power over Ethernet (for using on wireless access points)

- Power with signal overlaid on top.

- 2.5Gb on Cat 5e, 5Gb on Cat6

25Gb Ethernet

- 25Gb, 802.3by. 25GBASE-T, 50GBASE-T. Available, if copper only a few meters
- Introduced as 10GB not fast enough, but 40GB too expensive
- SFP28 transceivers, both optical and copper

40Gb Ethernet

- 40GBASE-T twisted pair 40GBit/s 30m. QFSP+ connectors, like Infiniband
- Terabit? still under discussion
- 802.3ba – 2010 – 1m backplane, 100m multi-mode fiber, 10km single-mode fiber
- 802.3bg –
- 802.3bq – 2016 – 4-pair balanced twisted pair 30m
- QSFP+ transceivers, quad small form-factor pluggable, four 10GB lanes

- 802.3ba-2010, 802.3bg-2011, 802.3bj-2014, 802.3bm-2015

100Gb Ethernet

- 4 lanes of 25GB

Terabit Ethernet?

- Maybe someday?
- Still under discussion

Energy Efficient Ethernet

- Energy Efficient Ethernet (EEE) IEEE 802.3az (2010)
- Turn off when connection not up
- Use less power on shorter cables
- Low-power mode when idle

Autonegotiation

- How figure out line speed and duplex
- Series of pulses sent along periodically
- If not received for 150ms, link is detected as down
- Encoded in the 16-bit series of pulses is the speed / duplex available

Auto-sensing vs Auto-negotiation

- When you connect, each side sends list of what it supports
- Problem, if they have the same list will they pick the same to use? Not always
- If somehow doesn't send list, has to guess, passively
- This is difficult, why fast ethernet uses auto-negotiating rather than auto-sensing
- need to detect speed, and duplex
- Speed – look at 1010101010 at start of frame

- Duplex – try to cause a collision

Auto-negotiation

- Similar to link integrity test pulses (LIT) unipolar positive-only pulses, 100ns
- That's how sense link
- For negotiation, fast-link pulse, 16-bit pattern describes link
- duplex, if it can't see a negotiation defaults to half-duplex as it's safe?

What does your machine have

- skylake machine:

```
[   18.240021] e1000e: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: Rx/Tx
```

- Raspberry Pi:

```
[   77.110505] smsc95xx 1-1.1:1.0 eth0: link up, 100Mbps, full-duplex, lpa 0xC5E1
```

- Haswell machine:

```
[    3.907651] tg3 0000:03:00.0 eth0: Tigon3 [partno(BCM95761) rev 5761100]
 (PCI Express) MAC address f0:92:1c:f5:e8:f3
[    3.919115] tg3 0000:03:00.0 eth0: attached PHY is 5761
 (10/100/1000Base-T Ethernet) (WireSpeed[1], EEE[0])
```

```
[    3.929838] tg3 0000:03:00.0 eth0: RXcsums[1] LinkChgREG[0] MIirq[0] ASF[1] TSOcap[1]
[    3.938174] tg3 0000:03:00.0 eth0: dma_rwctrl[76180000] dma_mask[64-bit]
[   13.758613] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[   15.404905] tg3 0000:03:00.0 eth0: Link is up at 100 Mbps, full duplex
[   15.411479] tg3 0000:03:00.0 eth0: Flow control is on for TX and on for RX
```

Linux OS Support

- When frame comes in, interrupt comes in
- Allocates `sk_buff` copies in
- Old: `net_if_rx()` interrupt, `net_rx_action()` interrupt/polling
- `net_if_receive_skb()`
- passes it to proper net level (`ip_recv()`, `ip_ipv6_recv()`, `arp_recv()`)

- for send
- `net_tx_action()`
  `dev_queue_xmit()` and then deallocate `sk_buff`
- `qdisc_run()` selects next frame to transmit and calls `dequeue_skb()`

Other Wired Connections

- Note: all high-speed copper interconnects have similar issues to ethernet
- Low speeds (i2c, SPI, etc) just plain NRZ
- Faster like USB are?
  Step up for each generation
- PCIe?