Network #1: Ethernet, DHCP, ARP, and WiFi

Meme of the Day

Meme of the Day (True: It's called "Machine Learning")

Outline

- Today's Focus, the low level LAN: Physical and Link Layer
- Ethernet
  - And then Wireless Ethernet
- Broadcast networks and packet injection
- Wireless security and (in)security
- The Key Broadcast Protocols:
  - DHCP:
    - How do I know what I should be
  - ARP:
    - How do I find out who to talk to?
- Fixing Broadcast: Smart Switches

So What Happens When You Search Google on Wifi...

- Step 1, join the Wireless Network:
- Your computer shouts out:
  - "Hey, does Wireless Network X exist?"
- Wireless points continually shout out:
  - "Hey, I'm Wireless Network Y, Join Me"
- If either match up...
  - Your computer then joins the network
  - ***Optionally*** performs a cryptographic negotiation

So What Happens When You Search Google on Wifi...

- Step 2, Configure Your Connection:
- Your computer shouts out on the ***local*** network:
  - "Hey, anybody, what basic configuration do I need to use?"
    - Internet address (IP address)
    - Gateway (where should I send packets destined to the Internet)
    - DNS server (the system which maps "www.google.com" to an IP address (eg, 102.14.183.12 for IPv4 (32b value, presented as 4 integers from 0-255), cafe:f00d:f00d:000f:02:21:1a:2 (128b value, presented as 8 hex groups of 16b each) for IPv6
- Some system on the local network says back:
  - Here is your configuration, enjoy

So What Happens When You Search Google on Wifi...

- Step 3, Generate DNS request
  - DNS uses the UDP Internet Protocol: Unreliable datagrams
- Your computer sends a message to the configured DNS server (Recursive Resolver)
  - Hey, what is the IP address for "www.google.com"?
- The DNS server then searches the general Internet
  - In an annoying disturbed process I'll talk about on Thursday
- The DNS server than answers back:
  - "www.google.com" is here....

So What Happens When You Search Google on Wifi...

- Step 4, Establish a TCP connection to the remote host
  - TCP is an in-order, reliable Internet protocol with congestion control
- Your machine sends a TCP "SYN" request to the Google server
  - Google's server responds with a "SYN/ACK"
  - Your machine then replies with an "ACK"
- After this 3-way handshake, your computer then starts to talk to the web server

So What Happens When You Search Google on Wifi...

- Step 5: Negotiate an encrypted TLS session over the TCP connection
- Your computer says:
  - "I want to use an encrypted connection to this host"
- Google replies with:
  - "OK, here's a certificate that proves my public key belongs to me, let's start talking"
- Handshake continues back and forth until the two sides agree on a common cryptographic key

So What Happens When You Search Google on Wifi...

- Step 6: Now its HTTP requests
- Your computer says:
  - I want to fetch the url / for the host www.google.com
- Google replies with:
  - "OK, here you go..."
- Now your browser starts running on the data
  - And this gets into the web security stuff much later in the course...

Layers And The Network

- The network breaks things up into abstraction layers
  - High level layers avoid having to know much about the lower level layers
- Your computer sees just high level operations
  - Open a network connection
  - Open an encrypted network connection
- Layers isolate things
- Major layers:
  - TCP or UDP
  - IP
  - Ethernet

Packets and The Network

- Modern networks break communications up into packets
  - For our purposes, packets contain a variable amount of data up to a maximum specified by the particular network
- The sending computer breaks up the message and the receiving computer puts it back together
  - So the software doesn't actually see the packets per-se
  - Network itself is ***packet switched***: sending each packet on towards its next destination
- Other properties:
  - Packets are received ***correctly*** or not at all in the face of ***random*** errors
    - The network does not enforce correctness in the face of adversarial inputs: They are checksums not cryptographic MACs.
  - Packets may be ***unreliable*** and "dropped"
    - Its up to higher-level protocols to make the connection reliabls

The Basic Ethernet Packet

- An Ethernet Packet contains:
  - A preamble to synchronize data on the wire
    - We normally ignore this when talking about Ethernet
  - 6 bytes of destination MAC address
    - In this case, MAC means media access control address, not message authentication code!
  - 6 bytes of source MAC address
  - Optional 4-byte VLAN tag
  - 2 bytes length/type field
  - 46-1500B of payload

DST MAC	SRC MAC	VLAN	Type	PAYLOAD

The MAC Address

- The MAC acts as a device identifier
  - The upper 3 bytes are assigned to a manufacturer
    - Can usually identify product with just the MAC address
  - The lower 3 bytes are assigned to a specific device
    - Making the MAC a de-facto serial #
- Usually written as 6 bytes in hex:
  - e.g. `13:37:ca:fe:f0:0d`
- A device ***should ignore*** all packets that aren't to itself or to the broadcast address (`ff:ff:ff:ff:ff:ff`)
  - But almost all devices can go into ***promiscuous mode***
    - This is also known as "sniffing traffic"
- A device generally should only send with its own address
  - But this is enforced with software and can be trivially bypassed when you need to write "raw packets"

The Hub...

- In the old days, Ethernet was simply a shared broadcast medium
  - Every system on the network could hear every sent packet
- Implemented by either a long shared wire or a "hub" which repeated every message to all other systems on the network
  - Thus the only thing preventing every other computer from listening in is simply the network card's default to ignore anything not directed at it
- The hub or wire is incapable of enforcing senders either
  - Any sender could simply lie about it's MAC address when constructing a packet

The Hub Yet Lives!

- WiFi is effectively “Ethernet over Wireless”
  - With ***optional*** encryption which we will cover later
- Open wireless networks are just like the old Ethernet hub:
  - Any recipient can hear all the other sender’s traffic
  - Any sender can use any MAC address it desires
- With the added bonus of easy to hijack connections
  - By default, your computer sends out “hey, is anyone here” looking for networks it knows
  - For open networks, anybody can say “Oh, yeah, here I am” and your computer connects to them

Rogue Access Points...

- Since unsecured wireless has no authentication...
  - And since devices by default shout out "hey, is anyone here network X"
- You can create an AP that simply responds with "of course I am"
  - The mana toolkit: https://github.com/sensepost/mana
- Now simply relay the victim's traffic onward
  - And do whatever you want to any unencrypted requests that either happen automatically or when the user actually does something
- I ***suspect*** I've seen this happening around Berkeley
  - Seen an occasional unencrypted version of a password protected network I'd normally use
- Recommendations:
  - Do ***not*** remember unsecured networks
  - Do ***not*** have your computer auto-join open networks

`tcpdump`

- The **`tcpdump`** program allows you to see packets on the network
  - It puts your computer's card into promiscuous mode so it ignores MAC addresses
- You can add additional filters to isolate things
  - EG, only to and from your own IP
  - **`sudo tcpdump -i en0 host {myip}`**
- Note: this is ***wiretapping***
  - DO NOT RUN on a random open wireless network without a filter to limit the traffic you see
  - Only run without filters when connected to your own network
    - But do run it when you get home!

Broadcast is Dangerous: Packet Injection

- If your attacker can see your packets…
  - It isn't just an information leakage
- Instead, an attacker can also ***inject*** their own packets
  - The low level network does not enforce any ***integrity or authenticity***
- So unless the high level protocol uses cryptographic checks…
- The target simply accepts the ***first*** packet it receives as valid!
  - This is a "race condition attack", whichever packet arrives first is accepted

Packet Injection in Action: Airpwn

But Airpwn ain't a joke…

- It is trivial to replace "look for .jpg request and reply with redirect to goatse" with "look for .js request and reply with redirect to exploitive javascript"
  - This JavaScript would start running in the target's web browser, profile the browser, and then use whatever exploits exist
- The requirements for such an attack:
  - The target's traffic must not be encrypted
  - The ability to see the target's traffic
  - The ability to determine that the target's traffic belongs to the target
  - The ability to inject a malicious reply

So Where Does This Occur?

- Open wireless networks
  - E.g. Starbucks, and ***any wireless network without a password***
  - Only safe solution for open wireless is ***only*** use encrypted connections
    - HTTPS/TLS, `ssh`, or a Virtual Private Network to a better network
- On backbones controlled by nation-state adversaries!
  - The NSA's super-duper-top-secret attack tool, QUANTUM is ***literally*** airpwn without the goatse!
    - Not an exaggeration: Airpwn only looks at single packets, so does QUANTUM!

It's also *too* easy

- Which is why it isn't an assignment!
- Building it in scapy, a packet library in python:
  - Open a sniffer interface in one thread
    - Pass all packets to a separate work thread so the sniffer doesn't block
  - For the first TCP data packet on any flow destined on port 80
    - Examine the payload with a simple regular expression to see if its a fetch for an image (ends in .jpg or .gif) and not for our own server
      - Afterwards whitelist that flow so you ignore it
  - If so, construct a 302 reply
    - Sending the browser to the target image
  - And create a fake TCP packet in reply
    - Switch the SYN and ACK, ports, and addresses
    - Set the ACK to additionally have the length of the request
    - Inject the reply

Detecting Injected Packets: Race Conditions

- Clients ***can*** detect an injected packet
  - Since they still see the original reply
- Packets can be duplicated, but they should be consistent
  - EG, one version saying “redirect”, the other saying “here is contents” should not occur and represents a ***necessary*** signature of a packet injection attack
- Problem: often detectable too late
  - Since the computer may have acted on the injected packet in a dangerous way before the real reply arrives
- Problem: nobody does this in practice
  - So you don't actually see the detectors work
- Problem: “Paxson’s Law of Internet Measurement”
  - “The Internet is weirder than you think, even when you include the effects of Paxson’s Law of Internet Measurement”
  - Detecting bad on the Internet often ends up inadvertently detecting just odd:
    Things are always more broken then you think they are

Wireless Ethernet Security Option: WPA2 Pre Shared Key

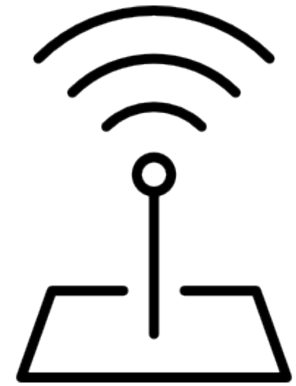
- This is what is used these days when the WiFi is "password protected"
  - The access point and the client have the same pre-shared key (called the PSK key)
  - Goal is to create a shared key called the PTK (Pairwise Transient Key)
- This key is derived from a combination of both the password and the SSID (network name)
  - PSK = PBKDF2(passphrase, ssid, 4096, 256)
- PBKDF2 is effectively a hash function that takes a passphrase, a salt, an iteration count, and an output size
  - The SSID as salt ensures that the same password on different network names is different
  - The iteration count assures that it is ***slow***
    - Any attempt to brute force the passphrase should take a lot of time per guess

The WPA 4-way Handshake

SNonce + MIC

Ack

Computed PTK = F(PSK, ANonce SNonce, AP MAC, Client MAC)



ANonce

GTK + MIC

Computed PTK = F(PSK, ANonce SNonce, AP MAC, Client MAC)

Remarks

- This is ***only*** secure if an eavesdropper doesn't know the pre shared key
  - Otherwise an eavesdropper who sees the handshake can perform the same computations to get the transport key
  - However, by default, network cards don't do this:
    This is a "do not disturb sign" security. It will keep the maid from entering your hotel room but won't stop a burglar
- The MIC is really a MAC, but as MAC also refers to the MAC address, they use MIC in the description
- The GTK is for broadcast
  - So the AP doesn't have to rebroadcast things, but usually does anyway

Actually Making it Secure: WPA Enterprise

- When you set up Airbears 2, it asks you to accept a public key certificate
  - This is the public key of the authentication server
- Now before the 4-way handshake:
  - Your computer first handshakes with the authentication server
    - This is secure using public key cryptography
  - Your computer then authenticates to this server
    - With your username and password
- The server now generates a unique key that it both tells your computer and tells the base station
  - So the 4 way handshake is now secure

But Broadcast Protocols Make It Worse...

- By default, both DHCP and ARP broadcast requests
  - Sent to ***all*** systems on the local area network
- DHCP: Dynamic Host Control Protocol
  - Used to configure all the important network information
    - Including the DNS server:
      If the attacker controls the DNS server they have complete ability to intercept all traffic!
    - Including the Gateway which is where on the LAN a computer sends to:
      If the attacker controls the gateway
- ARP: Address Resolution Protocol
  - "Hey world, what is the Ethernet MAC address of IP X"
  - Used to find both the Gateway's MAC address and other systems on the LAN

So How Do
We Secure the LAN?

- Option 1: We don't
  - Just assume we can keep bad people out
  - This is how most people run their networks:
    "Hard on the outside with a goey chewy caramel center"
- Option 2: ***smart*** switching and active monitoring

The Switch

- Hubs are very inefficient:
  - By broadcasting traffic to all recipients this greatly limits the aggregate network bandwidth
- Instead, most Ethernet uses switches
  - The switch keeps track of which MAC address is seen where
- When a packet comes in:
  - If there is no entry in the MAC cache, broadcast it to all ports
  - If there is an entry, send it just to that port
- Result is vastly improved bandwidth
  - All ports can send or receive at the same time

Smarter Switches: Clean Up the Broadcast Domain

- Modern high-end switches can do even more
  - A large amount of potential packet processing on items of interest
- Basic idea: constrain the broadcast domain
  - Either filter requests so they only go to specific ports
    - Limits other systems from listening
  - Or filter replies
    - Limits other systems from replying
- Locking down the LAN is very important practical security
  - This is ***real*** defense in depth:
    Don't want 'root on random box, pwn whole network'
  - This removes "***pivots***" the attacker can try to extend a small foothold into complete network ownership
- This is why an Enterprise switch may cost $1000s yet provide no more real bandwidth than a $100 Linksys.

Smarter Switches: Virtual Local Area Networks (VLANs)

- Our big expensive switch can connect a lot of things together
  - But really, many are in ***different*** trust domains:
    - Guest wireless
    - Employee wireless
    - Production desktops
    - File Servers
    - etc...
- Want to isolate the different networks from each other
  - Without actually buying separate switches

VLANs

- An ethernet port can exist in one of two modes:
  - Either on a single VLAN
  - On a trunk containing multiple specified VLANs
- All network traffic in a given VLAN stays only within that VLAN
  - The switch makes sure that this occurs
- When moving to/from a trunk the VLAN tag is added or removed
  - But still enforces that a given trunk can only read/write to specific VLANs

Putting It Together:
If I Was In Charge of UC networking...

- I'd isolate networks into 3+ distinct classes
  - The plague pits (AirBears, Dorms, etc)
  - The mildly infected pits (Research)
  - Administration
- Administration would be locked down
  - Separate VLANs
  - Restricted DHCP/system access
  - Isolated from the rest of campus