

# Information security

John Roberts

The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards. And even then, I wouldn't trust my life to it. —  
Gene Spafford

# Information security is...

- Secure communication and storage
  - Confidentiality
  - Integrity
  - Authentication
- Operational security
  - Let the goodies in, keep the baddies out
- Now and in the future!

# How much should you worry

- Don't: ignore it entirely
  - Example; don't broadcast credit card details on an open radio channel
- Don't: Get paranoid
  - Example; don't use incredibly tough encryption for sending an email to your mother.
- You want an educated awareness of risk

# “But who would do such a thing?”

- Hackers/crackers/attackers/etc...
  - Attack systems with technical knowledge
- Social engineering
  - Attacks systems using the weakest link; the users
- War-drivers
  - Opportunistically looking for exposed networks

# “And why?”

- Money!
- More money.
- Political agenda
- Fun
- Out of the goodness of their heart
- Sometimes, money.

# Roots

- Phreakers
  - Exploited the telephone network for fun and profit
  - Cap'n Crunch and the 2600Hz
  - Kevin Poulson
    - Phreaked a radio networks phone system so he was the 102<sup>nd</sup> caller

# What can they get?

- I don't know. What can they get?
- Split into groups, and come up with a short list.

# My list:

- Browser history
- Email addresses
- Usernames
- Card details
- Passwords
- Personally identifying information
- Confidential documents
- A compromised machine
- Access to websites
- Money
- Reputation
- And much more....

# And how do they get it?

- Vulnerabilities
- Malware
- Denial of service attacks

# Vulnerabilities

- Intrusions
  - An attempt to gain unauthorised access to your system
- Malware
  - An automated attack on your system
- Denial of Service attacks
  - Denies authorised users access to the system

# Vulnerabilities

- Mistakes in programs that allow Bad Things to happen.
- Can expose private data, allow unauthorised code to run, or permit access to higher-level system functions.
- Can be simple; see the Moonpig exploit (<http://www.ifc0nfig.com/moonpig-vulnerability/>) or complex.

# Malware

- Automated attacks designed to exploit common vulnerabilities
- Again, can be simple
  - :(){ :|:& };:
- Or incredibly complex
  - Stuxnet
  - Sality

# Denial Of Service

- An attack that does not steal data, but prevents system use
- Examples:
  - locking users out of email
  - Preventing users from accessing websites
  - Not releasing a shared resource or file.
- Can be done by accident
  - See ‘slashdotting’

# Where are your weak points?

- Web browser
- Web servers
- The Network
- The Cloud

# The major browser vulnerability

- The number one, un-patchable, un-workaroundable universal browser vulnerability: USERS.
- Users are:
  - Unskilled
  - Gullible
  - Easily led astray
- Quick activity; work out ways to exploit the number one vulnerability.

# More browser vulnerabilities

- No separation between applications within the browser
- Cross site scripting
- Cross site request forgery
- Cookie injection
- Fake UIs
- Man in the browser attacks

# Server vulnerabilities

- Server = computer with an open door to the Internet.
- Vulnerabilities in
  - Server OS
  - Server software
  - Other servers
  - Misconfiguration

# Network vulnerabilities

- Any shared information has to pass through a network
- This information can be stolen en-route
  - Man in the Middle attacks
  - Router attacks
  - Ethernet traffic sniffing
  - DNS attacks

# “The Cloud”

- The cloud: a collection of servers full of sensitive data...
- ...you connect to over a public network...
- ... usually via a web browser.
- Anyone see any issues with this?

# So what do you do?

- Take steps!
- Cryptography
- Devices
- Protocols
- Security policies

# Cryptography

- The root of all online security
- It...
  - Encrypts messages
  - Authenticates identities
  - Prevents tampering of documents
  - Works as a signature

# Devices

- Dedicated security computers
- Firewalls:
  - Filters traffic in both ways
- Proxy servers
  - Isolates internal and external networks
- Intrusion detection systems
  - Keeps a lookout for attacks

# Protocols

- Agreed methods of secure communications
- Secure Socket Layer (SSL)
  - Protects web traffic
- IPSec
  - Protects network traffic
- Pretty Good Protection (PGP)
  - Protects email

# Security policies

- Your security plan
- If you don't have one, you're part of someone else's intrusion plan.
- Approaches
  - Perimeter security
  - Layered security
  - Some mix of the two

# Perimeter security

- Nothing leaves unless explicitly authorised
  - All information is checked at perimeter
  - External connections are strongly controlled
- Used in contexts where any leaked information is bad
  - Government
  - Military
  - Finance (sometimes)

# Layered security

- Most sensitive information is accessed by the least number of people
  - And vice-versa
- Internal access controls are placed on data
  - Prevents Marketing from accessing payroll data
- Cheaper and simpler than perimeter security
- Widely used in business

# Course structure

- Next week; a reintroduction to the network stack and associated protocols
- Week 3 – 7: Attacking the network
  - Attack methodology and exploits
  - Denial of service attacks
  - Web security
  - Malware

# Course structure (cont.)

- Weeks 8-10: Protecting the network
  - Cryptography
  - Authentication and access control
  - Network protection
  - Data protection
- Week 11: Revision/drop-in/exam panic session.

# Module structure

- 70% exam
  - Four questions, pick three
- 30% coursework
  - Due week 10
  - Details on next slide

# Coursework: phpBB

- Step 1: Install a web server
- Step 2: Install an old version of phpBB (forum software)
- Step 3: Compromise phpBB
- Step 4: Harden phpBB
- Step 5: Write a report on steps 2-4 and hand it into the student information office on Monday 25<sup>th</sup> of March

# Reading materials

- The Web Application Hacker's Handbook
  - Stuttard and Pinto
- Cryptography Engineering: Design Principles and Practical Applications
  - Fergusun, Schneier, Kohno
- Security Engineering (free!)
  - Ross Anderson
  - <http://www.cl.cam.ac.uk/~rja14/book.html>

# Websites

- [www.exploit-db.com](http://www.exploit-db.com)
- [www.owasp.org](http://www.owasp.org)
- [www.symantec.com](http://www.symantec.com)
- [www.kaspersky.co.uk/about/news/virus](http://www.kaspersky.co.uk/about/news/virus)
- [www.googleonlinesecurity.blogspot.co.uk](http://www.googleonlinesecurity.blogspot.co.uk)
- [www.f-secure.com/weblog/](http://www.f-secure.com/weblog/)
- [www.schneier.com](http://www.schneier.com)

# Next week:

Revisiting the network

# Network Architecture

OR: how the Internet works (and doesn't)

# What is the Internet?

- Internet
- 'ɪntənɛt/
- *noun*
- noun: **Internet**
- a global computer network providing a variety of information and communication facilities, consisting of interconnected networks using standardized communication protocols.

# But really, what is the Internet?

- A set of computers and devices all communicating with one another using an agreed set of protocols.
- The Internet Protocol Suite
- Defined by the Internet Engineering Task Force

# A worked example

- You type [www.example.co.uk](http://www.example.co.uk) into your browser.
- Your **application** prepares a HTTP GET request
- The GET request is sent to the **transport** layer, where it is encapsulated into a TCP or UDP packet.

# TCP/UDP

- Used to send data between applications
- Both use logical ports tied to an application/protocol
  - 80: http, 25: SMTP
  - Ports numbered between 1 and 65535; most common <1024
- Transmission Control Protocol:
  - Optimised for reliability
- User Datagram Protocol
  - Optimised for speed

# A worked example

- You type [www.example.co.uk](http://www.example.co.uk) into your browser.
- Your **application** prepares a HTTP GET request
- The GET request is sent to the **transport** layer, where it is encapsulated into a TCP or UDP packet.
- This packet is passed to the **IP layer**, where an IP header is added

# Internet Protocol

- Used to send data between hosts
- IP header contains
  - Source host (How does it know this?)
  - Destination host (How does it know this?)
  - Some other things
- IP body contains everything from earlier
  - TCP/UDP data
  - Application data

# IP addresses

- Provides a unique identifier for a host
  - 32 bits(IPv4) or 128 bits (IPv6)
- IPv4: 255.255.255.255
  - We ran out of these in February 2011
- IPv6: 2001:0:5ef5:79fd:3820:2df3:a88e:621f
  - Much larger address space
- IPv4 and IPv6 do not talk to one another
  - A translator is required

# A worked example

- You type [www.example.co.uk](http://www.example.co.uk) into your browser.
- Your **application** prepares a HTTP GET request
- The GET request is sent to the **transport** layer, where it is encapsulated into a TCP or UDP packet.
- This packet is passed to the **IP layer**, where an IP header is added
- The packet is then sent to the **link layer**, where it is framed and leaves the computer.

# Link layer

- Governs communication between adjacent nodes in a network
- The link layer is told by the IP layer where the frame needs to go (How?)
- Link layer protocols know which wires to light up to pass a message onto a given MAC address
  - MAC address: unique identifier of a networked device
- At this point, it becomes a problem for engineers (circuits, chips and electricity)

# Routing

- All above is fine, if host is plugged straight into [www.example.co.uk](http://www.example.co.uk)
- It probably isn't; therefore, it will have to be routed.
- The IP layer of host knows the first 'hop' on network
  - Sends the packet to this address
  - Packet is then examined by router, passed to the next 'hop'
  - So on, until destination is reached.
- How does your computer find the first 'hop'?

# Supplementary protocols

- Protocols used to manage the internet
- DHCP
- DNS
- ARP
- IMCP

# Dynamic Host Configuration Protocol

- Application layer protocol for managing the network
- You turn your laptop on
- Your laptop broadcasts a discovery packet on 255.255.255.255
- DCHP server sends to your laptop's MAC address:
  - A free IP for it to use
  - The name and IP address of the DNS server
  - The IP address of the 'first hop' router
- Your laptop accepts the IP address and starts using it

# Domain Name System protocol

- Application level protocol that ties domain names and IP addresses together
- You type [www.example.co.uk](http://www.example.co.uk) into your browser
- Your system creates a DNS request, encapsulates it in UDP (port 53) and sends it onto your first hop.
- The request is routed to the DNS server
- DNS server returns the IP address of www.example.co.uk

# Address Resolution Protocol

- Link-layer protocol used to associate IP and MAC addresses
- Your laptop wants to send a packet onto 1.2.3.4
- First, looks inside it's ARP table to see if 1.2.3.4 has an associated MAC address.
- If it doesn't find it, it broadcasts "Who is 1.2.3.4?" to the MAC address FF:FF:FF:FF:FF, that is heard by all computers
- 1.2.3.4 responds with it's MAC address, which your laptop stores.

# Some more on TCP

- TCP is a connection-oriented protocol
  - Aims to establish a two-way channel between hosts
  - Also aims to make that channel as accurate as possible
- Therefore, there is a handshaking protocol;
  - Client sends TCP SYN A packet to server
  - Server responds with SYNACK A+1, B
  - Client sends ACK A+1, B+1
  - A and B are random numbers
- Two way connection has now been established

# Tools

- Internet Control Message Protocol
  - ipconfig/ifconfig
  - ping
  - tracert
- Higher level tools
  - Nmap
  - Wireshark
  - ZAP

# ICMP

- ipconfig:
  - Used to obtain local information about network IP addresses, MAC addresses, gateways, DHCP hosts, ect.
  - Type 'ipconfig' or 'ipconfig /all' into the command line
- Ping
  - Used to test connections between hosts
  - Sends 'are you there?' messages
  - Host replies 'I am here'
  - Type 'ping 1.2.3.4' or 'ping [www.example.co.uk](http://www.example.co.uk)' into the command line

# ICMP cont

- Tracert
  - Mega-ping: return the route taken by the packet to the host
  - Sends an echo packet out with time to live 1, then time to live 2...
  - Use in the same way as ping; ‘tracert [www.reading.ac.uk](http://www.reading.ac.uk)’
- Redirect
  - Used by routers to tell hosts ‘send messages meant for 1.2.3.4 to 5.6.7.8’
  - Used to ensure efficient routing

# NMap

- Port scanning software
- Uses sneaky tricks to establish
  - Ports a target machine has open
  - IP addresses of target machines
  - Operating systems of target machines
  - Device types of target machines
- More on this next week
- [www.nmap.org](http://www.nmap.org)

# Wireshark

- Traffic monitor
- Used to capture messages moving through network card
- Can save and filter capture sessions
- Used for network analysis and troubleshooting
- [www.wireshark.org](http://www.wireshark.org)

# OWASP ZAP

- Zed Attack Proxy
- Used to intercept and edit HTTP requests
- Penetration tester's tool
- Also contains functionality for fuzzing, spidering, messing around with SSL...
- <https://www.owasp.org/index.php/ZAP>

# Next week

Now we know how it works

How is it attacked?

# Attacking the network



- Last lecture: how the network works
- This lecture: How networks are attacked
  - Attacker's methodology
  - Attacker's goals
  - Reconnaissance

# The Methodology

1. Footprint
2. Scan
3. Enumerate
4. Penetrate
5. Attack
6. Cover tracks
7. Install back doors

# Footprinting

- Gathering information about the target: research
- Technical information
  - IP addresses
  - Web presence(websites, email servers, ect)
  - Recent security events
- Or not:
  - Contact details
  - Phone numbers

# Footprinting tools

- Tell you information about the server
- Many web tools
  - who.is
  - w3dt.net
- Also OS based tools
  - nslookup
- Always try multiple tools to get a good read

# Footprinting

- Don't forget about ~~cyberstalking~~ social engineering:
  - Linkedin, Facebook, Researchnet, ect to get information staff
  - Dumpster diving: stealing old info from rubbish bins
- Could get emails and personal information for a spear phishing attack

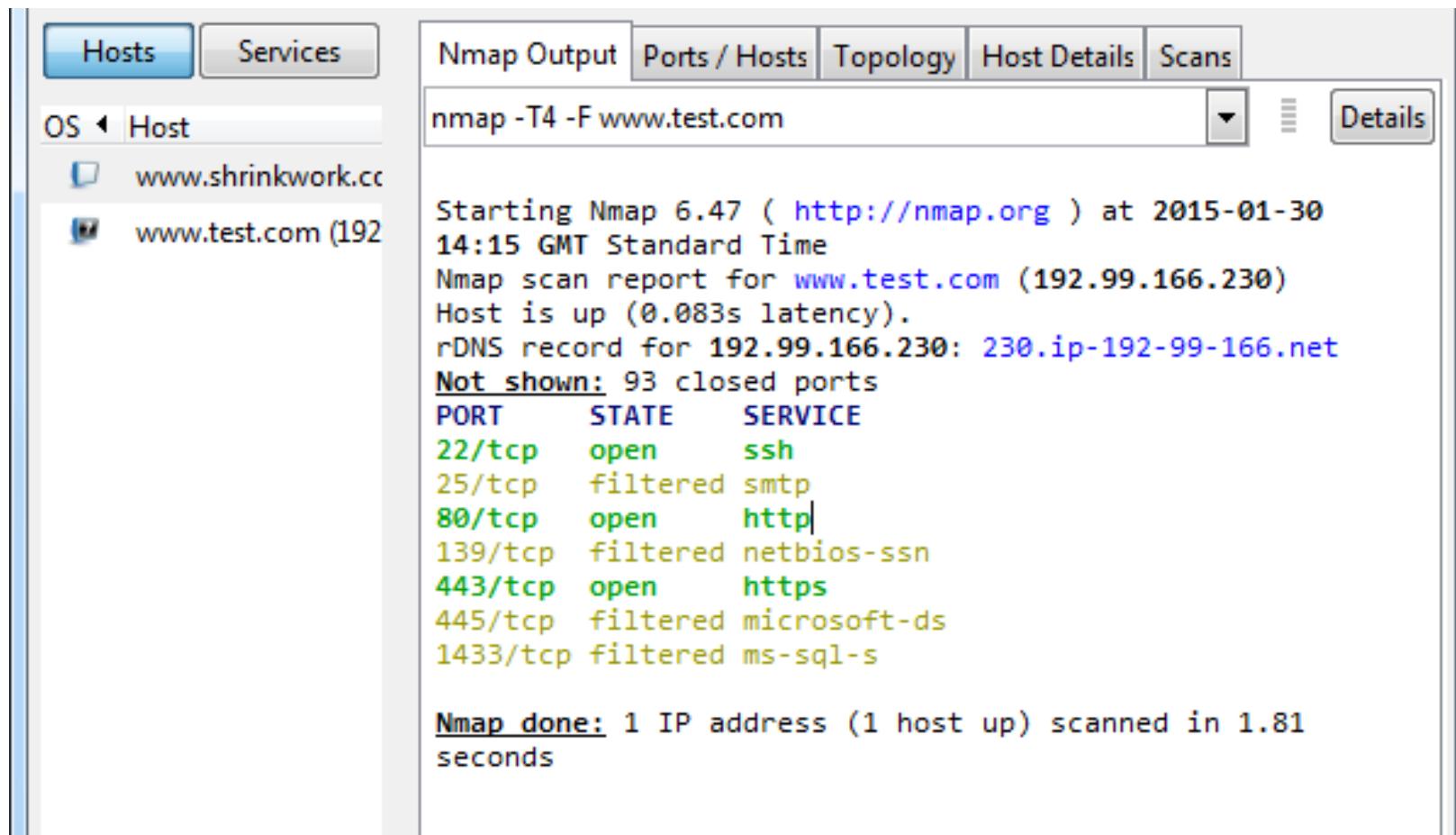
# Scanning

- Determines which of the systems are 'net accessable
  - Which IP addresses are accessable
  - What they are running
  - Any obviously open doors
- At it's most basic:
  - ping the address to see if it's alive
  - Scan the ports to see if any are open

# Scanning

- Identifies open ports
- Google the port numbers: gets applications
- Google the applications: get vulnerabilities
- More on port numbers:
  - 1-1023 are *system*: used by the OS
  - 1024-49151 are *registered* for a specific purpose
  - 49152-65535 are *custom*; used for whatever the user wants.

# Scanning: Nmap output



The screenshot shows a network scanning interface with the following layout and content:

- Left Panel (Hosts):** Shows two hosts: `www.shrinkwork.cc` and `www.test.com (192.99.166.230)`.
- Top Bar:** Includes tabs for **Hosts** (selected), **Services**, **Nmap Output**, **Ports / Hosts**, **Topology**, **Host Details**, and **Scans**.
- Search Bar:** Contains the command `nmap -T4 -F www.test.com`.
- Output Area:** Displays the Nmap scan results for `www.test.com (192.99.166.230)`. The output includes:
  - Scan start information: `Starting Nmap 6.47 ( http://nmap.org ) at 2015-01-30 14:15 GMT Standard Time`
  - Host status: `Host is up (0.083s latency).`
  - DNS information: `rDNS record for 192.99.166.230: 230.ip-192-99-166.net`
  - Ports section:

PORT	STATE	SERVICE
22/tcp	open	ssh
25/tcp	filtered	smtp
80/tcp	open	http
139/tcp	filtered	netbios-ssn
443/tcp	open	https
445/tcp	filtered	microsoft-ds
1433/tcp	filtered	ms-sql-s
  - Summary: `Nmap done: 1 IP address (1 host up) scanned in 1.81 seconds`

# Scanning

- To connect to a port: port must be Open and Listening
- Can just try and establish a TCP connection to the port:
  - Gives the system your IP address, details of the program used to connect, ect.
- Instead, attackers use sneaky manipulation of TCP protocol

# Scanning

- Send a TCP FIN packet
  - Does not make a connection attempt; circumvents firewalls
  - If the port is in LISTEN, no reply
  - If the port is in CLOSED, responds with RESET
- Send a SYN packet
  - If port is open, responds with SYN/ACK
  - You return RESET, no connection

# Scanning: dorking

- Dorking: using advanced Google searches to reveal vulnerable websites
  - Example: `inurl:"installer-log.txt" intext:"DUPLICATOR INSTALL-LOG"`
  - Searches for any Wordpress site moved using the Duplicator plugin
  - Reveals directory information, plugin version, server software, internal structure.
- [www.exploit-db.com/google-dorks/](http://www.exploit-db.com/google-dorks/)

# Enumeration

- The process of identifying 'low hanging fruit' and user accounts
- Scanning shows you the doors, enumeration identifies how to get through them safely
- Long process, usually involving many automated queries of some kind

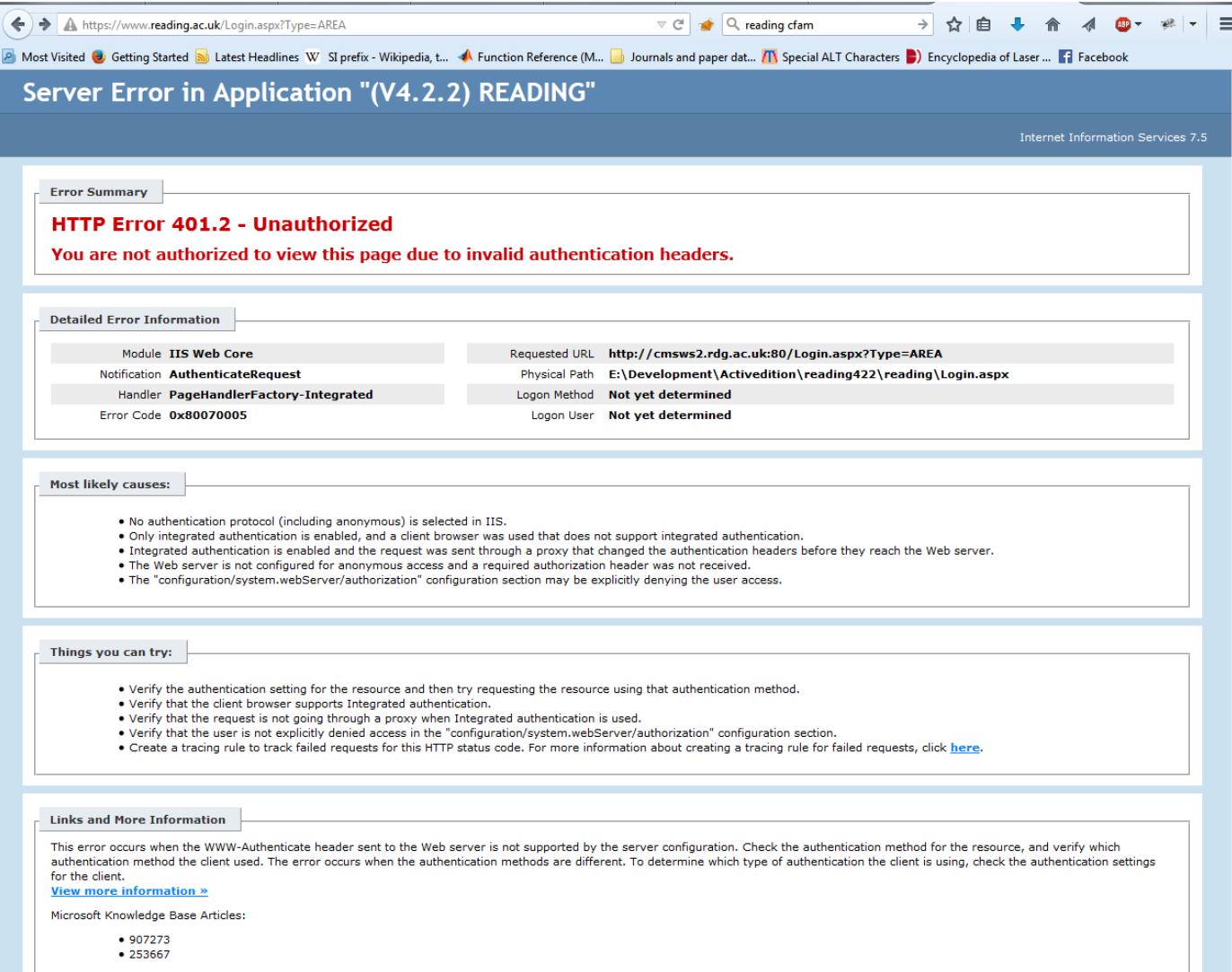
# Enumeration

- An example: Usernames
- Target: some system with password-based authentication (i.e. most of them)
- From your footprinting, you have identified a list of users of your target
- So, start testing possible usernames in the logon
  - Email addresses
  - Known aliases

# Enumeration

- By inspecting the return, you can tell whether a username exists or not
  - Eg. The server returns ‘invalid username’ if the user does not exist, and ‘invalid credentials’ if it does
- Using this, you can build a list of valid usernames

# An example from close to home



The screenshot shows a browser window displaying an IIS 7.5 error page for a 401.2 Unauthorized error. The URL in the address bar is <https://www.reading.ac.uk/Login.aspx?Type=AREA>. The error page title is "Server Error in Application "(V4.2.2) READING".

**Error Summary**

**HTTP Error 401.2 - Unauthorized**  
You are not authorized to view this page due to invalid authentication headers.

**Detailed Error Information**

Module <b>IIS Web Core</b>	Requested URL <b>http://cmsws2.rdg.ac.uk:80/Login.aspx?Type=AREA</b>
Notification <b>AuthenticateRequest</b>	Physical Path <b>E:\Development\Activatedition\reading422\reading&gt;Login.aspx</b>
Handler <b>PageHandlerFactory-Integrated</b>	Logon Method <b>Not yet determined</b>
Error Code <b>0x80070005</b>	Logon User <b>Not yet determined</b>

**Most likely causes:**

- No authentication protocol (including anonymous) is selected in IIS.
- Only integrated authentication is enabled, and a client browser was used that does not support integrated authentication.
- Integrated authentication is enabled and the request was sent through a proxy that changed the authentication headers before they reach the Web server.
- The Web server is not configured for anonymous access and a required authorization header was not received.
- The "configuration/system.webServer/authorization" configuration section may be explicitly denying the user access.

**Things you can try:**

- Verify the authentication setting for the resource and then try requesting the resource using that authentication method.
- Verify that the client browser supports Integrated authentication.
- Verify that the request is not going through a proxy when Integrated authentication is used.
- Verify that the user is not explicitly denied access in the "configuration/system.webServer/authorization" configuration section.
- Create a tracing rule to track failed requests for this HTTP status code. For more information about creating a tracing rule for failed requests, click [here](#).

**Links and More Information**

This error occurs when the WWW-Authenticate header sent to the Web server is not supported by the server configuration. Check the authentication method for the resource, and verify which authentication method the client used. The error occurs when the authentication methods are different. To determine which type of authentication the client is using, check the authentication settings for the client.

[View more information »](#)

Microsoft Knowledge Base Articles:

- 907273
- 253667

# Enumeration: spidering

- Spidering: automated mapping of a website or filesystem
- A program that recursively follows all links in a HTML document
- Can expand to include common directory names, hidden files, ect
- Can reveal old, insecure pages, backups, unreleased content, databases connected to site....

# Penetration

- Entering the system using information discovered up till now
- Once a system is penetrated, the attacker has all the access required for their objective
- More on this later!

# Attacking

- The attacker accomplishes their goal
- Goal could be:
  - Theft of info
  - Denial of service
  - Defacement of web pages
  - Escalation of privilege
    - Using one attack to effect another, larger attack
  - See if they can
- Attacks are only limited by the attackers imagination

# Covering tracks

- Inexperienced attacker leaves evidence
  - Attacker's IP address
  - Attacker's ISP
  - Attacker's organisation
  - Attacker's machine

# Covering tracks

- Professional attackers:
  - Turn off event logging
  - Clearing event logs
    - Better for them to know that someone was there easily than find out who with a little work
  - Hide malicious files that they've left behind (see the next bit)
    - Disguise files as other files
    - Hide evil files inside good files

# Installing backdoors

- Your attacker has done all this hard work: why do it again?
- Installs a back-door to make intrusion easier
- Could be:
  - Rogue user account
  - Scheduled job
  - Hidden program
  - Startup process

# A quick example

- Create two files, ‘good.txt’ and ‘evil.txt’
- In the command line, ‘type evil.txt > good.txt:closet’
  - This pipes the content of evil into a datastream called ‘closet’ in good
- Delete ‘evil.txt’, look in the file
- Then, ‘more < good.txt:closet > evil.txt’
- Ta-da!

# Attacking a web server

1. Identify all entry points
  - Entry point: something you can do to make the server do something
2. Examine the structure of the entry points
  - What form does the request take?
  - What does it look like each part of it does?
  - What happens if you circumvent validation?

# Attacking a web server

3. Are there none-HTTP ways of communicating with the server?
  - Text service? Email commands? API?
  - Does the server use another system that you can manipulate or spoof?
  - What does Nmap say?
4. Identify the server software
5. Enumerate, as detailed earlier
6. Research for known vulnerabilities

# Summary

- Information is all!
  - The more you know about your system, the more you can defend it
  - The less the attacker knows, the harder it is for them to attack
- As admin, you must
  - Know how the attackers work
  - Know their tools
  - Know the methodologies

One more time:  
Use these tools ethically and legally!  
Find gaps, and plug them.

Next week: Denial of Service attacks

# Denial of Service

# Denial of Service attacks

- Computers are not perfect; they only have so many resources
  - Processor cycles
  - Memory
  - Storage space
  - Communication ability
- What if you want to deny those resources for legitimate users?

# DoS targets

- Computational resources
  - Space, processor time, ect.
- Configuration information
  - Routing info
- State information
  - TCP sessions
- Physical network components

# Processor-based DoS

- Remember this? `:(){ :|:&};`
  - A process that spawns copies of itself exponentially
- Takes up system resources until none left for the OS
- No OS = dead machine, needs a restart

# The Billion Laughs

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
  <!ENTITY lol "lol">
  <!ELEMENT lolz (#PCDATA)>
  <!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
  <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
  <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
  <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
  <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
  <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
  <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
  <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
  <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```

# The Billion Laughs

- Another example of an exponential attack
  - XML document has root node “lol9”
  - “lol9” is a macro that substitutes ten copies of “lol8”
  - Each “lol8” is ten copies of “lol7”....
- Results in the XML parser trying to parse ~3GB of entities

# Network DoS attacks

- In a DoS, attacker will spoof IP address
  - Doesn't need any data from server, so n
- Nuke attacks
  - Attacker attempts to crash service through exploit
- Flooding attacks
  - Attacker sends too many packets for targets to handle

# Nuke attacks

- Instead of overwhelming from the outside
- Exploit known bugs in OS
- Examples:
  - Teardrop Attack
  - Land Attack
  - Echo/Chargen Attack
  - Slow Loris Attack

# Nuke: Teardrop Attack

- IP breaks messages up into fragments, reassembles them at destination
- Teardrop: sending packet fragments that can't be properly reconstructed
  - Frames overlap
  - Frames have gaps
- Target cannot rebuild messages, panics, dies.

# Nuke: Land Attack

- Send a packet with the target as both the source and destination
- OS ties itself in a knot trying to connect with itself
- Simple example; send TCP SYN packet
  - Source 1.2.3.4:5050
  - Destination 1.2.3.4:5050

# Nuke: Echo/Chargen Loop

- Character Generator Protocol: Port 19
  - Sends arbitrary characters to the connecting host
- Echo protocol: Port 7
  - Sends back whatever it receives
- What happens if you feed echo into chargen?
- These services are largely disabled now

# Flooding Attacks

- Send target as many packets as possible
  - Need a protocol that firewalls can let through
- Examples
  - UDP Flood
  - TCP SYN Flood
  - ICMP Flood
  - Smurf IP attack
  - POST DoS

# UDP Flood

- Attacker sends UDP packet to some port
- No service listening on port
  - Target sends ‘destination unreachable’ packet
- Target exhausts resources replying to packets
- Attacker doesn’t care about response
  - Can spoof IP address

# TCP SYN Flood

- Attacker sends SYN packet with spoofed IP
- Target replies with SYNACK packet
- Attacker keeps sending SYN packets
- Target keeps sending SYNACK packets
- No resources left to serve legitimate users

# ICMP Flood

- ICMP used by
  - Network protocols to communicate
  - Admins to set up and
  - Generally allowed by firewalls
- ICMP Flood Attacks
  - Eg Ping flood: send many ping packets
  - Ping is let through firewalls by default
- Simple protection: disallow ICMP packets from outside the network

# Smurf attack

- Variant of a ping flood attack
  - Ping sends echo packets to destination
  - Destination replies with reply packets to source
- Instead of sending many echoes to one target with false sources...
- Send many echoes to many ‘smurfs’ with the source IP as the target
- All the ‘smurfs’ send reply packets to target

# Problems With Spoofing

- Spoofed attacks can still be traced
  - ISP can backtrack packets to original source
- Not all that good
  - Systems today are much better than they used to be
  - Application attacks much better (see later)
- What is an attacker to do?

# Distributed Denial of Service

- Attacking machine completes three-way handshake, starts conversation
  - No IP spoofing
  - Attacking machine's ID revealed
- Attacker uses 'zombies'
  - Compromised machines controlled by attacker
  - Zombies are cutout between target and attacker
  - Zombies make many 'legitimate' requests

# DDoS example: TNF2k

- TNF2K: Botnet
  - Network of compromised machines
- Master
  - Attackers machine
- Handlers
  - Zombie machines
  - Each handles up to 1000 agents
- Agents
  - Zombie machine controlled by Handlers
  - Perform the DDoS itself

# Botnet for rent

**ATTACK METHODS**

- ✓ SUDP
- ✓ UDP-LAG (Coming Soon)
- ✓ SSYN
- ✓ CSSYN (Coming Soon)
- ✓ ARME (Coming Soon)
- ✓ RUDY (Coming Soon)
- ✓ AND MORE!

**MORE FEATURES**

- ✓ Cloudflare Resolver
- ✓ Skype Resolver
- ✓ Host to IP
- ✓ Referral System
- ✓ Friends & Enemies
- ✓ IP Logger
- ✓ Geo Location
- ✓ Auto Setup

**1 DAY TRIAL**

MAX BOOT TIME  
60 SECONDS

\$1.50

**BRONZE MONTHLY**

MAX BOOT TIME  
200 SECONDS

\$5.00

**SILVER MONTHLY**

MAX BOOT TIME  
350 SECONDS

\$10.00

# Application Attacks

- Instead of attacking the OS, attack an application
  - Send legitimate traffic maliciously
  - Guaranteed to get through firewall
  - Let application tie up machine
- Examples:
  - HTTP Flood
  - POST DoS

# HTTP Flood

- Have each zombie spider target website
  - Follow every link on each site recursively
- Target consumes resources responding to zombies
- Generates too many HTTP requests for server to handle

# POST DoS

- Attacker sends many POST requests to server with large content-length headers
- Then sends the body of the POST 1 character every 10-100 seconds
  - Quickly enough not to time out, but only just
- Server eventually hits the active connection limit

# Improving DDoS

- Reflection attacks
  - Send packets with target's address as source to known service
  - Attack is 'reflected' off known service
- Amplification attacks
  - Use intermediary that generates large/multiple packets
  - Each zombie action is multiplied by the intermediary

# Weaknesses of DoS/DDoS

- If the flood stops, the attack stops
  - Fix the zombies, stop the attack
- ISPs can trace DoS and possibly DDoS
  - Attacker's machine is exposed if not careful
  - Sophisticated botnets use stenography and encryption to hide commands

# Defending against DoS

- Keep AV updated
- Keep OS updated
- Keep software updated
- Keep everything updated!
- Proper firewall setup
  - Disallow ICMP packets from outside the network
- Protocol modification
  - SYN cookies
  - Random drops

# TCP Intercepting Firewall

- Firewall sits between server and Internet
  - Talks to incoming connections
  - Validates connections
  - Connects to server
- Contains hardened TCP stack
  - Aggressively fast timeouts
  - Configurable thresholds
- As firewall is just setting up sockets, can handle much more than server

# SYN Cookies

- Prevents reservation of resources on initial request
- Encrypt connection info in the sequence number  $n$  in SYNACK
  - Stores connection information ‘in the client’
  - Decrypt information when client sends ACK  $n+1$
- Alternative: RST cookies
  - Reply with SYNACK on first connection
  - If client responds with RST, then legitimate

# Random Drops

- Means of dealing with non-spoofed DDoS attacks
  - Assumes that real user will reconnect, but zombie won't
- Drop connection requests at random
  - A legitimate user will reconnect
  - In the event of an attack, more attackers will be dropped than legitimate users

# Further DDoS defence

- Provide excess bandwidth (expensive)
- Have replicated servers (expensive)
  - When one service falls, switch legitimate users to another
- Limit the rate of certain traffic
- TCP intercepting firewalls (again)
- Use a DDoS scrubbing service
  - All traffic is routed to dedicated cleaning server
  - Legitimate traffic is passed on, malicious is not

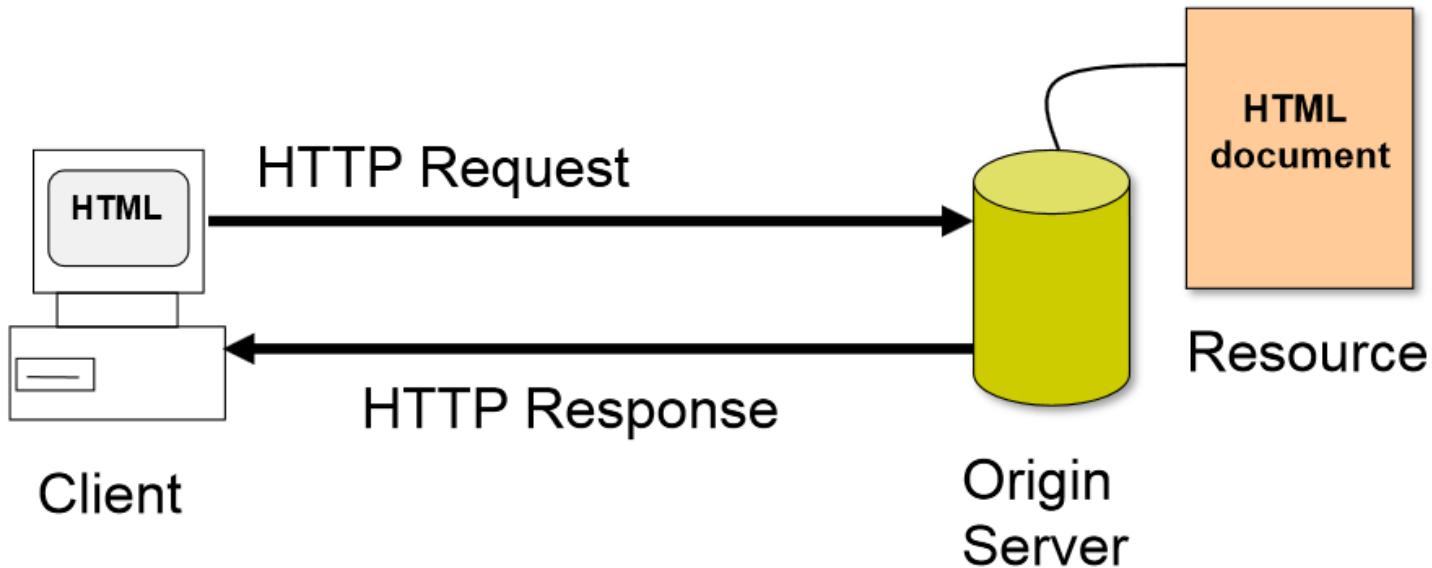
# Summary

- Denial of Service Attacks are
  - Simple
  - Cheap
  - Painful
- Denial of Service Defense is
  - Complex
  - Expensive
- This is an ongoing research area!

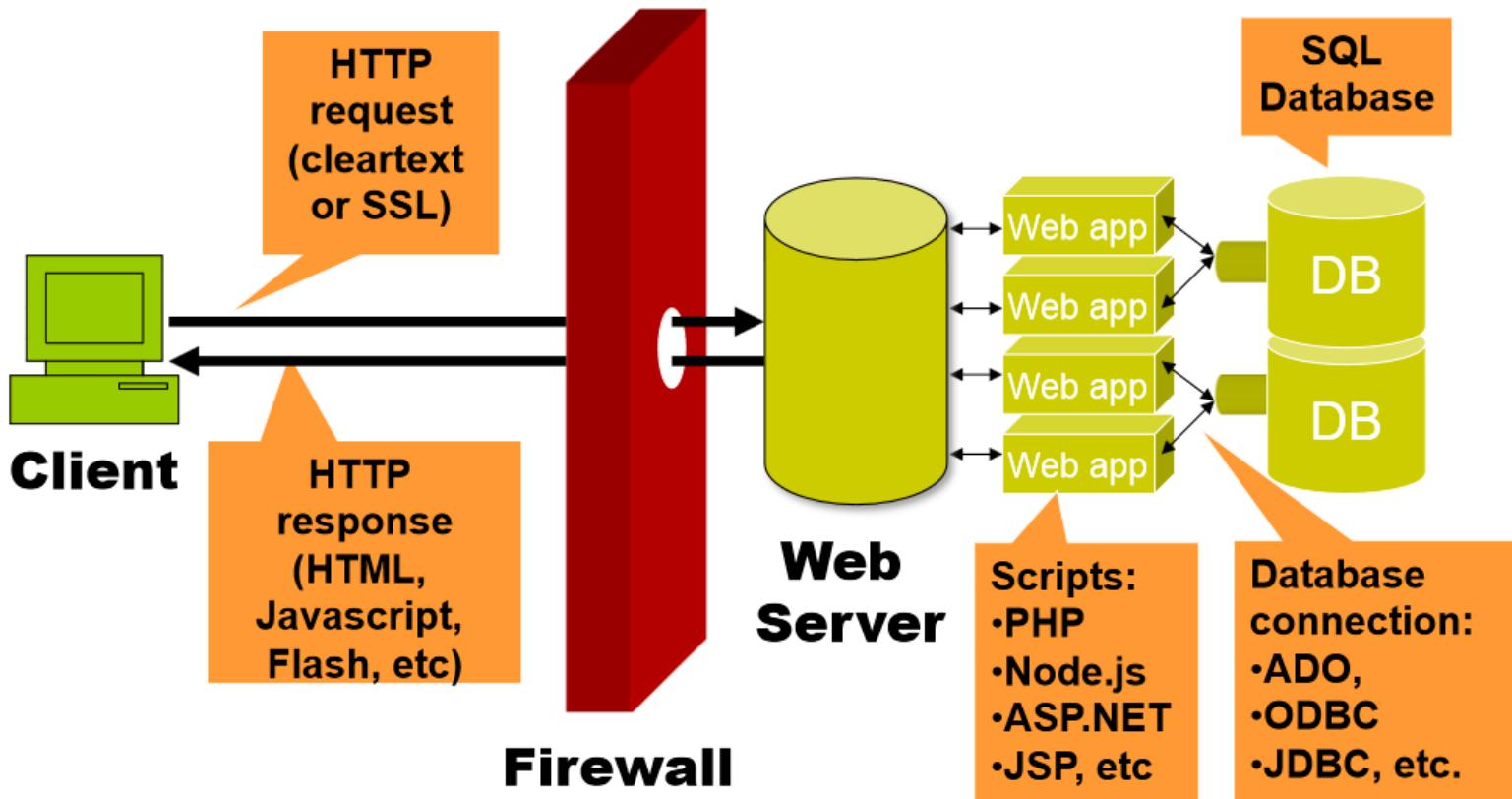
Next week: Websites

# Web Security

# Original architecture

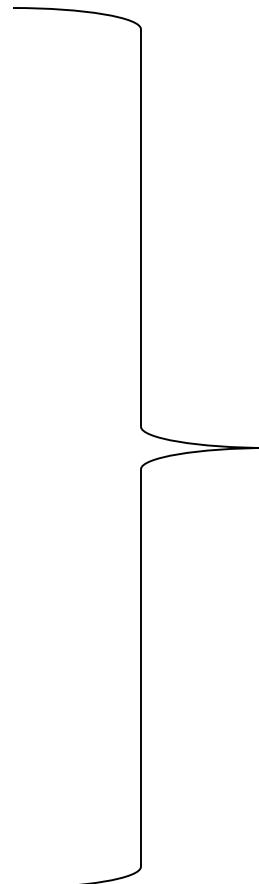


# Present day



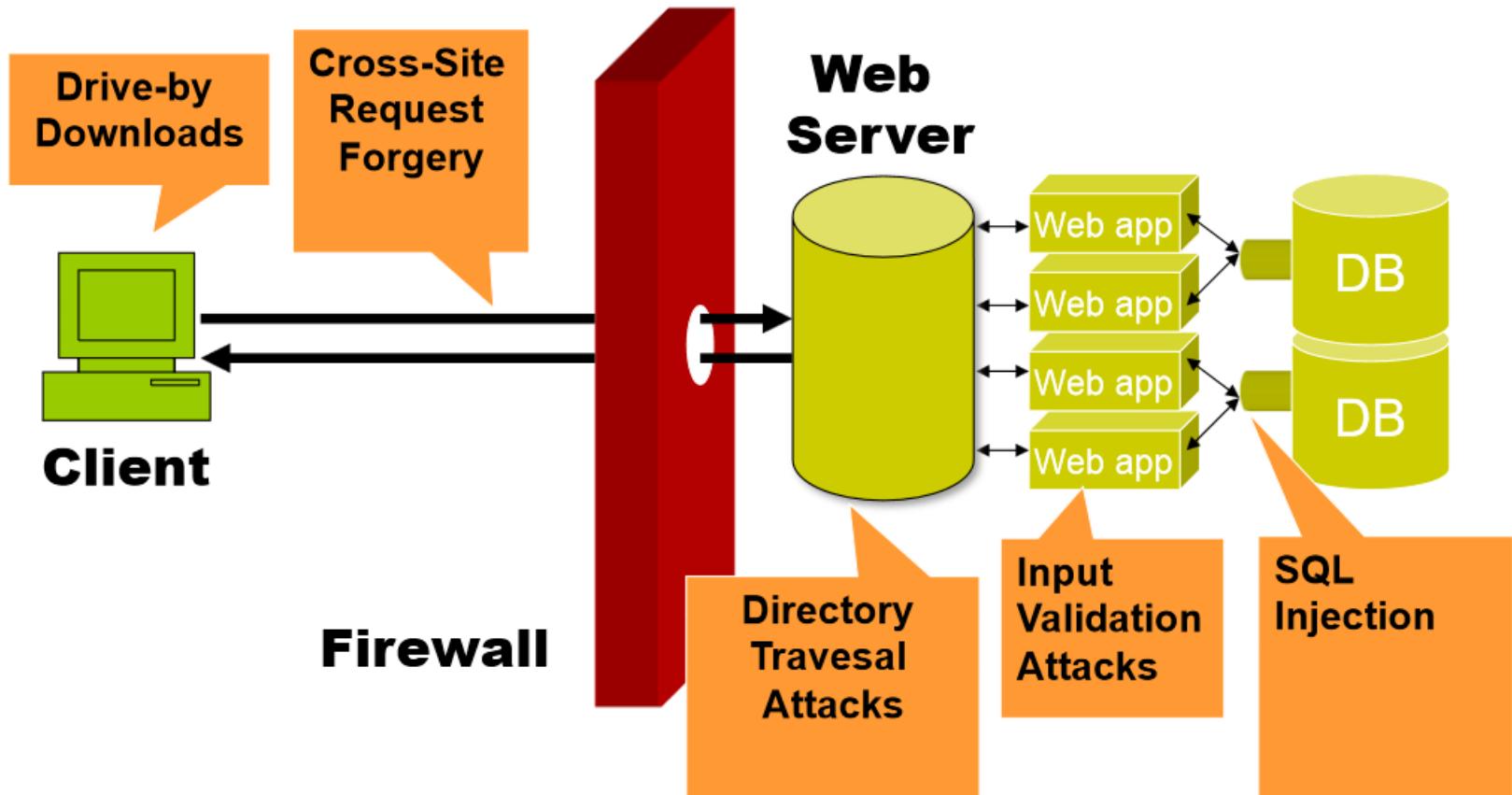
# Components of a web server

- Network
- OS
- Firewall
- Server program
- Database
- FTP server
- Mail server
- Web apps



All have  
vulnerabilities

# Vulnerabilities



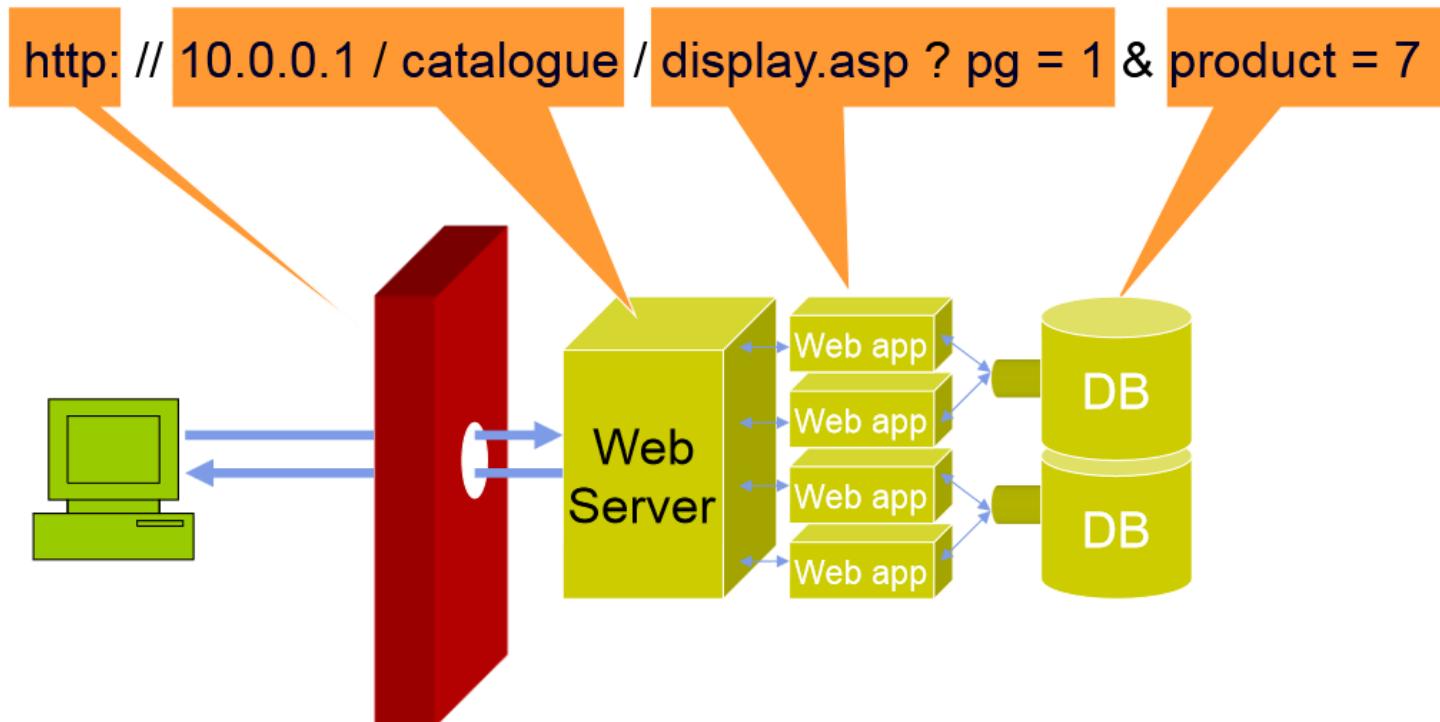
# Firewalls

- A barrier between trusted and untrusted networks
- Prevents unauthorised traffic from reaching the network
- Will stop most network traffic
  - i.e. will prevent, say, FTP requests from outside the trusted network if not required
- They won't stop HTTP!

# URLs

- Uniform Resource Locator
  - Refers to a resource
  - Can also pass data to a server
- Consists of
  - Protocol (http, ftp, ect)
  - Domain ([www.bbc.com](http://www.bbc.com), 1.2.3.4)
  - Path (/news, /login.php)
  - Query string (?username=Stubbs)

# URLs



# Attacking this

- Buffer overflows
  - Directory traversal
  - Double encoding
  - SQL injection
  - Cross site scripting
  - Cross site request forgeries
  - Bad setup!
- ....and plenty more

[http://owasp.com/index.php/Category:OWASP\\_Top\\_Ten\\_Project](http://owasp.com/index.php/Category:OWASP_Top_Ten_Project)

# Buffer overflows

- Cause
  - Poorly written programmes
- Effect
  - Kills servers
  - Can return error messages to attacker
  - Can allow attacker to run own code on machine!

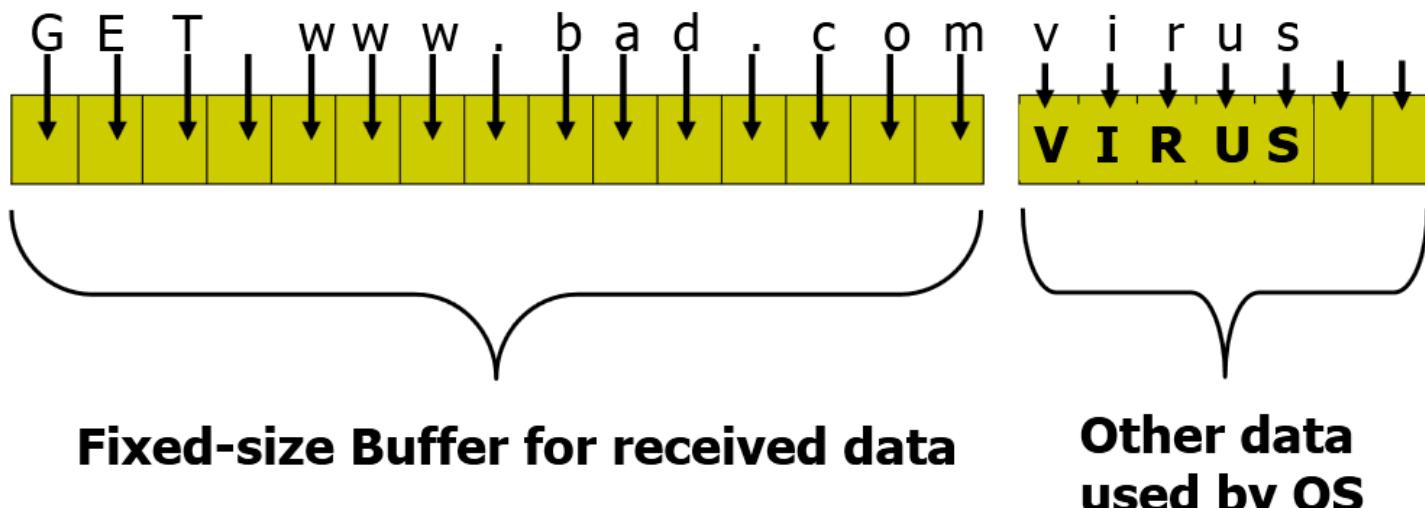
# Buffer overflows

- Remember doing C/C++, all those years ago?

```
buffer char[30];
```

- What happens if you force > 30 characters into buffer?
- The buffer runs over into other memory
  - If you're lucky, you get a general protection fault
  - If you're not, it interferes with other progs/OS

# Deliberate overflowing of buffer



**Poorly-written programs offer no protection**

# Countermeasures

- If you are writing code
  - Always assume someone will try to break it
  - Always validate your input
  - Higher level languages provide protection
    - BUT: You're relying on the language maintainer for that
  - If using a low-level language, use safe types
    - fprintf, not printf
- If you are using code
  - Patch!

# Directory traversal

- Attacker uses tricks to get to places they're not supposed to be
  - `www.chump.com/home/../../winnt/system23/cmd.exe?/c +dir+c:\`
  - Will provide command access for server
- Easily prevented:
  - Filter '../' and similar strings

# Double encoding

- Sneaky trick: Server backend processing is insecure
  - `../` = `%2E%2E%2F` in hex
  - `'%2E'` = `'.'`
  - `'%2F'` = `'/'`
- What if filter catches this?
  - `%2E%2E%2F` = `%252E%252E%252F` in hex
  - ...and so on until filter is bypassed

# Double encoding

- As the backend ‘understands’ hex, it reconstitutes the malicious URL
  - So

`http://victim/cgi/%252E%252E%252F%252E%252E%252Fwinnt/system32/cm  
d.exe?/c+dir+c:\`

- Becomes
- `http://victim/cgi/../../winnt/system32/cmd.exe?/c+dir+c:\`
- But behind the filter!

# Double encoding

- Versatile tool
  - Can be used to bypass many text-based security filters
    - e.g, XSS and SQL injection filters (more later)
- Defence:
  - Filter dangerous characters
    - Have a whitelist of allowed characters
  - Reencode at every ‘boundary’

# SQL injection

- Server has some relational database backend
- Site has confidential ownership information stored in backend database
- Stores item-username pair
  - Example; user Mario wants to log in, check what he has in his storage
  - Puts string for username into web form

# SQL injection

- Server has code

```
$username= $_GET["username"]  
$data = mysql_query("SELECT * FROM items WHERE owner  
= '".$username."' ;")
```

- Final query string :

```
SELECT * FROM items WHERE owner='mario' ;
```

- What if user puts in (quotes included)

' or 1=1 ''

# SQL injection

- SQL query becomes

```
SELECT * FROM items WHERE username= '' OR 1=1;
```

- WHERE clause always evaluates to true

- Spits out entire table of items

- Attacker can see who owns what

- Could be embarrassing when applied to medical records, order history, ect

- But there's more

# SQL injection

- Very versatile attack
- Can inject
  - INSERT statements
  - UPDATE statements
  - DELETE statements
  - DROP DATABASE
    - ‘Little Bobby Tables’

# SQL injection

- Can fingerprint database technology
  - Oracle: 'serv' || 'ices'
  - MS-SQL: 'serv'+ 'ices'
  - MySQL: 'serv' 'ices'
- Can also derive number of columns in table
  - GROUP BY 1,2,3...
  - Keep going until you get an error
- So useful as footprinting tool
- [https://websec.ca/kb/sql\\_injection](https://websec.ca/kb/sql_injection)

# Countermeasures

- Do not trust the user!
  - Do not concatenate strings into SQL
  - Use library-specific parameter bindings
    - `bindParam('owner', $owner)`
- Filter code
  - for evil SQL characters
  - for own table/column names
- Check query output as well as input
- Log all queries
- Make sure every SQL account has only the permissions they need
- Do not sanitise code clientside
  - Client can get around that easily

# Cross-site scripting (XSS)

- Some web service stores and retrieves user text
  - Guestbook
  - Comment system
  - Messageboard
- Text is stored, integrated into returned HTML by server software
- What if that text is not just text?

# XSS

- Say attacker writes a comment

```
<script>alert ('gotcha')</script>
```

- Page HTML now looks like:

```
<comments>
  <comment>
    <script>alert ('gotcha')</script>
  </comment>
</comments>
```

- Anyone visiting the page has the code run
- Code could
  - capture authentication cookie
  - cause user to be redirected to site
  - Anything a website can do, a XSS can do

# XSS

- Two main varieties
  - Stored
    - Malicious script is stored in the server, as in previous example
    - XSS persists as long as admin doesn't clean up server
  - Reflected
    - Malicious script is embedded in a URL  
`www.chump.com/something.php?comment=<script>alert('hahaha')</script>`
    - Script does not persist, leaves less trace on server

# XSS countermeasures

- Validate input!
  - Have a limit on input length
  - Don't allow <script> tags, ect.
  - Check it with a regex, if required
- Also validate output!
  - HTML encode all data to be sent to client
  - Makes sure that browser displays data and does not run it

# Cross-Site Request Forgery

- Some app relies on cookies for authentication
- Cookies are available to every website the user visits
- Attacker can trick victim into requesting something bad for them

# CSRF example: bank transfer

- Say Andy is logged into bank.com
- bank.com uses GET request to perform money transfers

GET http://bank.com/transfer.do?acct=BOB&amount=100 HTTP/1.1

- Sends £100 to user BOB
- Maria sends Andy an email, knowing he's logged in, with code:

`<a href="http://bank.com/transfer.do?acct=MARIA&amount=100000">View my Pictures!</a>`

- Andy's browser accesses the URL and thus tells bank.com to send Maria £100000

# CSRF countermeasures

- Synchronised Token Pattern
  - When server generates web page, include random hidden token
  - This token persists for user's session
  - Browser passes this token back to server with every request.
  - Server checks token, validates request.

# And many more

- <http://owasp.com/index.php/Category:Attack>
  - Big list of attacks and countermeasures
  - STRONGLY RECOMMEND you read the top 10 at least
- [ha.ckers.org](http://ha.ckers.org)
  - Case studies
  - Examples

# Protecting yourself

- Server-side scripters:
  - All input is EVIL: do not trust it
  - Check data sizes
  - Check ranges
  - Check and remove evil characters
  - Sanitize your input
  - Do not rely on authentication cookies

# Protecting yourself

- Webmasters
  - ALWAYS change default passwords
  - ALWAYS remove sample applications
  - ALWAYS PATCH
  - Test your own server
    - Think like attacker
  - Do not keep confidential docs on server machines if you can avoid it
  - Switch off error reporting

The end

# Malware



# What is malware?

- Malware: code written to cause unanticipated or undesired effects in programs or program parts, caused by an agent intent on damage
  - Agent: Writer or person who causes distribution
- Key: “Written to cause”
  - Malware is written with the intent to cause harm

# The menagerie

- Virus
  - Self-replicating program that installs itself without users consent
  - Modifies other files and programs
  - Spreads through any medium (network, storage, ect)
- Worm
  - A program that copies itself through a network
  - Is a standalone program; does not attach to other files
  - Operates solely though networks

# The menagerie

- Trojan horse
  - Innocent code with malicious code hidden inside
  - Eg custom phone keyboard that captures keystrokes
- Spam injection
  - Inserts spam links into web server
- Logic/time bomb
  - Code that executes when certain condition is met

# The menagerie

- Backdoor/trapdoor
  - Code designed to make a system simpler to penetrate for a future attack
    - Can also appear in cryptographic algorithms
- Wabbit
  - Continually replicating program that aims to exhaust resources

# Exploit

- A weakness or mistake in a program that malware can use
- An exploit that has not been noticed by the devs is a **zero day exploit**.
  - Devs have had zero days to fix the problem
- Once noticed, devs will have a patch to fix the exploit as soon as possible
  - Sometimes

# Viruses

- Parasitic software **fragment** that attaches itself to existing executable content
- They have
  - Infection mechanism
    - Means by which it propagates
  - Trigger
    - Event or condition that causes payload to happen
  - Payload
    - Some (probably) bad thing the virus does

# Virus life-cycle

- Dormancy
  - Virus sits in memory/storage/wherever
  - Waits for correct conditions to...
- Propagation
  - Virus uses its replication mechanism to spread
- Triggering
  - Trigger condition is met...
- Execution
  - Payload is dropped, Bad Thing happens

# Symantec model

- Three component model:
  - Wild component: How much the virus already exists
  - Damage component: How much damage the virus would do
  - Distribution component: How quickly the virus spreads
- Each rated high, med or low

# Wild

- Measures the extent to which a virus has already spread
  - Number of independent sites infected
  - Number of computers infected
  - Geographic distribution
  - Ability of current tech to combat threat
  - Complexity
- Ratings:
  - High: 1000 machines/10 sites/5 countries
  - Medium: 50-999 machines/2 sites/2 countries
  - Low: Anything else

# Damage

- Measures the amount of damage the virus could cause
  - Clogged email servers
  - Deleted/modified files
  - Release of confidential info
  - Performance degradation
  - Ease of fixing damage
- Ratings:
  - High: File destruction/modification, destructive triggers, bad stuff
  - Medium: Non-critical setting altered, buggy routines, easily repairable traffic
  - Low: Nothing intentionally bad

# Distribution

- Measures how quickly a program can spread itself
  - Large-scale code attack (worm)
  - Executable code attack (virus)
  - Spreads only through download/copy (trojan)
  - Network awareness
  - Difficulty of movement/repair
- Ratings:
  - High: Worms, network-aware malware, uncontrollable threats
  - Medium: Most viruses
  - Low: Most Trojans

# Basic types of virus

- Boot-sector
  - Original virus
  - Sits on boot sector of HD/floppy, activated on boot
- File
  - Infects some executable (.exe, .bat, ect), activated on run
- Macro
  - Sits in an application with scripting capabilities, opens on run

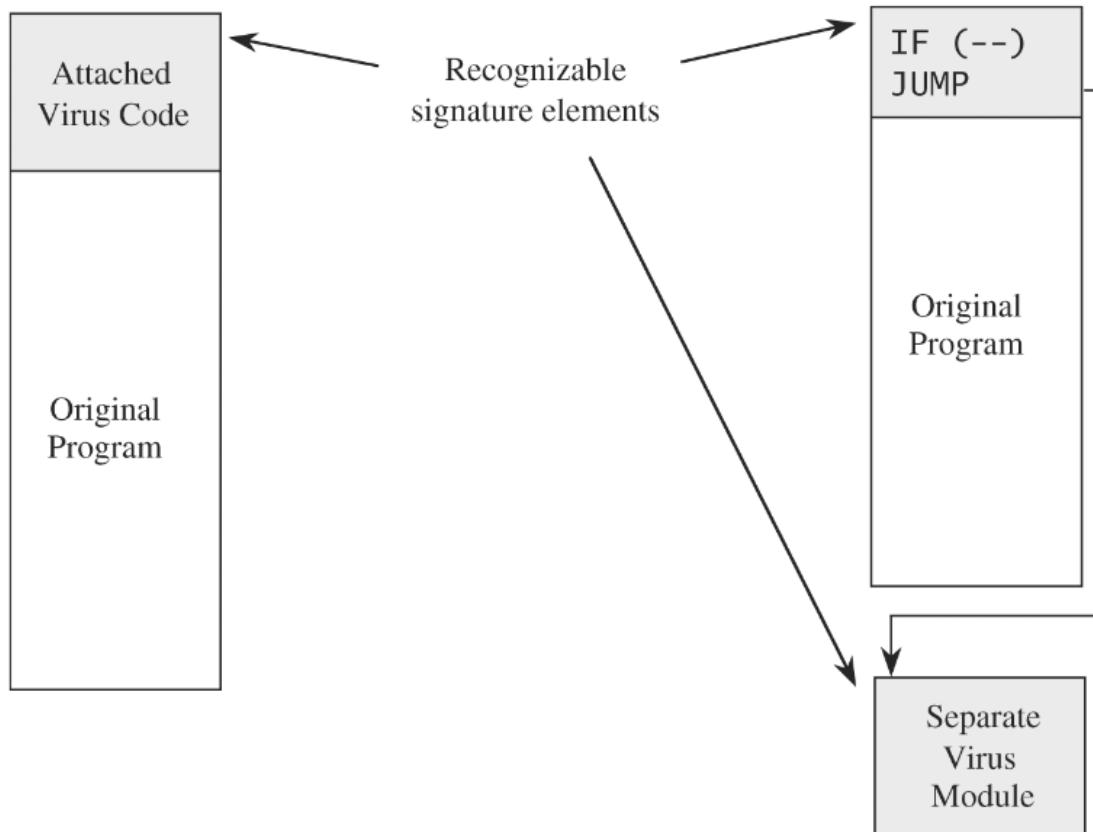
# How they work

- Virus (V) has to be invoked instead of target (T). Could be:
  - V assumes T's name by replacing or appending code in file structure
  - V overwrites T in storage
  - V changes pointers in file table to itself instead of T
  - V changes sequence that would have run T to run V

# First generation: boot virus

- Boot/file viruses
  - Virus code tacked onto end or start of file
  - Either way, stored somewhere on disk
  - Virus checker just had to have list of bytes (signature) to look for on HD

# Signature



# Second generation: encrypted

- Virus writers encrypt code
  - Every replication uses new key
  - New key = changed signature
- Problem:
  - Decryption code never changes!
  - Decryption routine becomes signature.
  - AV looks for decryption signature

# Third generation: polymorphic

- As gen 2, but contains a mutation engine
  - randomly mutates decryption routine
  - Decryption routine is always the same function
  - But extra bits added, or certain functions are re-ordered, numbers shuffled ( $2+3 = 1+4$  ect)
  - Gives same functionality, but different bit-pattern
  - Semantic equivalence; same functionality, different structure
- Result: no consistent signature for AV to look for

# Fourth generation: metamorphic

- Entire virus is rewritten on every infection
  - Polymorphic virus cannot rewrite own polymorphic engine
  - Creates logically equivalent program; works in a different way, gives same result
  - Some very sophisticated viruses rewrite themselves for different OSs

# Anti-virus: integrity checking

- Viruses must infect **existing** file
  - Achilles heel
- Basic approach; append hash code, checksum, ect to end of clean file
  - On infection, checksum becomes invalid
- Advanced approach: generate large key, store separately from file
  - Virus can't generate new checksum, as doesn't have key
- Problem: only works when file is known to be clean
  - What if new file comes pre-infected?

# Anti-virus: sandboxing

- Virus has to decrypt on execution
- So run program in virtual environment
  - Sandbox consists of CPU emulator, sig scanner, control module (stops virus from breaking out)
  - Virus decrypts, signature becomes apparent to AV
  - Sandbox is isolated from real machine, so safe
- Issues:
  - Slows machine
  - Clever virus will wait until decrypting
  - Won't catch metamorphic viruses (no sig)

# Anti virus: behaviour blocking

- Why look for the virus, when it's the payload that's dangerous?
- Host-based Behaviour-Blocking software looks for suspicious behaviour and stops it
  - Attempts to manipulate files
  - Attempts to format drive
  - Modification of executables
  - Modification of system settings
  - Network communications
- Closely tied to OS

# Worms

- Requires no host
- Propagates by itself
  - Acts as automated hacker
  - Exploits software vulnerabilities
- Aim: to spread across a network
  - Usually carries a payload

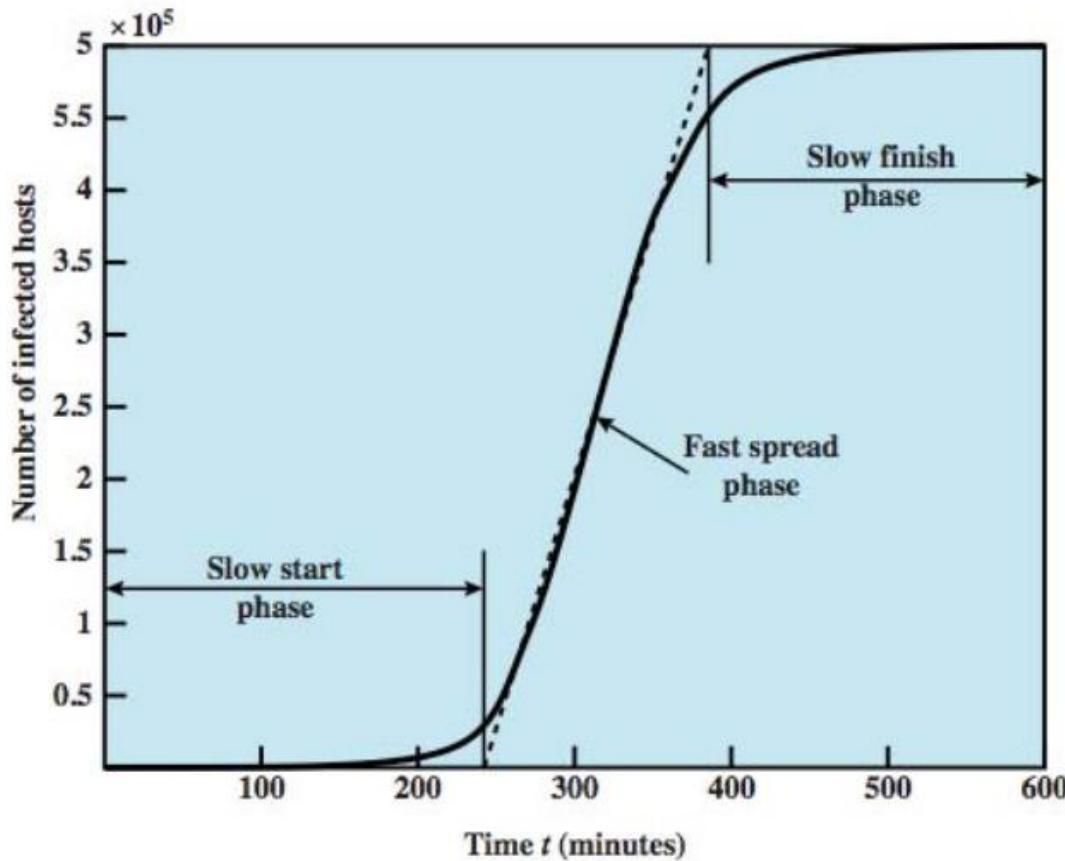
# Entrypoints

- Email/IM
- File sharing
- Remote file access
- Remote execution
- Remote login
- Web-facing services
- As payloads from other attacks

# Target discovery

- Equivalent to virus propagation phase
- Worm looks for systems to infect
- Scans and fingerprints remote systems
  - Just like real attacker
- Multiple scanning strategies
  - Random: hit random IP addresses
  - Hit-list: Work down a list
  - Topological: uses infected machine to find more hosts
  - Local subnet: uses subnet's address structure to scan

# Stallings propagation model



Moral: catch them when they're young!

# Modern worms

- Multiplatform
  - Can attack anything, even low-level DAQ equipment
- Multi-exploit
  - Come with many exploits, not one
- Ultrafast spreading
  - Spread as fast as possible
- Can also be poly/metamorphic, as viruses, ect

# Worm countermeasures

- Catch worm as it's propagating over network
- Filter-based worm containment
  - Scan messages for worm's sig
- Payload-classification based worm containment
  - Scans network packets
  - Looks for flow-control structures
  - Can cause false positives

# Worm countermeasures

- Rate limiting: aims to slow the worm
  - Limits scan-like traffic from host
  - Limits number of new hosts a host can connect to within a set window
  - Limits number of unique IP addresses a host can scan
- Rate halting
  - Looks for outgoing traffic threshold
  - Once exceeded, shuts down traffic

# Distributed intelligence gathering

- Sensors across network detect potential worm
  - Alerts central server
  - Server analyzes the incoming alerts, determines worm likelihood
- Suspect is shunted by server to controlled environment
  - Sandboxed
  - Tested using an instrumented version of software
  - Patch is auto-generated, tested in sandbox, distributed

# Trojan horse

- Program that appears to be useful, but contains hidden functions
- Do not require a host
- Do not self-replicate
- Relies on social engineering (ie lies) to propagate
- Often used by attackers for privilege escalation and backdoors

# Drive-by download

- Exploits bugs in user applications to install malware
  - Browsers + browser plugins
  - PDF readers
  - Outlook
- If user views compromised site, opens email, ect, is infected
  - Possible payload of a web-based attack

# Spam injection

- Automated attack on web server
- Could be self-contained or payload
- Aim:
  - Inject spam keywords and links into web server
  - Prevent webmaster from knowing
  - Swap legitimate ads with spammers
- Result:
  - Spammer gets ad money and rankings that should be yours!

# Payloads

- Most attacks have payload; some action carried out on the compromised system
  - Damage
  - Zombification/botting
  - Info theft
  - Backdoors/rootkits
  - Annoyances (opening/closing CD tray, changing background ,ect)

# Damage

- Data destruction
  - Delete files
  - Encrypt files for ransom
  - Randomly edit files
- Physical damage
  - Chernobyl virus; re-wrote BIOS, bricking PC
  - Stuxnet worm; interfered with control software, causing equipment to fail

# Zombies and Bots

- Malware places program onto machine
- Program turns machine into bot
- Machine then used in
  - spam mailing
  - DDoS
  - Traffic sniffing
  - Infecting further machines
  - Rigging online polls

# Information theft

- Malware leaves spyware on machine
  - Monitors key presses
  - Records browsing history
  - Redirects user to evil copies of legitimate pages
  - Captures passwords, email addresses, ect

# Backdoors and rootkits

- Backdoors
  - Secret entry into system
  - E.g. opens some network service on some weird port
- Rootkit
  - Gains admin access to system
  - As is admin, can hide own existence

# Kernel-mode rootkits

- Rewrites parts of the kernel
- Very difficult to detect
  - Operates below AV programs
  - Modifies commands before they get to the application layer
- Can infect startup code to get around full-disk encryption (bootkit)

# General Countermeasures

- Prevention
  - Best solution
  - Least practical
- Detection
- Identification
- Removal

# Prevention

- Reduce system vulnerabilities
  - Upgrade all software, patch, blah blah
- Harden system
  - Only allow programs/users to access what they require
  - Can slow infections + attacks
- Malware awareness
  - Educate users as to what to watch out for

# Detection and removal

- Anti-virus software (duh)
- Host-based detection (talked about earlier)
- Network-based detection
  - Perimeter security mechanisms
  - Third party monitoring
  - Intrusion Detection Systems

# Perimeter scanning

- AV on firewall and IDS
  - Scanning built into emails and proxy services
  - Traffic analysis component of IDS
- Ingress + egress monitors
  - Sits between trusted and untrusted network
  - Looks for incoming traffic to unusual IP addresses
  - Looks for suspicious outgoing traffic
    - Why is that machine looking at that known botnet C&C server?

# Case study: Sality

- First noticed in 2003-2004
  - Prepended host with executable
  - Collected data + emailed to attacker
  - Very basic; exfiltration address hard-coded in virus, payload not updatable
- Improved in 2004-2008
  - Polymorphism added
  - C&C server contacting capabilites added
    - Contacted one of three hard-coded servers
- 2008-2011
  - Distribution scheme improved
  - Virus was now peer-to-peer (!)

# Sality (cont)

- Five parts (all separately threaded):
  - Injector
    - injects Sality into running processes
  - Protector
    - protects from AV, security, etc
  - Infector
    - Propogates virus
    - Also attempts to damage security software
  - Downloader
    - Downloads additional malware from URLs supplied by P2P component
  - Peer-to-peer controller

# Sality P2P

- Virus carries bootstrap list of some infected peers
  - On first infection, contacts peers via UDP, records how they responded
  - Also contains signed URL list for malware
  - New malware URLs + executables distributed via peers
- No C&C server; virus user just needs keys to upload fresh URLs to Sality

# Conclusions

- Viruses and worms most dangerous
- Constant battle between malware and anti-malware writers
- Ongoing research area

# Next week: cryptography

(may contain maths)

# Cryptography

# The CIA

- Confidentiality
  - Not letting people who aren't allowed to see the information see the information
- Integrity
  - Preventing information from unauthorised modification
- Availability
  - Letting people who are allowed to see the information, see the information

# The cast

- Alice and Bob
  - Want to communicate securely without being overheard by...
- Eve
  - Bad actor who wants to eavesdrop on Alice and Bob
- In the real world, Alice and Bob could be browsers, routers, server and client, ect...

# Eve's objectives

- Evesdropping (sic)
  - Intercepting messages
- Inserting
  - Inserting fake messages into conversation
- Impersonating
  - Pretending to be Alice or Bob
- Hijacking
  - Replacing Alice or Bob mid-conversation
- Denial of service
  - Preventing service from being used by others

# Terminology

- Plaintext (**P**) (aka cleartext)
  - Unencrypted message
- Ciphertext (**C**)
  - Encrypted message
- Encryption
  - Turning **P** into **C**
- Decryption
  - Turning **C** into **P**
- Encryption scheme
  - Mathematical method of encryption and decryption

# Terminology

- Cryptography
  - The art and science of keeping messages secure
- Cryptanalysis
  - The science and art of breaking codes and ciphers
- Cryptology
  - The branch of mathematics covering the cryptography and cryptanalysis
  - **P** and **C** almost always represented as numbers (easier to work with)

# What cryptography is used for

- Cryptography directly protects:
  - confidentiality
  - integrity
- Plays a part in protecting
  - authentication
    - Proving identities
  - non-repudiation
    - Prevents denial of previous actions

# One approach

- Restricted algorithms
  - The algorithm is secret
  - Security is provided by obscurity of the algorithm

Example:

$P = FACE = 7 \ 1 \ 3 \ 5$

$C = 14 \ 2 \ 6 \ 10$

- Problem: once algorithm is discovered, security vanishes

# Key-based algorithm

- Some variable (key) is used as part of the algorithm
  - Key is normally very large number
- As key is shared secret, algorithm can be exposed safely
  - Exposure of algorithm allows checking for weaknesses by community
  - Also allows implementation and use by anyone

# Simple example

- Caesar cipher (key = n)
  - $P \rightarrow C$ : Replace each letter in  $P$  with the letter  $n \% 26$  places to the right
  - $C \rightarrow P$ : Replace each letter in  $C$  with the letter  $n \% 26$  places to the left
- Even if Eve knows that the Caesar cipher is in use, cannot decipher without the key
  - Well, not right away.

# Types of cipher

- Stream cipher
  - Ciphertext is produced bit by bit, byte by byte or char by char
- Block cipher
  - Ciphertext is produced from block of plaintext bits (usually 64 or 128) of same length
  - Empty spaces are padded
- Hash function
  - Takes variable length block and produces a fixed length value
  - Not necessarily a cipher, but very useful!

# Cryptographic techniques

- Substitution
  - Replacing symbols in plaintext with corresponding ciphertext
- Transposition
  - Shuffling the plaintext around
- Concealment
  - Scattering extra characters in the plaintext
- Modern techniques use all three

# Substitution

- Replace symbols in **P** with some corresponding symbol in **C**
- Simplest example: substitution cipher
  - Replace every letter with another letter
  - Vulnerable to frequency analysis attack
- Sophisticated example: polyalphabetic ciphers
  - Change the substitution scheme after every letter

# ENIGMA

- Mandatory ENIGMA example
  - Symmetric encoding system used by the Germans in WW2
- Keyboard connected to lights
  - Press a key, corresponding light illuminates
- Polyalphabetic substitution cipher
  - Every time a key is pressed, the substitution scheme changes

# Transposition

- Some rearrangement of the plaintext
- Example; simple columnar transposition cipher

ENS XSI AWN MEG AR5

E	X	A	M	A
N	S	W	E	R
S	I	N	G	5

EXAM ANSWERS IN G5

# Disadvantages

- Still vulnerable to frequency analysis
  - Expensive in terms of memory
  - Can be cracked by computer
- 
- However, can be strengthened by repetition!

# Concealment

- Mix message up with other symbols
  - Gives considerable additional security, but expands the message
  - EDADNVBRSBNT XMURSVERINAE  
AMAEWEHBNEFG MBRDEFBRGBAD  
AETBRRBA5BEJ
  - ENS XSI AWN MEG AR5

# Stenography

- Hiding a message in another channel
- NOT cryptography:
  - Cryptography conceals the content of the message
  - Stenography conceals the existence of the message
- Digital stenography
  - Can hide information in the least significant bits of image data

# Cryptanalysis

- Or; breaking ciphers for fun and profit
- Strong key-based algorithms cannot be broken by algorithm knowledge alone
- Therefore, attacker must recover
  - Plaintext or
  - Key (hence plaintext)

# What is ‘secure’?

- Unconditional security: absolutely unbreakable when implemented properly
  - One-time pad; use a key the length of the message
- Computation security
  - Encryption can be broken given sufficient time and effort
  - Trick is to make it prohibitively expensive

# One-time pad

- Alice and Bob exchange a set of randomly-generated disposable keys
  - Along with some pre-arranged order of use
- Use each key **only once**, and make the key the **same length as the message**.
  - Violating these two principles prevents the algorithms from being effective
- When used correctly, OTP cannot be broken.
  - Even if the attacker guesses the right key, they have no way of verifying it

# Cryptanalysis techniques

Depends on what Eve has access to:

- Ciphertext only attack
  - Brute-force; try every key until plaintext that makes sense appears
  - Statistical analysis
- Known-plaintext attack
  - Eve has known plaintext-ciphertext pair
- Chosen-plaintext attack
  - Eve can get the ciphertext for some chosen plaintext

# Statistical analysis attack

- Uses fact that letters in a natural language are not evenly distributed
- Relative frequency of letters is not affected by message
- Therefore key can be extracted by analysing structure of message
  - Say the bigram SB occurs in simple sub ciphertext
  - Can infer from following table that S=T and B=H

# Character and bigram occurrence



Bigram	Percentage	Bigram	Percentage
TH	3.15	HE	2.51
AN	1.72	IN	1.69
ER	1.54	RE	1.48
ES	1.45	ON	1.45
EA	1.31	TI	1.28
AT	1.24	ST	1.21
EN	1.20	ND	1.18

# Brute-force attack

- Systematically trying all keys in keyspace
- All but one algorithm vulnerable to it
- However:
  - Computationally expensive
  - e.g; 56-bit key, means a keyspace of  $2^{56} = 7200000000000000$  possible keys
  - SSL (common algorithm) uses 128 bit keys.

# Examples

- Alphabetic Caesar cipher:
  - Key space size: 26
  - Small key space; vulnerable to brute force attack
- Substitution cipher
  - Key space size:  $26! = 2^{88}$
  - Brute-force resistant

# Types of key crypto

- Symmetric key
  - Uses one key for both encryption and decryption
- Asymmetric key
  - Uses two keys, one for encryption and one for decryption
- Hash functions
  - Doesn't encrypt message or even require a key

# Symmetric key crypto

- Alice and Bob pass some key  $K$  to each other over a secure channel
- $K$  is used to encrypt and decrypt message
- Two flavours; stream and block
  - Stream: Performs operations on individual bits
  - Block: Performs operations on discrete, equally-sized blocks of plaintext

# Example: AES

- Advanced Encryption Standard
- 128-bit block cipher
- Key size of 128, 192 or 256
- Makes use of the Rijndael cipher family
  - Repeated substitution, permutation and key xoring
  - Premise; changing 1 bit of **P** produces massively different **C**

# AES (cont)

- Bytes represented as matrices
- Algorithm is repeated a set number of rounds
  - Each round generates a new key from the master key
- Each round
  - Each byte is substituted with another according to a lookup table
  - The rows and columns of the matrices are transformed
  - The result is xor'd with the round key
- To decrypt, just do steps backwards.

# Issues

- Key must remain secret
  - if key is found, all messages are exposed
- Compromised key can lead to false messages
- Separate keys must be used for each pair of users
- Must be implemented correctly; see right...

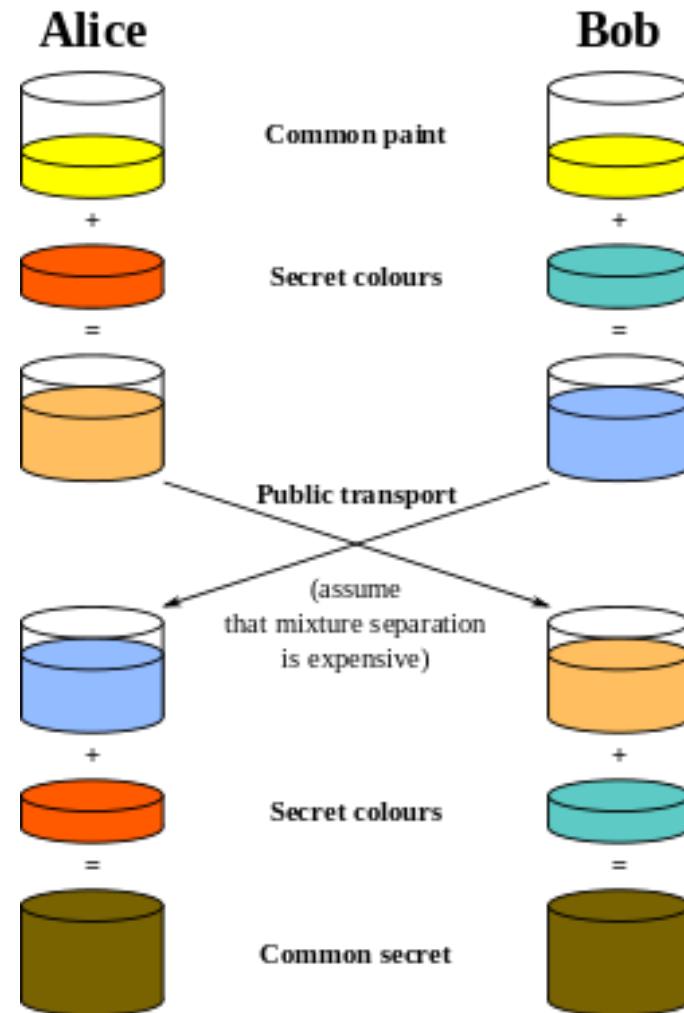


# Key exchange problem

- If the key is captured, the session is compromised
  - So, send the key over a secure channel
  - But how do you send the key to secure that channel?
- How does Alice send Bob the key?
  - Diffie-Helman
  - Asymmetric encryption

# Diffie-Helman

- Means of exchanging a private key over a public channel
- Mix a secret with a public fact, and send the mixture
- At other end, mix other secret
- This gives a common key

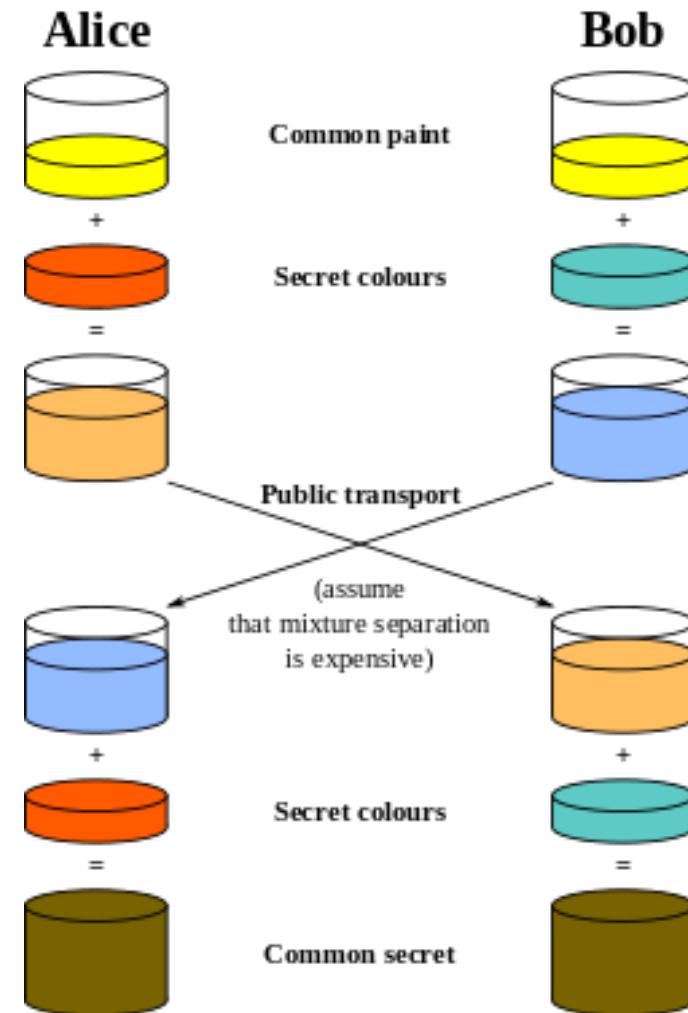


# Diffie-Helman

- Based on the Diffie-Helman problem
  - For a group  $g$  with members  $g^x$  and  $g^y$ , compute  $g^{xy}$
- In practice, Alice and Bob agree in advance on:
  - Some large prime  $P$ , to use as the modulo value for the group
  - Some generator  $G$  that is a primitive root modulo of  $P$ 
    - for every integer  $a$  coprime to  $p$ , there is an integer  $k$  such that  $g^k \equiv a \pmod{p}$
    - This ensures that the secret can be any number between 1 and  $p-1$ .
- System works because
  - $A^b = g^{ab} = g^{ba} = B^a$  (*all under mod p*)

# Diffie-Helman

- Alice and Bob randomly pick a secret number ( $a$  and  $b$ )
- Alice sends Bob  $A = g^a \text{ mod } p$ 
  - Bob does the same with  $b$
- Alice then computes  $s = B^a \text{ mod } p$ 
  - Bob does the same with  $A^b$
- As  $A^b = B^a$ , both know  $s$ .



# Public key cryptography

- Asymmetric algorithm
  - Public key known to anyone
  - Private key known only to message receiver
- Messages can be encrypted with public key but only decrypted with private key
  - No secure channel needed; it doesn't matter if Eve knows the public key!

# Public key example

- RSA (Rivest, Shamir, Adleman)
  1. Generate two large ( $>100$  digits) prime numbers,  $p$  and  $q$ . Define  $n = p \cdot q$ , and make  $n$  public
  2. Select **public key**  $e$  such that  $e$  has no common factors with  $(p-1).(q-1)$
  3. Compute **private key**  $d$  such that
$$e \cdot d \bmod ((p-1)(q-1)) = 1$$
  4.  $C = P^e \bmod n$  (for  $P < n$ ,  $P$  being an integer)
  5.  $P = C^d \bmod n$

# RSA continues

- RSA designed in 1977, still going strong
- Two approaches to brute force RSA:
  - iterate through the private key  $d$ , or
  - factor  $n$  to get  $p$  and  $q$ , and thus  $d$
- Therefore, make sure  $n$  is large enough to not be factored
  - Also; make sure that the RNG used to generate  $p$  and  $q$  is random enough

# Comparison

- Diffie-Helman
  - Provides a key only
  - Computationally cheap
  - Provides no inherent authentication
    - As secret  $a$  is not shared, there is no proof that Alice is who she says she is
- RSA
  - Can encrypt entire messages
  - Provides authentication
    - Messages cannot be decoded without
  - Computationally expensive

# Session keys

- Bob and Alice use RSA to exchange a symmetric key
- This **session key** used to encrypt rest of conversation
- Session key can be changed to enhance security

# Key security

- If Eve can brute-force session key fast enough, above is worthless
- Therefore, make sure key is big enough!
  - 32 bits; approx. 4 billion values; could do on this PC
  - 40 bits; > 1 trillion values; still trivial
  - 56 bits; 72 quadrillion value; cracked in 22 hours and 15 minutes
- Past 2020, 100 bits (symmetric) or 4096 bits (asymmetric) recommended minimum length

# Summary

- The CIA triad
  - Confidentiality, Integrity, Accessability
- Three basic crypto functions
  - Substitution
  - Transposition
  - Concealment
- Symmetric and asymmetric systems
  - Remember how these work!
  - Especially asymmetric!
- <http://article.sapub.org/10.5923.j.algorithms.20120105.01.html#Sec3>

Next week; authentication

# Validation and Storage

# Authentication

- Strong encryption makes sure that a message is not intercepted and read
- But how do we know who wrote it?
- How do we know that people are who they say they are?

# User authentication

- Something someone knows
- Something someone has
- Something someone is
- Something someone does

# Things the user knows

- Username/password, PIN, ect
- Password equivalent to shared secret
- Both user and system need to know password
- Password must be transmitted without interception
- Advantantages:
  - Easy to implement and use
  - Cheap!
- Disadvantage
  - Easy to lose, forget, steal

# Things the user has

- Dongle, smart card, key
- Advantages:
  - Very secure algorithms
  - Doesn't rely directly on human memory
- Disadvantages:
  - Expensive – requires specialised hardware
  - Things the user has can be taken away from them!

# Something the user is/does

- Biometrics
- Some unique property of the individual
  - Fingerprint, iris, face recognition, voiceprint, DNA
- Advantages:
  - Very difficult to forge
  - Can't be forgotten or stolen
  - For typing/mouse behaviour signature, can provide continuous authentication
- Disadvantages:
  - If faked/forged, cannot change
  - Same key is used for everything
  - Unlike the rest, not 100% effective

# Access control

- Don't want to give everyone access to everything on a system
  - Good security; give users access to nothing more than they need to fulfill their roles
  - Make sure users have appropriate rights and permissions
- Permissions: access to resources
  - Eg access to certain files/programs
- Rights: Authorization to do things
  - Eg power to edit other users rights and permissions

# Access control list

- Table defining access rights attached to each object
  - Tells OS which users can access what files
  - On request, system checks ACL
- Can be
  - individual user based
    - Joe can read file x, but not y
  - Role/group based
    - Testers can edit this file, but not that one
  - Access level based
    - Each user has a clearance level, restricts access accordingly

# DAC/MAC

- Discretionary access control
  - Admin provides users with initial p
  - User has complete control over programs it owns and executes
  - User can pass permissions for owned programs onto other users
- Mandatory access control
  - All rights controlled by administrator
  - Users have no power to transfer permissions



# Message Integrity

- Verifying that:
  - Message content has not been tampered with
    - Hashing
  - Message is from where you think it is
    - Message Authentication Code
  - Message is not a recording of a previous message
    - Nonce
  - Message is in sequence with other messages
    - Sequence counter (next lecture)

# Hashing

- Hash function: takes **arbitrary-length** input (message) and produces a **fixed-length** output
  - Irreversible; cannot compute the input given the output
  - Many-to-one; there may be many inputs that produce the same output
  - Easy to calculate
  - Output should be not related to input

# Good hashing

A good cryptographic hash should have:

- Pre-image resistance
  - It should be infeasible to find the original message from the hash
- Second pre-image resistance
  - Given some input  $m_1$ , it should be infeasible to find another input  $m_2$  that gives the same hash
- Collision resistance
  - It should be infeasible that two different messages produce the same hash (a ‘collision’)

# Use of hashes

- Tamper detection
  - Sender calculates hash, appends to message
  - On receipt, receiver also calculates hash
  - If they match, message has not been disturbed
- What if attacker also tampers with hash?

# Message Authentication Code

- Combine message with pre-shared secret, then hash to produce MAC
  - Same process at other end; take message, combine with secret and hash, then compare MACs
- Shared secret verifies identity of sender and recipient
  - Also protects against hash computation

# Replay attack

- MAC verifies sender, but not time of sending
- Replay attack: attacker re-sends a recorded message
  - Attacker does not need to know the content of the message to resend it
- Could timestamp, but a fast replay can ‘hide’ in lag

# Cryptographic nonce

- When a request is sent, a random Number Used Once is sent along with it and stored
- This is hashed with the key and message
  - So full hash function is  $h(\text{message}, \text{key}, \text{nonce})$
  - If a message that uses the same nonce is detected, it's a replay attack

# MAC limitations

- Usual key exchange problem
- Not scalable
- Does not provide non-repudiation
  - As symmetric key is used, Alice cannot prove Bob created a message

# Digital signatures

- Goal of authentication; to prove that only one person could have generated that message
  - In encryption terms, many can decrypt, but only one can encrypt
  - Essentially RSA ‘in reverse’
- If I encrypt a message with my private key
  - (That only I know)
- It must have been me that wrote it!
  - You can check (authenticate) by decrypting with my public key

# Signing document with RSA

- Assume private ( $e$ )/public( $d$ ) key pair and  $n$  already generated
- To ‘encrypt’
  - $C = P^e \text{ mod } n$
- To ‘decrypt’
  - $P = C^d \text{ mod } n$
- (As decryption key is widely known, useless as actual encryption)

# What does this gain?

- Authenticity
  - You all know I wrote the message, as you couldn't decrypt it with my private key otherwise
- Forgery-proofing
  - Only I know my key
- Not reusable
  - Signature can't be transferred to another document

# What does this gain?

- Tamper-resistance
  - If the document is altered en-route, my key will not decrypt it
- Non-repudiation
  - I cannot deny I wrote it; it's signed with my key!
- Weaknesses:
  - No protection against repeat attacks
  - Slow and expensive

# Combining

- Encrypting entire message just to prove identity inefficient
- So: hash document, then encrypt hash!
  1. I write my document and hash it.
  2. I encrypt (sign) the hash with my private key
  3. I send the document on to you
  4. You
    1. hash the document
    2. decrypt the attached hash
    3. Compare the two
  5. If they match, I sent it!

# One more problem

- Someone could come along, say they're me and then just use their own private/public key pairs
- Need some way of tying keys to entities
  - Some sort of 'third party' that isn't me or you

# Certificate Authorities

- Entities that independently verify that key holders are who they say they are
- Issue digital certificates containing the public key and identity of the owner
  - Certificate is signed using CA's private key, so can also be verified
- CAs are considered to be trustworthy by all parties in communication



# Storage

- Hypothesis: With proper communication and validation in place, data is totally safe.
- Wrong:
  - Sysadmin may have missed something
  - New zero-day exploit may be revealed in software
  - Lead-pipe cryptanalysis may be used
  - Sysadmin could be source of breach!
- Secure storage is required

# Storage of data at rest

- Data not in transit or use is at rest
  - Already covered in transit
  - Data in use must be unencrypted
- Confidential data should be encrypted when at rest.
  - If stolen, thief still has to break encryption
  - Who should have the keys?
- Above assumes structured data

# General data controls

- Full-disk encryption
  - Comprehensive approach
  - Encrypt entire storage device
  - Data decrypted by OS as required
  - Unencrypted partition required for boot + authentication
- File-system encryption
  - Encrypt files + folders, but not metadata
  - Unauthorised person can see filenames, ownership, permissions, ect but not content
- With both, applications see unencrypted data

# Structured data controls

- Data encryption
  - Data is stored encrypted, but as normal data files
  - Passed to another application to decrypt
  - From db's point of view, just a big chunk of data
  - Data is secure until explicitly decrypted
- Partial/full encryption
  - Certain rows/columns are encrypted (partial)
  - All data encrypted (full)
  - Either case, dbms performs auth and decryption before passage to application
  - Data is only secure when at rest in database

# Password security

- Password; confidential identifier for validating a user
- Ideally only the user should know the password
  - So can't be stored in plaintext.
- Solution: hash the password, store the hash
  - When validation is required, hash the user input +compare to stored hash
  - Hash must be broken before password can be used

# Pre-computed hash attack

- Scenario: attacker has obtained table of hashed passwords
  - Want to obtain plaintext passwords
  - Could brute-force every single pass, or...
- Could pre-compute hashes for a range of common passwords
  - Birthday paradox; much easier to find two matching things when you don't care what the two things are, just that they match

# Hash-chains

- Practical pre-calc tables very large. Instead:
  1. choose + store random set of initial passwords  $P$
  2. Pick some reduction function that can produce a possible password from a hash
    - Eg, take the last  $n$  letters of the hash
  3. Repeatedly hash and reduce  $P$  some number of times, store the endpoint  $E$

# Hash-chain (cont)

- To crack a hash  $H$ :
  1. Apply  $R$  to the hash
  2. Check the result against  $E$
  3. If a match is found, go to the corresponding  $P$  and hash-reduce until  $H$  is found. The preceding plaintext is the password!
- Issue; if two different  $P$  values produce the same hash, the chains will merge.

# Rainbow table

- Instead of using a single  $R$ , use a set of reduction functions  $R_1, R_2 \dots R_k$ 
  - Prevents merging, as  $R_1(\text{hash}) \neq R_2(\text{hash})$

To crack  $H$

1. Reduce  $H$  using  $R_k$ , check if it appears in last column
2. If not, reduce  $h$  using  $R_{k-1}$  then  $R_k$ , ect
3. If  $R_k(H)$  is present, go to start point and work along chain until plaintext is found

# Defeating rainbow tables

- Pre-computation attacks dependent on stored hash relying only on plaintext
- Solution: salting
  - Salt: stored random string for each hash
  - For each salt, a new table must be generated
  - Exponential increase in complexity of attack
- Precomp attacks still valid if salt is poor!
  - Insufficiently random
  - Too short
- Also; make sure that the salt is properly mixed in
  - Poorly combined salt can be removed

# Review

- User authentication
  - User based, role based DAC, MAC
- Message authentication
  - MAC
  - Digital d
- Storage
  - Store passwords as salted hashes

# Final lecture: securing the network

# Protecting the Network



# Today's lecture

- HTTP authentication
- Transport Layer Security
- Firewalls
- Intrusion Detection Systems

# HTTP security

- HTTP: text based
  - Exposed to interception
  - No encryption, etc
- HTTP authentication mechanisms
  - Basic:
    - Base64 encoding
    - Obfuscates, not encrypts
  - Digest
    - Hashes credentials with MD5

# HTTP Basic

- Web browser sends username+password using base64 encoding
  - Sent in request header
  - Authorisation:Basic username:password
- This header sent in every subsequent request
- Security through obscurity
- May as well be plaintext

# HTTP digest

- Hashes password using MD5
  - Hashes password, URI, timestamp, nonce
- Problems:
  - MD5 is a poor hash
  - Also vulnerable to dictionary attacks

# Transport Layer Security

- Widely used security protocol
  - In use in some form since 1993 (SSL)
  - Originally designed for e-commerce
- Fulfills the CIA triad
- Available to any TCP application
  - Sits between the TCP/IP layer and the application layer

# Requirements

- Want to send byte streams and interactive data
- Want a set of secret keys for the entire connection
- Want to exchange certificates as part of protocol

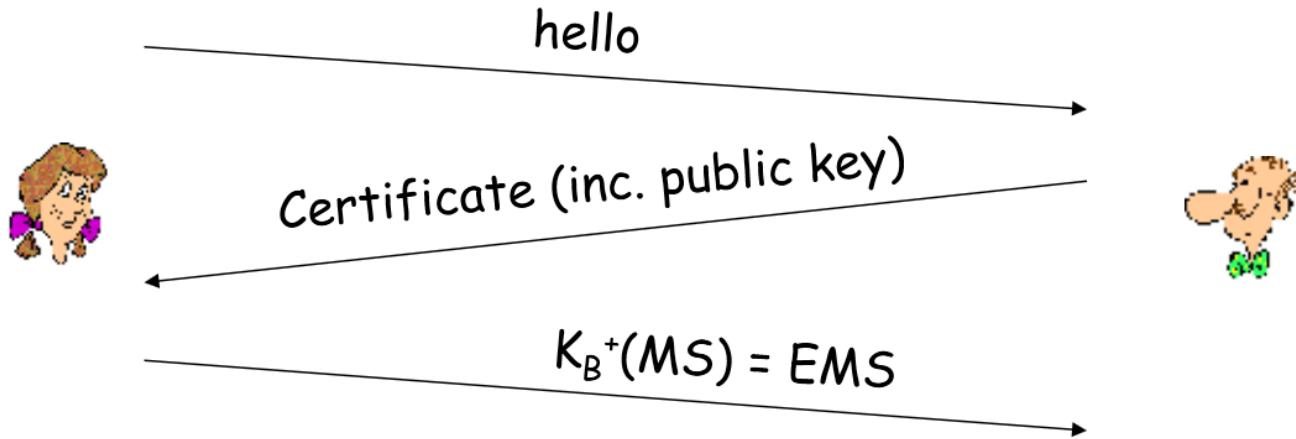
# Baby's first TLS

- Handshake
  - Alice and Bob exchange master secret key using asymmetric crypto
- Key derivation
  - Master key used to generate set of session keys
- Data transfer
  - Data is broken into records for travel
- Connection closure
  - Special message is sent to ensure proper closure

# 1. Handshaking

- Alice says 'hello'
- Bob sends Alice his certificate
  - Certificate contains public key
  - Also contains CA key (to prove Bob is Bob)
- Alice generates a Master Secret Key (MS)
  - Encrypts using Bob's public key
  - Sends to Bob

# In diagram form:



- MS: Master Secret
- EMS: encrypted master secret

### 3. Data transfer

- Break data into records
- Generate MAC using  $M_c/M_s$ (depending)
- Append MAC
- Encrypt record and MAC



## 2. Key derivation

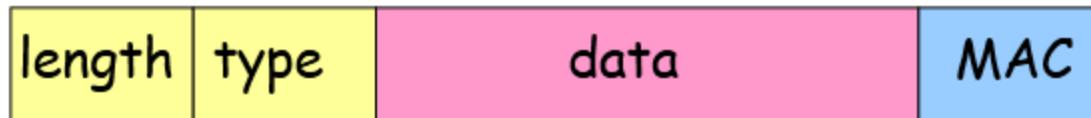
- Could use MS as symmetric session key
  - Bad practice to use same key for more than one crypto operations
  - The more a key is used, the easier it is to break
- Use MS to generate four keys
  - $K_c$  = Data encryption key from client to server
  - $M_c$  = MAC key for client to server
  - $K_s$  = Data encryption key from server to client
  - $M_s$  = MAC key for server to client
- Keys derived from Key Derivation Function

# Problem: replay attacks

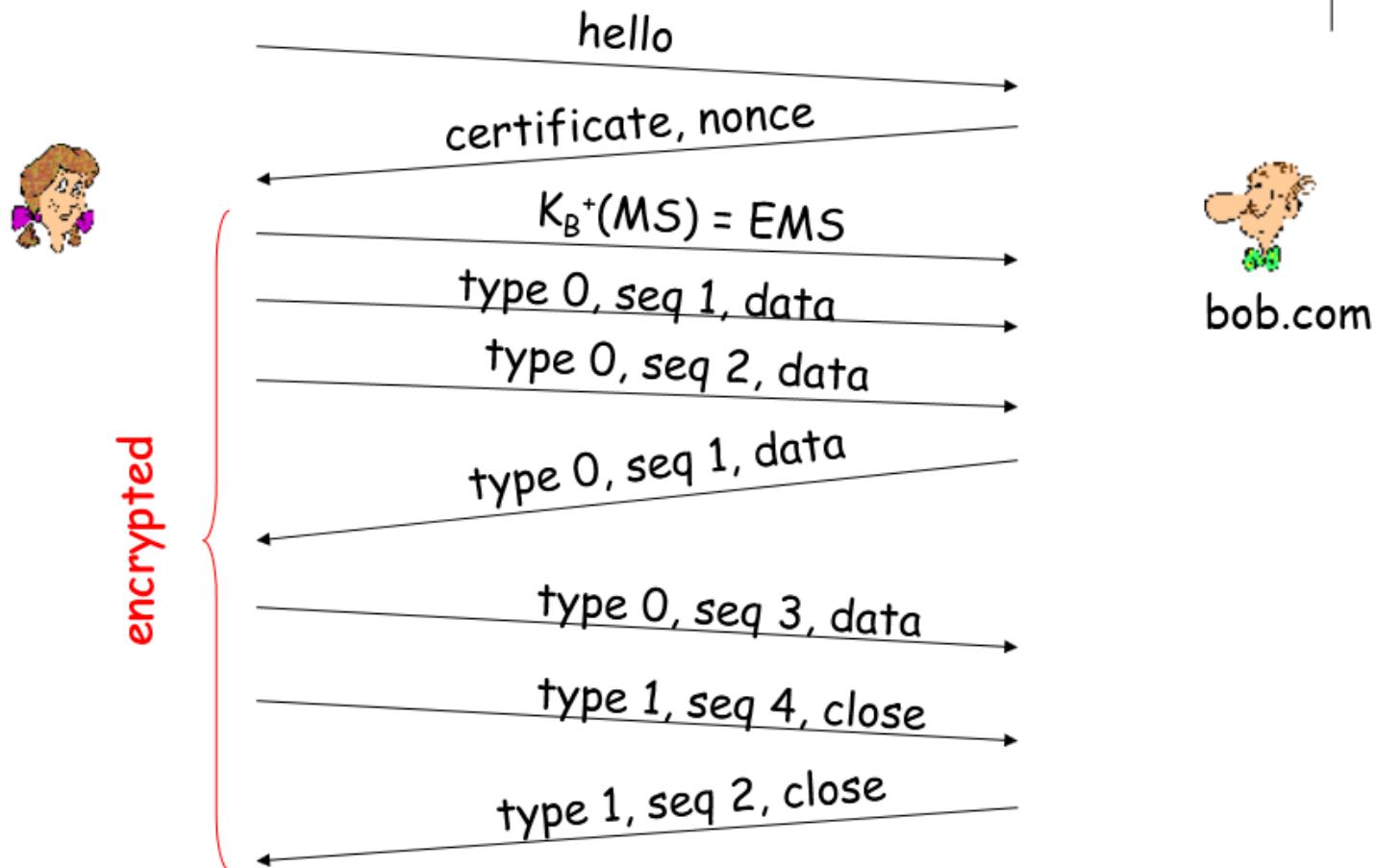
- Attacker can capture and re-order records
- Solution: hash sequence number into MAC
  - $MAC = \text{hash}(M_c, \text{sequence}\#, \text{data}, \text{nonce})$
  - Each party keeps track of sequence numbers

# 4. Connection closure

- Problem: Truncation attack
  - Attacker forges TCP close segment
  - One or both parties think there is less data than there actually is
- Solution: record types
  - Type 0: data, type 1: closure
- $MAC = \text{hash}(M, \text{sequence}\#, \text{type}, \text{data})$



# In a diagram:



# Issues with this approach:

- Assumes client and server have pre-agreed many things
  - Field length
  - Encryption protocol
- In real TLS, negotiation required
  - Allows clients and servers to support multiple algorithms
  - Common algorithm is negotiated before data transfer

# Real TLS

- Any URL with `https://` tells browser to engage TLS
- Uses port 443, not port 80
- Supports a variety of cipher suites
  - Symmetric encryption algorithm
  - Asymmetric encryption algorithm
  - MAC algorithm
- On connection establishment
  - Client offers choice of available suits
  - Server picks one

# TLS connection and session

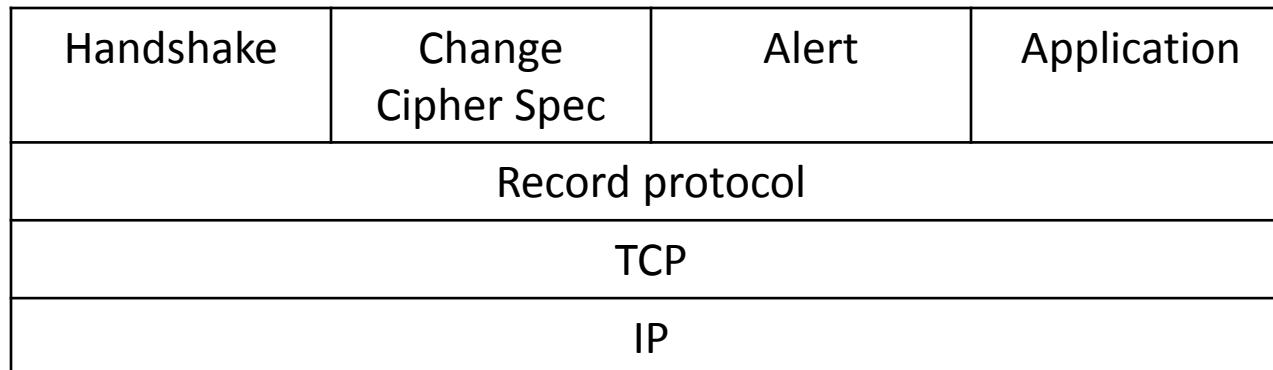
- TLS connection
  - Logical client/server secure link
  - Transient
- TLS session
  - Association between client and server
  - Defines a set of cryptographic parameters
  - Encompasses entire communication session, regardless of number of connections

# TLS session state

- Identifier
  - Arbitrary byte sequence chosen by server to identify state
- Peer certificate (maybe)
- Compression method
- Cipher suite choice
  - Also defines crypto attributes such as hash size
- Master secret
- ‘Is resumable’ flag
  - If true, this session can be used to initiate new connections

# TLS protocol architecture

- TLS actually has two layers
  - Handshake protocol, Change CipherSpec Protocol, Alert protocol
    - Manages exchanges
  - Record protocol
    - Provides basic security services to application



# Handshaking protocol

Purpose:

1. Server authentication
2. Negotiation
3. Key establishment
4. Client authentication (maybe)
  1. If so, client signature is sent at end of handshake

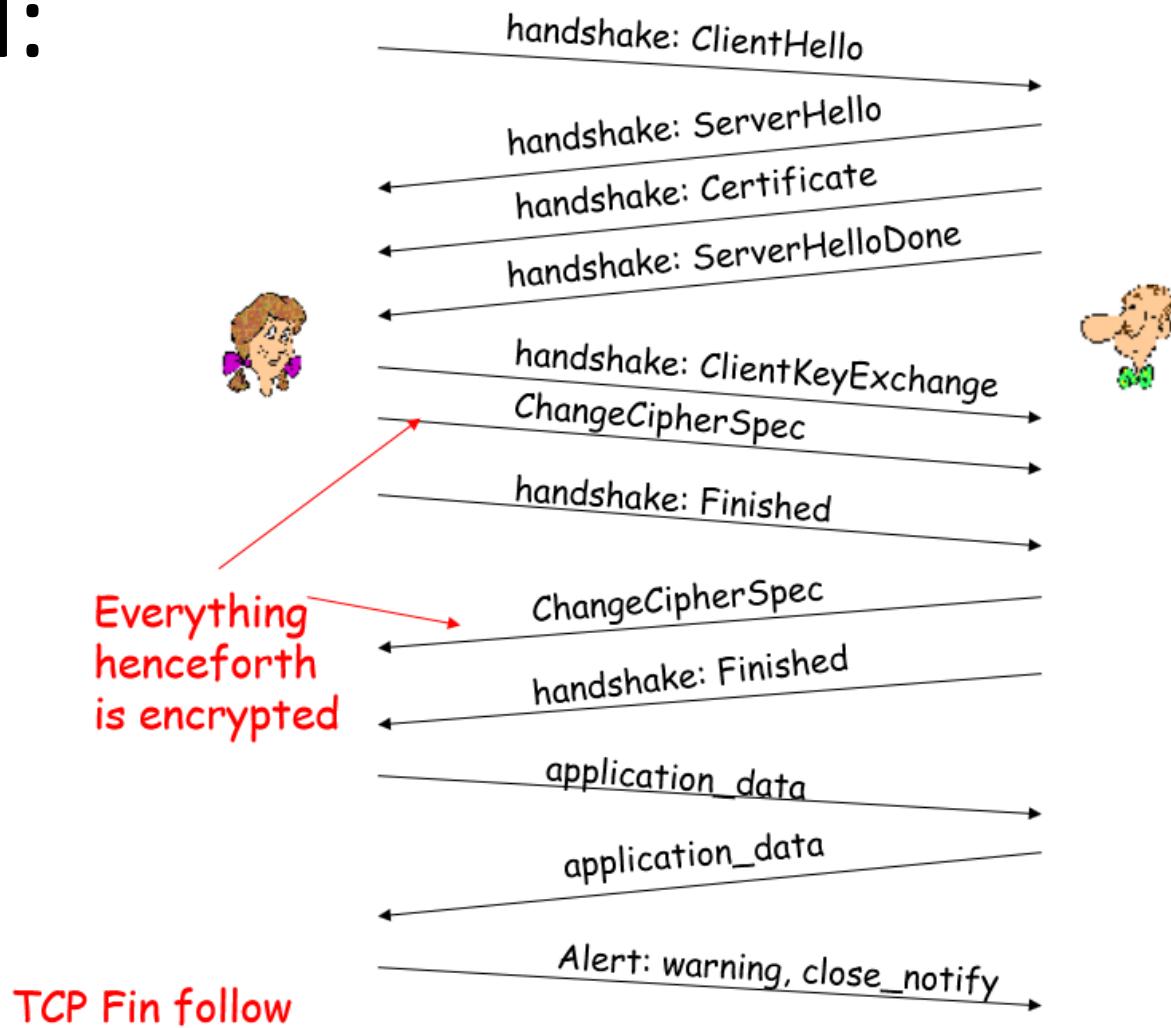
# Handshaking process

1. Client sends list of algorithms it supports and client nonce
2. Server chooses algorithm
  - Returns choice, certificate, server nonce
3. Client verifies certificate, generates some preMasterSecret; sends using server's public key
4. Both use preMasterSecret to compute keys
5. Client sends MAC of all handshake messages
6. Server sends MAC of all handshake messages
7. Client and server exchange ChangeCipherSpec messages

# Handshaking process

- Depending on cipher suite negotiated, there may be variations
  - e.g. server may generate PreMasterKey
- Steps 5 and 6 protect from man-in-the-middle attacks
  - MitM could remove strong algorithms from client list
  - MACs would not match

# In full:



# Change CipherSpec message

- ‘We will now start using the previously agreed upon cipher specification’
  - Single message with single byte, value 1
  - Used as a coordination system
- Client must send to server
- Server must return to client
- On client receipt, use the previously agreed ciphers and keys from then on

# Alert message

- Conveys TLS related alerts to other party
- Two levels:
- **Warning**
  - Notifies other party that connection may be unstable
- **Fatal**
  - Notifies other party of unrecoverable error or compromise in security

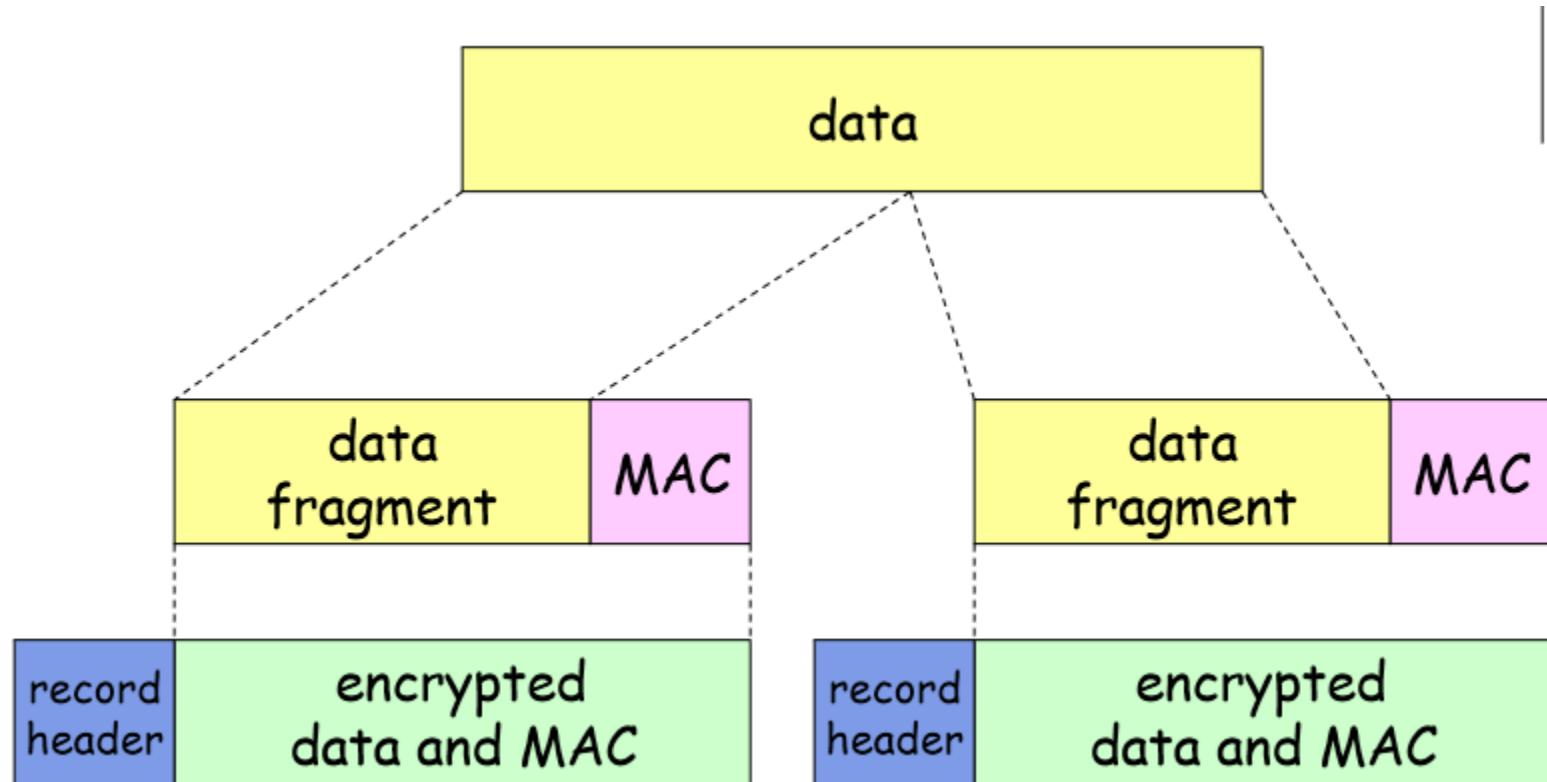
# Record protocol

- Sending
  - Gets application message
  - Fragments into blocks (~16 kbytes)
  - May compress data
  - Appends MAC
  - Encrypts data
  - Adds header
  - Transmits all fragments in TCP segment

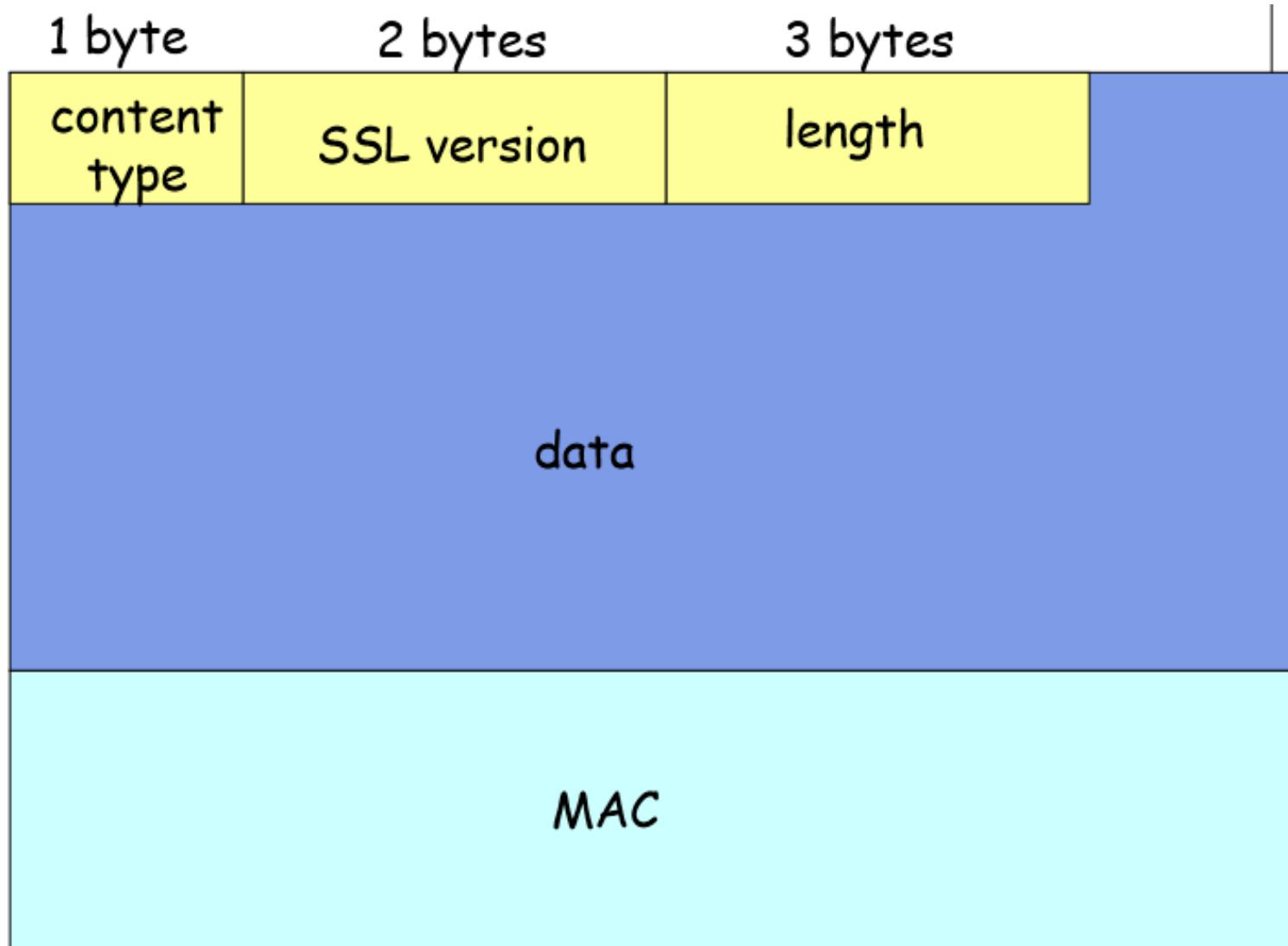
# Record protocol

- Receiving
  - Received data is decrypted...
  - ....verified...
  - ...decompressed...
  - ...reassembled
  - Resulting application data is passed to application

# Record protocol



# Datagram format



# TLS weaknesses

- Weak point: server certificate
  - Can be spoofed with relative ease
- FREAK
  - Old law limited RSA  $n$  sizes to 512 bits or less
  - Allowed easy breaking by NSA, but no-one else would have the resources.
    - (That's changed now, by the way)
  - MitM can force parties to use vulnerable cipher
- TCP/IP not encrypted
  - Sender and recipient still known to MitM

# Firewalls

- Isolates organisations internal network from Internet at large
  - Prevents DoS attacks
  - Prevents illegal access and modification of internal data
  - Allows only authorised access to inside network

# Firewalls

- Four techniques:
  - Service control
  - Direction control
  - User control
  - Behaviour control

# Service control

- Determines types of Internet services that can be accessed
  - Both inbound and outbound
- Filter according to
  - IP
  - Protocol
  - Port

# Direction control

- Controls direction of traffic and connections into and out of network
- Examples:
  - Ingress monitoring
    - Monitors traffic and connections coming into the network
  - Egress monitoring
    - Monitors traffic and connections going out of the network

# User control

- Controls access to service according to which user is attempting to access it
- Typically applied to users inside firewall perimeter
  - No online gaming in halls...
- Can also be applied to incoming traffic from external users
  - Requires some validation

# Behaviour control

- Controls how particular services are used
  - Spam filtering
  - Limits outside access to just a portion of a web server
  - Can prevent internal clients from accessing certain web servers

# Firewall types

- Three types:
  - Stateless packet filters
  - Stateful packet filters
  - Application gateways

# Stateless filters

- Internal network connected to internet via router/firewall
- Filters on a **packet by packet** basis
  - Has no ‘memory’ of what has gone before
  - Does not look inside packets, only at the headers
- Could be based on:
  - Source/destination IP
  - Port number
  - ICMP message type
  - ...

# Stateless examples

Policy	Setting
No incoming TCP connections except for the web server	Drop all incoming TCP SYN packets to any IP except 130.207.244.203:80
Preventing network from being used in a smurf attack	Drop all ICMP packets to broadcast addresses
Preventing students in halls from playing WoW	Drop all outgoing packets on port 3724

# Access Control List

- Table of rules applied top to bottom

action	source address	dest address	protocol	source port	dest port	flag bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----
deny	all	all	all	all	all	all

# Setting up ACLs

- Blacklisting
  - List of banned websites/resources/rules for firewall to reject
  - Assumes everything is fine unless otherwise stated
- Whitelisting
  - List of accepted websites/resources/rules
  - Assumes evil until otherwise stated

# Stateful filtering

- Stateless filtering admits ‘nonsensical’ packets
  - e.g. TCP ACK packet when no SYN/ACK has been received
- Stateful packets track TCP connections
  - On receipt of SYN packet, record connection as established
  - On receipt of FIN packet, record connection as closed
- Only accept packets on established connections

# Connection Table

- Table maintained by firewall of active connections
  - Records state of every TCP connection
- Used to see if packets make sense

Source	Dest	S. Port	D. Port
222.22.1.7	12.13.14.15	12699	23
222.22.93.2	199.77.240.48	48712	23

# Stateful filter issues

- Requires a little more memory than stateless filters
- Cannot examine app-layer data
  - So cannot prevent application-layer attacks
- Cannot support advanced user authentication schemes
  - Only understands connections and packets
- Cannot detect IP layer spoofing
- Easy to misconfigure

# Application gateways

- Filters based on application data, not just IP/TCP/UDP headers
- Acts as proxy server between host and destination
- Clever enough to inspect application data
  - Can see viruses, intrusions, policy violations

# Example: Telnetting

1. Require all telnet users to telnet through gateway
2. Gateway sets up telnet connection to destination
  - user<->firewall<->host
3. Firewall blocks all telnet connections not originating from gateway

# Firewall limitations

- IP spoofing: firewall router doesn't know where packets really come from
- Each app needing special treatment requires own application gateway
- Outside machines have to know about gateway
- UDP: require all-or-nothing policy
- You can't blacklist every threat!
- Tradeoff: **Communication vs security**

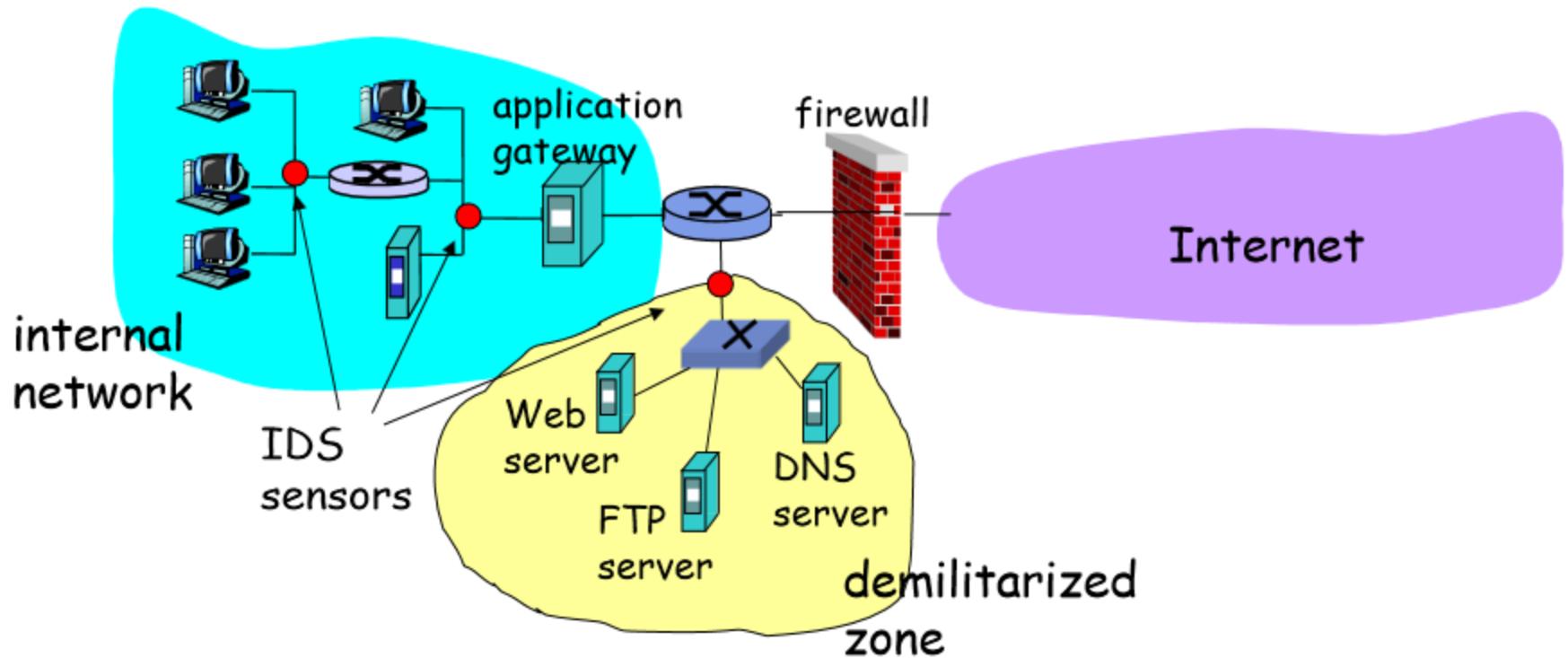




# Intrusion detection systems

- Sensors distributed throughout network
  - Feed back to some central server
- Performs deep packet inspection
  - Looks for, say, known virus attack strings inside packets
- Examines correlation among hits
  - Uses some heuristic to determine if attack is happening
  - May take appropriate action

# Illustration



# Bringing it all together

- Two types of attacker
  - Individual
  - Malware
- Same techniques, different motives and targets
  - Malware: wants to propagate
    - Blind attacks at a massive scale
  - Individual want something specific
    - Not in as many places at once
    - Much more flexible and adaptive

# Malware protection

- Antivirus
  - Must be kept up to date
  - Must not rely solely on signatures
- Firewalls
  - Can limit propagation post-infection
  - Application-level firewalls required to prevent infection
- TLS
  - Does nothing against malware
    - Encrypted malware is still malware
  - Protects communications from prying eyes
    - Malware has no eyes

# Individual protection

- Antivirus
  - Humans do not have signatures
  - Could detect attacker's tools during enumeration
- Firewalls
  - Packet filters easy to circumvent
    - Can deter casual attacks
  - High-granularity application firewall essential for real protection
- TLS
  - Strong defence
  - Can be compromised
  - Attacker needs to be determined to get through it.

Thankyou!

# Information Security revision

# The Plan

1. Whistle-stop tour of the course
2. Exam structure
3. Mock exam questions
4. Questions
5. Final words

# Introduction

- The CIA triad
  - Confidentiality
  - Integrity
  - Availability
- Four realms of security:
  - Physical
  - Network -----
  - Storage---We're mainly concerned with these two
  - OS

# Network architecture

- Application -> Transport -> IP -> Link
- Dynamic Host Configuration Protocol
  - Used to tell computers where networks are
  - New machine broadcasts DCHP discovery packet
  - DNS server returns first hop router
- Domain Name System protocol
  - Ties web addresses and IP addresses together
- Address Resolution Protocol
  - Used to associate IP and MAC addresses

# Attacking the network

- Port scanning:
  - FIN scanning:
    - if port is in LISTEN state (i.e. open), no reply
    - if port is closed, it responds with reset
  - SYN scanning:
    - If port is open, responds with SYN/ACK
    - You return RESET, no connection established

# Denial of service

- Denies resources to target
  - Network attacks: UDP Flood, TCP SYN Flood, ICMP Flood, Smurf IP attack
  - Nuke attacks: tie computer up
- Distributed denial of service attack
  - Uses multiple compromised machines
  - Sends too many legitimate requests for computer to cope with

# Web insecurity

- Buffer overflows
- Double encoding
- SQL injection
- XSS
  - Stored
  - Reflected
- Always sanitise user input!

# Malware

- Types of malware:
  - Virus: Modifies other programs
  - Worm: Standalone program that copies itself through networks
  - Trojan: Innocent code with malicious code hidden inside
- Types of virus:
  - Encrypted, polymorphic, metamorphic

# Malware (cont)

- Remember types of virus:
  - Encrypted: virus encrypts itself
  - Polymorphic; virus rewrites itself into a **semantically equivalent** form
  - Metamorphic: virus rewrites itself into a **logically equivalent** form
- Also; remember how to detect these!

# Cryptography

- Hiding information by means of a key and an algorithm
- Basic functions: Substitution, transposition, concealment
- Symmetric encryption
  - Messages are encrypted and decrypted by the same key
- Asymmetric encryption
  - Messages are encrypted using a public key and decrypted using a private one
- Use asymmetric to encrypt a symmetric key
- Algorithms to understand; RSA, Diffie-Helman, AES.

# Validation, authentication, storage

- Validation techniques:
  - knows, has, is/does
- Hashing:
  - Should not be reversable (pre-image resistance)
  - Given a message and a matching hash, it should be hard to pick another message that gives the same hash (second pre-image resistance)
  - It should be hard to pick two messages that give the same hash (collision resistance)

# Authentication (cont)

- Nonce; used to make sure an attacker can't reply a recorded message
- Message Authentication Code: used to say that a message is from who it says it is from
- RSA signing:
  - Hash of message is encrypted with private key
  - Message is decrypted with public key
- Certificate authorities

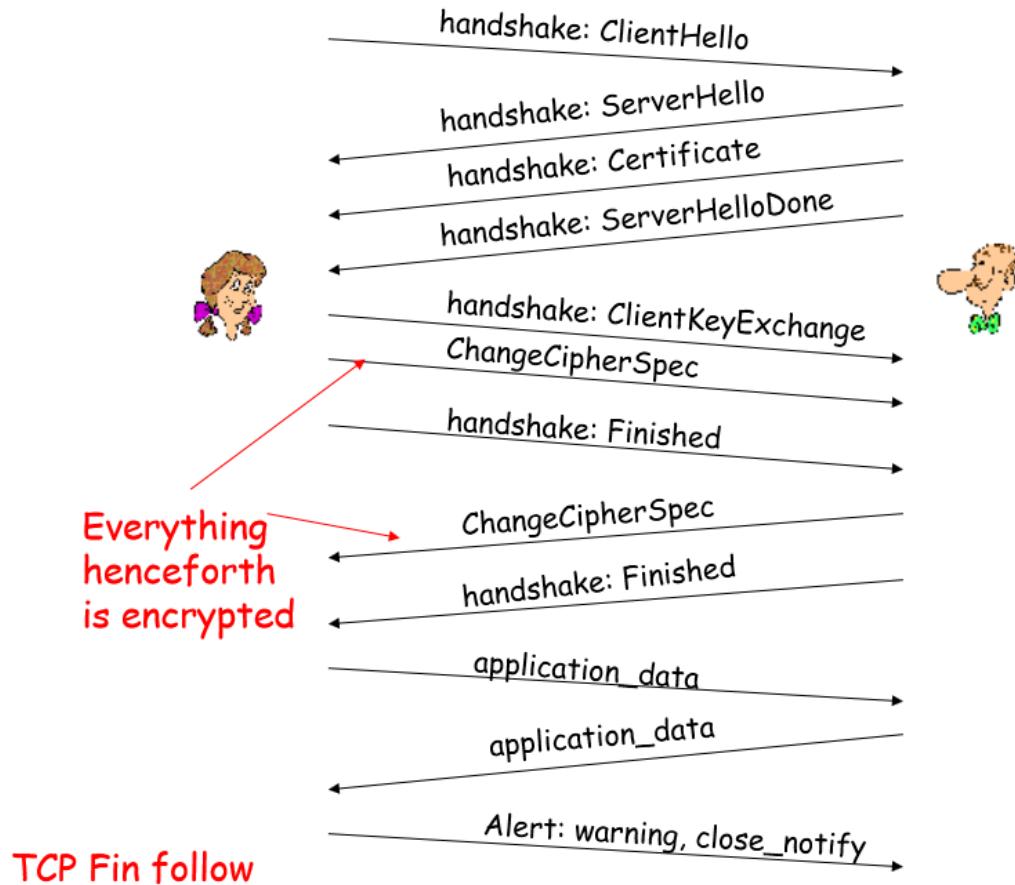
# Storage

- Encryption
  - Can encrypt entire disk
  - Can encrypt just filesystem
  - Can encrypt just data or parts of data
- Store passwords as hashes
  - Attacker can construct pre-hashed tables of passwords to reverse hashes
  - So store hashes with a random salt to prevent this

# Protecting the network

- HTTP security
  - Don't send confidential information through GET requests
  - Don't send confidential information unencrypted
  - HTTP's integrated security protocols are poor.
- Transport Layer Security
  - Applied asymmetric crypto

# TLS



# Network security

- Firewalls: Separates trusted and untrusted networks
  - Can be internal to networks; don't have to be on the edge!
  - Stateless, stateful and application
- Intrusion detection systems
  - Sensors distributed throughout network, looking for suspicious activity

# The exam

- 2 hours
- Choice of four questions: answer three.
- Questions will tend to a theme;
  - General purpose
  - Attack techniques
  - Web security
  - Encryption and hashing
  - Malware
- There may be some small overlap between questions
- No exam questions will require programming, although you may use it to illustrate your answers if you wish

# Example questions

- **State** two types of firewall **(2 marks)**
- Stateful and stateless
- **Define** two types of firewall **(4 marks)**
- A stateless firewall follows a set of rules for each individual packet; a stateless firewall remembers the state of every TCP connection that passes through it

# Example questions

- **Describe in detail** transport layer security, using an appropriate example. You may use diagrams (8 marks)

- The TSA algorithm is used to establish secure communication between two unfamiliar parties; for example, a new client talking to a bank. The client approaches the bank site with a set of algorithms that it supports; the bank site then responds with its security certificate....
- And so on.

# Example questions

- Describe **TWO** types of malware, and explain how they can be damaging to a business. (6 marks)
- Describe how, as an attacker with full control over all packets sent from a machine, you could mount a DoS attack on a remote host using the ICMP ‘ping’ command. Include an explanation of the mechanism of the attack. (6 marks)
- You have 15 minutes. Go!

# Mark scheme: Q1

- [1] mark for each malware name (max. 2), up to [2] for explanation
  - Virus; can corrupt files and cause loss of data OR carry payloads
  - Worm; can slow network traffic /carry payloads
  - Fork bomb; can bring down individual machines
  - Spam injection; can divert revenue away from business
  - Back door; can leave business vulnerable to further attacks

# Mark scheme: Q2

- Attacker sends ICMP ping packet [1 mark] with source IP [1 mark] set to that of the target [1 mark] to many other machines [1 mark]. This causes the other machines to send reply packets [1 mark] to the target [1 mark], impeding regular traffic.

Q and A

# Final tips

- Answer the questions with the amount of effort appropriate to the marks
  - Don't write paragraphs for a two mark question
- Manage your time well
- Don't be afraid to think outside the box!

Good luck!