CS 6262 Network Security Lecture Notes

Jie Wu, Jack

Summer 2023

1 Lesson 1: DDos Attacks

1.1 Introduction to Large Scale Attacks

Large-scale attacks include distributed denial of service attacks or DDoS and malware-based attacks.

1.2 Denial of Service Taxonomy Quiz One

hitlist scanning: a portion of a list of targets is supplied to a compromised computer

permutation scanning: all compromised computers share a common pseudo-random permutation of the IP address space

signpost scanning: uses the communication patterns of the compromised computer to find new target

random scanning: each compromised computer probes random addresses

1.3 Denial of Service Taxonomy Quiz Two

random spoofing: generate 32-bit numbers and stamp packets with them

subnet spoofing: generate random addresses within a given address space

fixed spoofing: the spoofed address is the address of the target

1.4 Denial of Service Taxonomy Quiz Three

infrastructure attack: the target of the attack is a crucial service of a global internet operation e.g. core router

server application attack: the attack is targeted to a specific application on a server

network access attack: the attack is used to overload or crash the communication mechanism of a network

1.5 Network DoS

One of the main approaches is **amplification**. The attacker sends a small number of packets to achieve a big effect such as rendering the site unavailable. There are two types of amplification attacks

DoS bug: exploit a bug or vulnerability on the server

DoS flood: command botnet to generate flood of requests

Network DoS can happen at any layer (application, transport, data link). Current internet is not designed to handle DDoS attacks.

1.6 Amplification Quiz

The reasons why the UDP-based NTP protocol is particularly vulnerable to amplification attacks are that

- a small command can generate a large response
- vulnerable to source IP spoofing—the attacker can send the request to a server by spoofing the IP addresses of the target
- it is difficult to ensure computers communicate only with legitimate NTP servers

1.7 Amplification Example

Amplification is accomplished by two factors: (1) the number of compromised bots involved (2) the server response is much larger than the requests.

1.8 TCP

From the security point of view the main weakness of IP is that there is no authentication of the source IP address thus an attacker can spoof an IP source address.

1.9 TCP SYN Flood I

SYN flood exploits the fact that server needs to keep in memory all clients' SYN handshake requests. The source IP address can be spoofed to some random target source IP address, thus the SYN/ACK packets may be lost and the server's memory is filled up with these SYN/ACK packets waiting for the ACK packet for them. As a consequence no further connection is possible (e.g. MS Blaster worm in 2003 launched SYN flood on port 80 to windowsupdate.com). Increasing the backlog queue size or decrease timeout is not a good solution because an attacker can simply send more packets or at a faster pace. A better solution is to use SYN cookies to remove state keeping from servers, which however brings in a small performance overhead.

1.10 SYN Cookies Quiz

SYN cookies do not require modified versions of TCP. They are only applied when there is a SYN flood attack thus would not slow down performance during normal operation. During SYN flood attack the server must reject all TCP options because it discards SYN queue entries.

1.11 SYN Flood II

It is very hard to filter the SYN packets from a SYN flood attack because they all look legitimate. One solution is to use a group of powerful intermediate servers as proxy, which will only forward established TCP connections to the real website. The proxy sends SYN/ACK packets in response to the SYN flood. When it receives ACK packets from legitimate clients it would then forward it to the real website.
However an attack can let the bots complete TCP handshake, which can then repeatedly send short HTTP head requests to the real website. With sufficiently many bots the attack can bypass the proxy to overload the real website. Nonetheless the attacker can no longer use random source IP addresses, because the TCP connection needs to be fully established thus the real IP addresses and locations of the bots would be revealed. Then the proxy can rate limit or block traffic from these bots. A real-world example is Github 2015.

1.12 Attack Quiz

With regards to a UDP flood attack, attackers can spoof the IP address of their UDP packets, which cannot be mitigated using firewalls because the packets involved in the attack all look legitimate. In addition the firewall itself is susceptible to flooding because it now needs to examine many packets.

1.13 DoS via Route Hijacking

The routing protocol can also be exploited to launch DoS. The Internet is divided into a large number of so-called autonomous systems or AS, each of which is responsible for logging packets in and out of a subset of the Internet defined by a prefix. If one domain declares a subset that covers an AS then all traffics that were originally correctly routed would be re-directed to that domain.

1.14 DoS at Higher Levels

DoS can also happen in SSL/TLS handshake because RSA decrypt is ten times more costly than RSA encrypt, i.e. the server has to do much more work than clients. Hence an attacker can send many such handshake requests to the server.
Similarly an attacker can send many simple HTTP requests for downloading a large file from the server.

1.15 DoS Mitigation

One remedy is to use client puzzles. The main idea is to slow down the attacker. For example find a number whose SHA-1's least n significant bits are all 0s. The server needs to compute the hash only once whereas the client needs to compute n times. It is implemented only during a DoS attack but also include legitimate clients. One advantage is that the hardness of the challenge can be adjusted based on the DoS attack volume. The shortcoming is that it requires changes applied to both clients and servers, and hurts legitimate clients with low computational power during attack.
A variant of client puzzle is to use memory-bound functions instead of CPU-bound functions, because the latter is hard to be scaled to very hard puzzles for low-end machines.

1.16 Puzzle Quiz

Client puzzles should not be difficult for the server to construct. Client puzzles should be stateless so that clients can find a solution even before being asked. Puzzle complexity should increase with the strength of the attack.

1.17 DoS Mitigation: CAPTCHAs

CAPTCHA stands for Completely Automated Public Turing test to tell Computers and Humans Apart. The idea is that the server should verify the connection is from a human rather than a bot or a malware. It is used for application layer DoS.

1.18 DoS Mitigation: Source Identification

The goal is to identify the source of attack packets so that we can block it. However the biggest problem for this proposal is that it requires all ISPs to implement ingress filtering, which requires global trust and incentive for deployment.

1.19 DoS Mitigation: Traceback

The goal is to determine the path to the source given a set of attack packets. The way to implement this is to change routers to record information in packets. The underlying assumption is that most routers remain uncompromised, and the attacker sends many packets and the route from the attacker to the victim remains relatively stable. A naive approach would be to have each router add its own IP address to the packet. However this requires too much space in packets if the path is long and changes to packet format would take too long to be implemented. A better implementation is to have each router store probabilistically its IP address, which occupies fixed space regardless of the path length.

1.20 Traceback Quiz

The traceback mechanism works regardless of whether attackers are aware of the tracing or not.

1.21 DoS Mitigation: Edge Sampling

The main component of the traceback mechanism is the edge sampling algorithm. An edge includes the start and end IP addresses and the distance from the start. When a packet arrives a router tosses a coin

head: write its IP address to the start address and 0 to the distance field

tail: write its IP address to the end address if the distance is 0, the distance is incremented regardless of whether the distance is 0 or not

The expected number of packets needed to reconstruct the path is bounded by

$$\mathbb{E}[X] < \frac{\log(d)}{p(1-p)^{d-1}}$$

where p is marking probability or the probability of tossing a head and d is the length of the path.

1.22 Edge Sampling Quiz

Multiple attackers can be identified since edge identifies splits in reverse path.

1.23 Reflector Attack

A reflector attack is typically launched by a bot master spoofing the victim's IP address, commanding many bots to send a lot of requests to many reflectors such as DNS servers, web servers, and Gnutella servers. Since the actual flooding traffic is from the reflectors, the traceback mechanism is ineffective because the reflectors may not do any marking of paths.

1.24 Reflector Attack Quiz

DNS traffic filtering should take place as far from the victim as possible. Server redundancy and traffic limiting are always helpful as self-defense against reflector attack.

1.25 Capability Based Defense

The idea is that a client can specify what packets they want and this is called **capability**. When a client sends a request it must request capability in its SYN packet and such a request should be rate limited. The server can respond with capability included in its packet that the client can use later. All routers will only forward packets with valid capability, dropping or blocking packets from sources of attacks.

1.26 DoS Reality

DoS attacks must be considered at Internet design time. Therefore Internet is ill-equipped to handle DoS attacks.

2 Lesson 2: Cybercrimes

2.1 Actors in the Underground

exploit developer: very smart people who reverse-engineer software and develop and sell exploits packs and kits

botnet master: buy exploits and turn them into malware to control vast number of zombie machines, rent out botnet to other actors

spammer: rent a botnet from botnet master to function as spammer, sending out spam content on behalf of other bad actors

phisher: set up scam sites to steal information, ask spammers to send out phishing URLs to victim users of the scam site

counterfeiter: use spams to sell counterfeit goods and collect money from victim users

"bulletproof" hosting provider: offer dedicated servers to bad actors to evade law enforcement, typically operating in lawless places and are expensive

crowdturfer: leverage human-powered crowd sourcing platform to create and manage fake accounts not tied to real users and to solve CAPTCHAs

2.2 Structure of the Underground

A botnet is used to launch a number malicious and fraudulent activities including credit card and bank account theft, DDoS and ransomware extortion, click fraud and Ad injection, spam, and bitcoin mining. Spam can further be used to facilitate other activities such as phishing, selling counterfeit goods, and malware installation. In summary bad actors form an inter-connected ecosystem because their activities or infrastructures support each other.

2.3 Underground Forums

Underground forums are one of the entry points of bad actors, especially for those new to the underground. Many operate in plain site and are just one-click away from Google search. Nevertheless black market forums are hugely valuable to security professionals as they can learn trends and detect unfolding attacks. The forums are populated by both trustworthy buyers and fraudulent rippers. Most messages on the forums are just advertisements. The actual deal making is typically done via private messages.

2.4 Exploits-as-a-Service: Decoupling and Specialization

In the past compromising computer systems and using them for profit are typically done by the same criminal. Nowadays they are specialized into different functions, e.g. exploit developers sell exploits kits, other bad actors buy the kits to attack hosts, and compromised hosts are then sold on the black market. These bad actors are paid using the pay-per-install model.
As an example of Exploits-as-a-Service, a malware distribution model relies on drive-by-download attacks against browsers. There are two components in this malware distribution model

exploit kit: the bad actor can either buy an exploit kit and deploy it themselves, or they can simply rent access to an exploit server that hosts an exploit kit

have computer visit exploit server: the most common way is to use spam or phishing to attract traffic to the exploit server

Traffic-pay-per-install or traffic-PPI services simplify this process which essentially combine the two components into a single service. Pay-per-install is now the most popular way of distributing malware.

2.5 Dark Web Quiz

deep web: not indexed by standard search engines

dark web: hidden web with content typically only exists on so-called darknets, which is an overlay network that can only be assessed with specific software, configuration, or authorization, often using non-standard communication protocols and ports (e.g. P2P and Tor)

surface web: readily available to the public and searchable with standard search engines, surface web is a very small portion ($\sim 4\%$) of the Internet

2.6 Traffic-PPI Example

There are three classes of actors in traffic-PPI: (1) exploit developer (2) client (3) victim. Payment flows from the client to the exploit developer while malware flows from the client to the victim.

2.7 PPI Quiz

doorway page: a webpage that lists many keywords hoping to increase search engine ranking, scripts on that page redirect to the attackers page

crypter: a program that hides malicious code from anti-virus software

blackhat search engine optimizer: increase traffic to the attacker's site by manipulating search engines

trojan download manager: software that allows an attacker to update or install malware on a victim's computer

2.8 From Malware to Botnets

Infected computers are valuable resources in that they have a unique IP address and bandwidth and spare CPU cycles. Thus attackers want to utilize these infected machines, and the way to do so is to turn them into a botnet.

2.9 Command and Control: IRC Channels

The key to a botnet success is efficient and robust command and control. The simplest and most efficient way to perform command and control is through centralized control (e.g. IRC command). However this kind of control has a single point of failure thus is not robust. A more robust command and control structure is to use P2P network, in which the bot master can insert commands into the DHT and make advertisement. The drawback is that the bot master does not have direct synchronized communication with all the bots. Nowadays the most popular approach for command control is for all the bots to connect to a command-and-control website, which is not always fixed on one physical server. Instead of using fixed domains that can be detected and blocked, bot masters now use random domain generation, generating many domains each day and registering only a subset of them. This enhances the robustness of the command-and-control website.

2.10 Spam Quiz

The two defining characteristics of internet spam are (1) inappropriate or irrelevant and (2) large number of recipients.

2.11 Spam

Spam can be used to push malware or phish to steal information. Spammers need a large number of IP addresses because sending a large number of emails from a few IP addresses will easily trigger detection and blocking by spam filters. For spammers access to credit card processing infrastructure is crucial. Many scams even have legitimate customer service departments. Spammers advertise the scams and collect commission on successful sales, typically 30–50% of the sale price. The key to the success of both scammers and spammers is the spam conversion rate—the percentage of spam messages that result in sale.

2.12 Spam Revenue Quiz

The top three countries where spam directed visitors added items to their shopping cart are US, Canada, and Philippines.

2.13 Scamming Ain't Easy

Scamming requires an ecosystem that include network infrastructure and payment system. At the outset one needs to register a domain name. Some registrars are known to ignore complaints but they charge more. After obtaining a domain one needs a "bulletproof" DNS to evade law enforcement which usually operate in lawless territories and are expensive. Same for setting up a web server. After that one needs to worry about collecting payments as most banks won't do business with scammers. Finally scam sites almost always ship products to customers, because customer complaints will prompt processors and banks to shut down accounts.

2.14 Pharmaleaks

Contrary to conventional wisdom repeat orders are an important part of the business, and the net profit isn't high.

3 Lesson 3: Penetration Testing

3.1 Introduction to Penetration Testing

Penetration testing is the first line of network defense. Social engineering is a fast, low-risk method to gain access to data.

3.2 Overview

Penetration testing is used to evaluate the strengths of all security controls including procedures, operations, and technologies. The benefits of penetration testing include discovery of vulnerabilities and demonstration of threats by exploiting these vulnerabilities. The scope of penetration testing also includes social engineering and physical access. The scale of testing includes the entire network.

3.3 Methodology

Penetration testing includes several steps

footprinting: find the general information of the network

scanning: find more detailed information about the network e.g. services available

enumeration: find more targeted information such as user account

gaining access: find vulnerabilities associated with network services and exploit them to gain access to the network

escalating privilege: gain root or super user access

pilfering: steal information from the network, this is one of the standard activities that an attacker would do to a network

covering tracks: hide evidence of a break-in so that security admin cannot easily find out

creating back doors: create easy access for future malicious activities

The last few steps, from gaining access to creating back doors, can be iterated to move from one port of the network to another.

3.4 Footprinting

The attacker or tester conducts reconnaissance and information gathering. Important network information includes IP addresses or phone number range (for modem access or social engineering), namespace, and network topology.
The techniques and tools used commonly used for footprinting are

open source search: Google search engine

find domain name, admin, IP addresses, name servers: whois

DNS zone transfer: nslookup

3.5 Scanning

Scanning can be used to find out which machines are up and what ports are open, which services are running, their versions and configurations so that the attacker or tester can look up the corresponding vulnerabilities on the web. The most promising avenues of entry are typically associated with services that are always up such as web services. On the other hand the attacker or tester wants to reduce the frequency of scanning and randomize the ports or IP addresses to be scanned in the sequence to avoid detection.
The techniques and tools used commonly used for scanning are

ping sweep: fping, nmap

TCP/UDP port scan: nmap, fscan

OS detection: nmap

3.6 Enumeration

Enumeration performs more targeted attack or testing by identifying poorly protected user accounts, thus enumeration is more intrusive than scanning.
The techniques and tools used commonly used for enumeration are

list user accounts: dumpACL, onsiteAdmin

list file shares: showmount, NAT legion

identify applications: netcat, rpcinfo

3.7 Gaining Access

Once a vulnerability of the target is identified from scanning, an attacker or tester can exploit it using existing tools or scripts associated with it or customize them via modification. If the vulnerability is new then the attacker or tester has to develop the exploit himself/herself, because in general automatically generating a working exploit from a new vulnerability is still an open problem.
The techniques and tools used commonly used for gaining access are

password eavesdropping: tcpdump, ssldump, L0phtCrack, readsmb

file share brute forcing: NAT legion

password file grab: tftp, pwddump2

buffer overflow: ttdb, bind IIS

3.8 Escalating Privilege

The goal is to gain super user access so as to gain complete control of the system.
The techniques and tools used commonly used for escalating privilege are

password cracking: John the ripper, L0phtCrack

known exploits of privileged services: Lc_messages, getadmin, sechole

3.9 Pilfering

After gaining access to the network the attacker or tester can steal valuable information which allows further access to the system.
The techniques and tools used commonly used for pilfering are

discover trust relationship among machines: rhosts, LSA secrets

search for cleartext passwords: user data, configuration files, registry

3.10 Covering Track

Once total ownership of the target is secured, it is important to cover up the track so that the attack cannot be easily detected and stopped.
The techniques and tools used commonly used for covering track are

clear logs: zap, event long GUI

hide tools: rootkits file streaming

3.11 Creating Back Doors

An attacker wants subsequent access to be easy and look normal. Backdoors or trap doors will be laid to ensure that subsequent privilege access is easily regained.
The techniques and tools used commonly used for covering track are

create rogue user accounts: members of wheel, admin

plant remote control services: netcat, remote desktop, VNC

schedule batch jobs: cron

install monitoring mechanisms: keystroke loggers, add account to secadmin mail aliases

infect startup files: rc, startup folder, registry keys

replace apps with trojans: login, fpnwcint.dll

3.12 Penetration Testing Quiz

All of the following events should trigger a penetration test

- infrastructure is added or modified
- applications are added or modified
- end user policies are changed
- security patches are installed

Penetration testing should also be done on a regular basis as well as on these triggering events.

3.13 Persistence and Stealth

To be persistent and stealthy the tester can install a back door through a malware so that there is a permanent foothold in the network. The malware can be placed in a strategic place such as a proxy so that the malware can listen and record all traffic within the network. By analyzing internal traffic the malware can capture user credentials and find out valuable information.

3.14 Social Engineering

Users are the weakest link in security. Social engineering is used to identify vulnerable user groups and policy gaps. Social engineering is effective when users are manipulated into undermining the security of their own systems. This can be accomplished by abusing trust relationships among users. Thus admins should fix policies and develop mechanisms including user education and training. Social engineering can be very easy and cheap for the attacker because the attacker does not need any specialized tools or technical skills.

3.15 RSA Breach Quiz

In 2011 RSA was compromised. Social engineering was used to penetrate the company's defense. Once in the attacker installed a back door using an Adobe Flash vulnerability. This was carried out in three steps

1. identify employees that are vulnerable
2. craft an email subject line (2011 recruitment plan) that entices an employee to open it
3. hide an executable file (Excel spreadsheet that contains a zero-day exploit) in the email that will install onto the victim's computer when the email is opened

3.16 Common Social Engineering Techniques

impersonation: help desk, third-party authorization, tech support, roaming the halls or tailgating, trusted authority/repairman, snail mail

computer-based techniques: pop-up window, instant message or IRC, email attachment, email scam, chain letter, phishing website

3.17 Impersonation

help desk: the attacker pretends to be an employee trying to recover a forgotten password, the exploit is that help desks often do not require adequate authorization

third-party authorization: the attacker claims that the third party has authorized him an access to the network, the exploit is the claim that a third party has authorized the target to divulge sensitive information, which is more effective if the third party is out of town

tech support: the attacker pretends to be tech support asking user for credentials, the exploit is that the user is not properly trained to guard their credentials

roaming the hall: the attacker dresses to blend in with the environment, the exploit is the sensitive information that has been left unattended

repairman: the attacker wears appropriate uniform thus is allowed into sensitive environment, the attacker can then plant surveillance equipment to capture useful information, the exploit is that people rarely question someone in a uniform

trusted authority figure: similar to the third-party authorization attack, the attacker pretends to be someone in charge of a company or department to gain useful information from a user, which can then be used to impersonate employee through the call to a help desk etc. the exploit is the trust in perceived authority

snail mail: the attacker sends mail to a user pretending to be an authority and asks for personal information, the exploit is that people tend to be more trusting of printed words than webpages

3.18 Computer Attacks

popup window: imitates secure network login, prompts users for login credentials, the defense is that users can check for visual indicators to verify security

instant message or IRC: imitates tech support, redirects users to malicious sites to download trojan and install surveillance program

email attachment: tricks users into downloading malicious software which are hidden in attachment e.g. executable macros embedded in PDF, camouflaged extension like .doc.exe

email scam: begins by requesting basic information

chain email: more of a nuisance than a threat, spread using social engineering techniques

phishing website: offers prize but requires a created login, capitalizes on users reusing login credentials

3.19 Computer Attacks Quiz

The top three industries that were targets of cyber attacks in 2016 are: defense contractor, restaurant, and software.

3.20 Countering Social Engineering Attacks

- never disclose passwords
- limit IT information disclosed (only the IT staff should discuss details of network system configuration, the IT staff should not answer any survey call)
- keep information in auto-reply emails to be bare minimum
- escort guests in sensitive areas
- question people you don't know
- educate employees about security, both physical and IT security
- centralize reporting and management of all suspicious behavior

4 Lesson 4: Browser Security Models

4.1 Common Application Attacks Quiz

using components with known vulnerabilities: uses unpatched third party components

missing function level access control: privilege functionality is hidden rather than enforced through access controls

sensitive data exposure: abuses lack of data encryption

security misconfiguration: exploits misconfigured servers

insecure direct object references: attackers modify file names because file names are direct object references

cross-site scripting: inserts JavaScript into trusted sites

broken authentication and session: program flaws allow bypass of authentication methods

injection: modifies back-end statement through user input, i.e. the attack action is injected into user input

4.2 Goals of Web Security

browse the web safely: sensitive data on the user's computer cannot be stolen and uploaded to the web, if the web browser has multiple open sessions with multiple sites the compromised session should not compromise others

support secure web applications: the web applications can have the same security protection as the other applications that run on our computers

4.3 Web Security Threat Model

We are going to compare the web security threat model and the network security threat model

web security threat model: an attacker sets up a malicious site and waits for users to visit the malicious site, a web attacker typically does not control the network

network security threat model: an attacker can intercept and control the network, e.g. intercept and drop the traffic or crack open the encryption key in the traffic, inject malicious traffic into the network

4.4 Attack Top 10 Quiz

According to the OWASP in 2013, the following are the top 10 attacks on web security

1. injection
2. broken authentication and session
3. cross site scripting
4. insecure direct object references

5. security misconfiguration
6. sensitive data exposure
7. missing function level access control
8. cross site request forgery
9. using components with known vulnerabilities
10. unvalidated redirects and forwards

4.5 Web Threat Models

A web attacker can control a malicious website. He can even obtain a SSL/TLS certificate for it. The attacker can wait for users to visit the malicious website, or set up some malicious or fake web app and wait for users to download and run it. Thus a web attacker is somewhat passive.
A network attacker is more powerful. He can perform both passive and active attacks

passive: wireless eavesdropping to crack the encryption key for the WIFI

active: evil router, DNS poisoning—the attacker changes the DNS entry so that a legitimate website now has an IP address of a server that is controlled by the attacker

The most general and powerful attack is through malware. By injecting malware into users' computer the attacker essentially escapes the browser's isolation mechanism and runs separately under the control of OS like any other applications. This is possible because browsers may contain exploitable bugs that enable remote code execution by websites. Even if the browser is bug-free it may still contain lots of vulnerabilities, in particular on the server side that enable cross-site scripting (XSS), SQL injection (bypassing web server to inject malicious code into the back-end SQL database), and cross-site request forgery (CSRF). The crux is that malware attackers can bypass the basic control of the web to attack users' computer or the web server.
So there are three main types of attackers: (1) malware attacker (2) network attacker (3) web attacker. Among them web attacker is the least lethal because he is the most passive, while malware attacker is the most lethal because he can inject code into users' computer to perform any action as desired.

4.6 Modern Websites

A typical website contains both static and active contents. On a typical webpage we have page code, third-party API, third-party libraries, and Ad code. Acting parties on a website include page developers, library developers, service providers, data providers, Ad providers, extension developers, and content distribution network or CDN. So the basic security questions are, with data and codes from so many different sources how do we protect sensitive information and how do we ensure data integrity?

4.7 Browsers

Similar to operating system a web browser can render multiple webpages to different sites and each page can contain data and code from multiple sources. Thus it is constructive to compare the two

	operating system	web browser
primitives	system calls, processes, disk storage	document object model, frames, cookies/local storage
principles	users: discretionary access control	origins: mandatory access control
vulnerabilities	buffer overflow, root exploit	cross-site scripting/request forgery, cache history attack

Under the basic execution model, given a webpage the browser goes through the following steps

1. loads the content
2. renders the content by processing HTML and scripts to display the page
3. responds to events—(1) user action e.g. on click, on mouse (2) rendering e.g. on load, on unload (3) timing e.g. set timout, clear timeout

The content can be from scripts, frames loading HTML pages, stylesheets (CSS), and objects like flash. By specifying "allowscriptaccess" flash objects can communicate with external scripts, navigate external frames, and open windows.
The basic idea of browser security is to sandbox web contents, more specifically safely execute Javascript code provided by a remote website. This means that a Javascript code cannot access the file system directly. It can only have limited access to the operating system, the network, and browser data, as well as content from other websites. The main policy is the so-called **Same-Origin Policy (SOP)** under which active code such as Javascript can only read properties of documents and windows from the same origin defined as the same protocol, domain, and port. Exception to SOP is allowed to scripts such as those signed by Microsoft, Google, Apple etc. to perform universal browser read/write, universal file read, and universal send mail.

4.8 Sandbox Quiz

The following is the characteristic shared by both sandbox and virtual machine

- anything changed or created is not visible beyond its boundaries

The following is the characteristic of sandbox

- if data is not saved e.g. malware, it is lost when the application closes
- it is lightweight and easy to set up

The following is the characteristic of virtual machine

- it is a machine within a machine
- disk space must be allocated to the application

4.9 Browser Same-Origin Policy

An **origin** is defined by protocol, domain, and port

protocol://domain:port/path?params

The Same-Origin Policy (SOP) for document object model or DOM states that Origin A can access Origin B's DOM if A and B have the same (protocol, domain, port). Different from DOM for cookies two cookies have the same origin if they have the same domain and path. The protocol is optional.

4.10 Frame Security

Frame and iframe are like many browser windows. A **frame** is typically rigid or fixed on a page whereas an **iframe** can be floating. Windows may contain frames from different sources. Frames provide a natural isolation or separation of different web contents. Browser provides isolation based on frames, and we should be able to achieve the same kind of isolation whether we use two different browser windows or use a frame within a window. Parent may work even if a frame is broken.
We apply the same SOP to achieve frame security—each frame of a page has an origin defined as protocol://host:port, a frame can access only the data from its own origin (network access, read/write DOM, storage cookies) so that different frames do not interfere with each other. In addition frame-to-frame access control policy can also be specified. For example

canScript(A,B): Frame A can execute a script that manipulates DOM elements of Frame B

canNavigate(A,B): Frame A can change the origin of content for Frame B

Likewise we can specify policy for frame to access principle. For example we can use readCookie(A,S) or writeCookie(A,S) to specify that Frame A can read or write cookies from Site S.

4.11 Browsing Context

A **browsing context** may be

a frame with its DOM: a frame with web contents

web worker: does not have a DOM, a web worker is a Javascript executed from an HTML page that runs in the background independently of other user interface elements that may also have been executed from the same HTML page

Every browsing context has an origin determined by protocol, host, and port, and as such is isolated from another context by the SOP. Different browsing contexts may communicate using postMessage, and they can make network requests through XHR or tags where XHR stands for XML HTTP Request. It is an API available to Javascript. Typically XHR is used to send HTTP or HTTPS requests to a web server. That is a Javascript uses XHR to request contents from a web server.
There are similarities between browsing context and process context. An opening system uses separation and isolation to allow multiple execution contexts and provide local storage and communication services. Similarly while a web browser provides common local storage, it uses isolation and separation to provide security protection to the browsing contexts.
The modern browser mechanisms that can be used for security protection include

HTML5 iframe sandbox: load with unique origin, limited privileges

content security policy (CSP): whitelist instructing browser to only execute or render resources from specific sources

cross-origin resource sharing (CORS): relax same-origin restrictions

HTML5 web worker: separate thread, isolate but same origin, not originally intended for security but helps

subresource integrity (SRI):

The idea of HTML sandbox is to restrict frame actions. We can use

sandbox directive: ensure that the iframe has unique origin, cannot submit forms, and APIs are disabled, and it can prevent contents from plugins

sandbox allow-scripts directive: only ensure that the iframe has unique origin but allow the rest of the actions

Other sandbox directives include allow-forms, allow-popups, allow-pointer-lock, allow-same-origin, and allow-top-navigation.

4.12 Content Security Policy

The goal of **content security policy (CSP)** is to prevent or at least limit the damage of cross-site scripting (XSS). A XSS attack bypasses the same-origin policy by tricking a site into delivering some malicious code along with the intended content. CSP prohibits inline scripts embedded in script tags, inline event handlers, JavaScript, and URLs etc., and also disables JavaScript eval, new function and so on. That means all the resources that a browser will load can be statically checked. Moreover the resources are loaded only from a whitelist. With CSP we can specify the whitelist for each type of web contents (directives include: script-src, connect-src, font-src, frame-src, img-src, media-src, object-src, style-src, and default-src).
The sources of web contents can be specified by

scheme: e.g. HTTP, HTTPS

host name: any origin on that host

fully qualified URL: e.g. https://example.com:443

You can also specify how to match the sources listed on a whitelist, such as wildcards accepted, none, or self (current origin but not subdomains) and so on. You can even create exceptions or allow inline JavaScripts (unsafe-inline) or allow eval functions (unsafe-eval).

4.13 CSP Quiz

CSP can be used for third-party software even if it has inline script. CSP allows third-party widgets to be embedded on your site as long as it is on the whitelist.

4.14 Web Worker

Web workers were ultimately not intended for security, but they help improve security because they allow JavaScript to run in isolated threads. A web worker has the same origin as the frame that creates it, but it has no DOM. The main iframe thread creates a worker and then starts the worker thread by sending a message using postMessage.

4.15 Subresource Integrity

To use subresource integrity, the website author who wishes to include a resource from a third party can specify a cryptographic hash of that resource in addition to the location of the resource. Then when a browser fetches the resource, it can compare the hash provided by the website author with the hash computed from the resource. If the hashes do not match then the resource is discarded.
When the integrity check fails, by default the browser can report the violation and does not render or execute the resource. If the directive simply says report, then the browser will report the violation but can still render or execute the resource.

4.16 Cross-Origin Resource Sharing

Cross-origin resource sharing (CORS) is a technique that we can use to relax the same-origin policy, so that JavaScript on a web page can consume content from a different origin. It basically uses a whitelist. The browser sends the origin header with XHR. The value of this header is the domain that served the parent page. The server can inspect the header and respond whether the access is allowed or not.

4.17 CORS Quiz

CORS requires coordination between the server and the client. However it is not widely supported by browsers. Moreover the CORS header cannot be used as a substitute for sound security.

4.18 SOP Quiz

An **origin** is a combination of Uniform Resource Identifier (URI) scheme (such as HTTP or HTTPS), hostname, and port number.

4.19 SOP and Cookies

There are a number of attributes that a server can set for a cookie

SameSite: do not send cookie on a cross-site post request, provide some sort of CSRF defense

strict: never send cookie on cross-site request, provide some sort of CSRF defense

HttpOnly: can only be assessed by the server, any attempt to access the cookie from script is strictly forbidden, provide some sort of XSS defense

The scope of a cookie is determined by the combination of domain and path, with scheme/protocol being optional.

4.20 Scope Setting Rules

A **domain** is any domain-suffix of a URL hostname except the top level domain (TLD) such as .com. For example the web server login.site.com can set cookies for login.site.com and .site.com but not for other.site.com or .com. Path can be set to anything within that domain.
Cookies are identified by (name, domain, and path). The browser sends all cookies in URL scope which is determined by domain and path. The goal is that a server should only see cookies in its own scope. For the following two cookies both set by login.site.com

- cookie1: name = userid, value = u1, domain = login.site.com, path = /, secure
- cookie2: name = userid, value = u2, domain = .site.com, path = /, non-secure

different servers see different cookies depending on their scopes

- http://checkout.site.com only sees cookie2 (.site.com)
- http://login.site.com only sees cookie2 (non-secure)
- https://login.site.com sees both cookie1 and cookie2

A JavaScript can set cookie values. It can also read out the attributes of a cookie. It can even delete a cookie. The exception is that if the cookie is set as HttpOnly then it cannot be accessed by client-side scripts, i.e. client-side scripts cannot read or write this HttpOnly cookie.

4.21 SOP Security Quiz

The URL has the same origin if it has the same protocol (HTTP vs. HTTPS), hostname, and port. Note that unspecified port is different from specified port, and www.example.com is a different hostname from example.com.

4.22 Cookie Quiz

super cookie: a cookie with an origin of a top-level domain

zombie cookie: a cookie that is regenerated after it is deleted

same-site cookie: a cookie that can only be sent in requests originating from the same origin as the target domain, can be used to defend against CSRF

HTTP only cookie: a cookie that cannot be accessed by client-side APIs, can be used to defend against XSS

third-party cookie: a cookie that belongs to a domain that is different from the one shown in the address bar

session cookie: an in-memory cookie, it does not have an expiration date, it is deleted when the browser is closed

persistent cookie: a cookie that has an expiration date/time, also called tracking cookie

secure cookie: a cookie that can only be transmitted over an encrypted connection

4.23 Cookie Protocol Problems

A server does not see all the cookie attributes, nor does it see which domain sent the cookie. Actually all the server sees is some selected attributes sent by the browser. This can be exploited by attackers. A network attacker can intercept and rewrite HTTPS cookies, which means that even with a HTTPS cookie its values cannot be trusted.
The path separation for cookies is done only for efficiency not for security. Recall that the scope of a cookie is determined by domain and path, which means that x.com/A does not see cookies of x.com/B because they are different paths. However this is not a strong security measure, because x.com/A still has access to the DOM of x.com/B since they are the same origin as far as DOM is concerned. Hence x.com/A can use alert(frames[0].document.cookie) to print out or read the cookie of x.com/B.
Another security problem of cookies is that cookies have no integrity. A user can change or even delete cookie values.

4.24 Cryptographic Checksums

We can use cryptography to provide data integrity protection. The main idea is that when a server sets a cookie attribute, it will attach an integrity check value for that attribute, and it can later on check whether that attribute has been modified. To do this the server uses a secret key k that is unknown to the browser, and for each attribute value that is set it computes an integrity check, calling it a tag T which essentially is a message authentication code, i.e. $T = \text{MACsign}(k, \text{SID} \parallel \text{name} \parallel \text{value})$. When it sets the cookie it attaches T to each attribute value. When a browser later on presents that cookie to a server, the server can use the secret key k to compute the check and verify that the result is the same as T, i.e. $\text{MACverify}(k, \text{SID} \parallel \text{name} \parallel \text{value}, T)$.

4.25 Checksum Quiz

Cryptographic hash functions that are not one-way are vulnerable to preimage attacks. A good cryptographic hash function should employ an avalanching effect. A difficult hash function should be very hard for attackers to analyze, but we want all the hash function to be efficient to calculate.

5 Lesson 5: Web Session Management

5.1 Session

A **session** is a sequence of requests and responses from a browser to a server. The goal of session management is to authenticate user only once so that all subsequent requests are tied to the authenticated user. The general idea behind session management is to use session tokens. At the initial handshake between the browser and the web server, the server sets an anonymous session token. Once the user has been authenticated, the server can elevate the token from anonymous browsing token to an authenticated token. When the user logs out or checks out, this login session token should be cleared.
There are many ways to store the session tokens

browser cookie: the problem is that a browser can send a cookie with every request even when it does not, this gives rise to CSRF attack

embed in all URL links: the problem is that if the application is not returned securely or if the user posts URL in a public forum, there can be token leaks via http referrer header

in a hidden form field: the problem is that every user action must result in a submission of a form or you lose the session token

Thus none of these methods are perfect. The best solution is to choose a combination of these three options depending on the application.
When a browser sends a URL request to a server, if the request contains a HTTP referrer header, it tells the server the page that you are coming from, meaning your referrer. By checking the referrer the web server can see where the request originated. In the most common situation, this is the last page the user was on, the one that contains the hyperlink the user clicked. The problem with referrer is that it can leak the session token to the previous server. The solution is that he can suppress the referrer, which means that don't send referrer when you refer to a site.

5.2 Session Logout

After the user logs out he should be allowed to log in with a different account, and a website should prevent a user from accessing content left behind by a previous user. Thus a logout procedure should

1. the session token on a browser should be deleted

2. the session token should be marked as expired

The problem is that many websites do 1 but not 2. This is especially dangerous for websites that use HTTPS for login but then fall back to the clear text HTTP after login. This is because an active network attacker can intercept the cleartext HTTP traffic and steal a copy of the session token. Then even after the user logs out, because the server does not expire the session token, the attacker can continue to use that session token.

5.3 Session Token Quiz

Tokens expire. But there should still be some mechanism to revoke them if necessary. Tokens must be stored somewhere, and unlike cookies which are always small, tokens can be quite large depending on how much information is stored.

5.4 Session Hijacking

A major threat in web session management is **session hijacking**, where the attacker waits for user to log in and then steals the user's session token and hijacks the session. Session hijacking is not limited to active network attacker that intercept traffic. For example

- if a counter is used as a session token, then when a user logs into a website he will get a counter value, then he can view sessions of other users because he would know other counter values
- even if the token is protected using cryptography, if the cryptographic algorithm or the key is weak, then a user can still break the protection, get the counter value, and then view sessions of other users

Hence we should use tokens that are not predictable, and there are underlying frameworks that allow us to generate random session IDs e.g. ASP, Tomcat, Rails. However even when a session token is random, there is still a security threat of session token theft. For example

- if a web site uses HTTPS for log in but subsequently use HTTP for the rest of the session, then an active network attacker can use a tool such as Firesheep to intercept the cleartext HTTP traffic and steal the session token, another way for the attacker to steal the session token is to play man-in-the-middle at the beginning of the SSL connection
- use cross-site scripting attacks, if the server does not invalidate a session token after the user has logged out, then the stolen token can still be used by the attacker after the user has logged out

One idea to mitigate session hijacking is to bind a session token to the user's computer, embedding some machine specific data in the session ID. We want the machine specific data to be unguessable and unique to the machine but still quick to generate. IP address is not a good option because the user's computer can change its IP address due to DHCP (Dynamic Host Configuration Protocol), then the user will be locked out of his own session. Browser user information is not a good option either, because such information is easily stolen or guessable by the attacker. Hence while it is appealing to use some kind of site information in a session token, there is not a good solution when we consider both security and convenience. Therefore the best approach is still an unpredictable session token generated by the sever.

5.5 Session Fixation Attacks

In addition to stealing tokens an attacker can also fix session tokens. It can trick the user into clicking a URL that sets a session token, or it can use cross-site scripting attacks to set token values. The steps an attacker can use session fixation attack to elevate his anonymous token to a user log-in token are

1. the attacker gets anonymous browsing session token from site.com
2. the attacker sends a URL to the user with the attacker's session token
3. the user clicks on the URL and logs into site.com
4. the attacker uses the elevated token to hijack user's session

To mitigate such attacks, when elevating a user from anonymous to logged in a website should always issue a new session token. Thus after the user logs in the token will change to a different value unknown to the attacker, so that the anonymous token that the attacker had originally obtained is not elevated.

5.6 Session Hijacking Quiz

active session hijacking: involves disconnecting the user from the server once that user is logged in, social engineering is required to perform this type of hijacking

passive session hijacking: the attacker silently captures the credentials of a user, passive hijacking is less likely to raise suspicions

5.7 Session Management Summary

We should always assume cookie data retrieved from a client is adversarial or not trusted. Cookies by themselves are not secure (e.g. CSRF, overwritten).

6 Lesson 6: HTTPS

6.1 HTTPS

Hyper Text Protocol Secure or HTTPS is essentially HTTP over SSL, the secure socket layer, which is now called transport layer security or TLS. With HTTPS all traffic between a web browser and a web site is encrypted, whereas HTTP is a cleartext protocol meaning that the traffic is not encrypted, thus using HTTP a network attacker with access to the link can intercept the traffic data and learn the user's password. HTTPS allows for secure communication over untrusted or public network. It encrypts traffic and uses public key to authenticate the web server and if possible even the browser. HTTPS can still provide security even when only one side of the communication is secure.
Nonetheless with all these benefits HTTPS is not yet used for all web traffic. The reason is that

- crypto operations can slow down the web service (but not much if done right)
- some ad networks do not support HTTPS, because with HTTPS ad publishers cannot learn the web contents as being viewed by the users

6.2 HTTPS Quiz

Request URL, query parameters, headers, and cookies can be encrypted by HTTPS. Host address and port numbers are used to route traffic so they are not encrypted. The amount of transferred data and length of session can be inferred by observing the traffic so the attacker can learn this anyway.

6.3 Network Attacker

A network attacker can control network infrastructures such as routers and DNS servers. It can eavesdrop to learn traffic contents, inject data, block traffic, or even modify the contents. HTTPS was designed to thwart such network attackers.

6.4 SSL/TLS Overview

SSL/TLS uses public key for authentication and key exchange. In public key cryptography Bob has a pair of public key and private key. After obtaining Bob's public key Alice can use Bob's public key to encrypt message into cyphertext, which can only be decrypted by Bob using the corresponding private key.

6.5 Certificates

An essential problem in public key cryptography is how Alice can obtain the public key of Bob. The standard is to use certificate issued by a certificate authority or CA. There are a large numbers of CAs. A browser typically accepts certificates from 60 top level CAs and 1200 intermediate CAs. The steps are as follows

1. every entity has installed the public key of CA
2. Bob asks the CA to generate a certificate for his public key, the CA keeps the signing private key to itself
3. the CA signs Bob's public key using its signing private key and the signature is put into the certificate
4. Bob presents the certificate to Alice
5. because Alice has the CA's public key, she can verify that the signature was constructed properly, which means that Bob's public key has been certified by the CA

A public key certificate contains important information such as a unique serial number, a valid time period, a public key, and a signature produced by the CA.
A certificate is for an entity or subject that is identified by a common name, which can be an explicit name e.g. cc.gatech.edu, or a wildcard e.g. $*$.gatech.edu. If a wildcard is used it can only be the leftmost component and it does not match a dot, e.g. $*$.a.com matches x.a.com but not y.x.a.com.

6.6 SSL and TLS

The handshake is carried out in the following steps

1. the browser sends a hello message to a server and the server's response includes a public key certificate
2. the browser verifies the certificate, meaning that now the browser knows the server's valid public key
3. the browser performs key exchange using for example Elliptic curve Diffie-Hellman key exchange or EC-DHE
4. with the server's public key, the browser and the server can perform secure key exchange and prevent man-in-the-middle attack

The goal of this handshake is to authenticate the server and optionally the browser. More importantly at the end both will have a shared secret key that can be used to encrypt HTTP traffic.

6.7 HTTPS in the Browser

HTTPS is integrated into a browser, or it is indicated in the browser GUI with a lock icon. The goal is to let the user know where a page came from and that the page contents are protected. When the lock icon is displayed on the browser, it means that all the elements on the page are fetched using HTTPS. But for the browser to even accept this HTTPS connection, the browser should have trusted the certificate and verified that the certificate is valid. In addition the domain URL should match the Common Name or Subject Alternative Name in the certificate.

6.8 HTTPS Disadvantage Quiz

The disadvantages are

- need to buy an SSL certificate
- mixed modes issues—loading insecure content on a secure site, this will continue to be the problem until many sites are all using HTTPS
- public caching cannot occur because all traffic is encrypted so there is no public caching possible.

6.9 Problems with HTTPS and the Lock Icon

The two main problems with HTTPS and the lock icon are (1) SSL stripping (2) forged certificate.
SSL stripping, also known as HTTP downgrading attack, prevents the browser from upgrading from HTTP to HTTP. With SSL stripping the browser will not display any SSL certificate errors, and the user has no clue that such an attack is happening. For example when a user wants to transfer money to his account using an online banking service, and in the background the user's computer happens to be connected to the attacker's machine, the attacker forwards the request to the bank and waits for the response. The connection between the attacker and the bank is secure, i.e. the traffic is transferred using an SSL tunnel. The attacker has access to the login page and can modify the response from the server from HTTPS to HTTP and then forward the login page in HTTP to the user. From this point on, all of the user's requests go out in plain text, and the attacker can access the data and collect the credentials.
The solution to SSL stripping attack is to use HSTS, which stands for Strict Transport Security. This policy can be set for a maximum of one year. It basically tells the web browser to always use HTTPS even for its subdomains. When a web browser visits a website for the first time, the website can tell the browser to always use HTTPS. A web browser can also have a preloaded list of HSTS websites. The HSTS flag set by a website can be cleared when the user selects clear private data.
For **forged certificate** if a CA is hacked the attacker can issue rogue certificates e.g. for Gmail. Once a rogue certificate is issued the attacker can set up a fake website calling itself Gmail. With a rogue certificate an attacker can play man-in-the-middle even in HTTPS connection. The attacker plays the bank server to the user and the user to the bank server, and both sides of connections are in HTTPS. Several countries have been caught issuing unauthorized certificates. The ISPs in these countries can play man-in-the-middle between a user and the real server.

6.10 Solution to Certificate Issues

One approach to deal with rogue certificate is to use dynamic public-key pinning. This means that a website will declare the CAs that sign its certificate. When a browser first visits a website, the website tells the browser the list of authorized CAs. On subsequent visits the browser will reject any certificate issued by other CAs.

Very similarly there is a public-key pinning extension for HTTP or HPKP. This feature tells a web browser the list of public keys to be associated with the website. When the browser visits a website for the first time, the browser sends a list of public-key hashes. On subsequent visits the browser expects the server to use one or more of these public keys in its certificates.
Another way to deal with forged certificate is for the CAs to be transparent. That is the CAs must publish in a public log of all the certificates that they have issued. A browser will accept a certificate only if it is published on the public log. Companies like Google can constantly scan the public logs to look for invalid or forged certificates.

7 Lesson 7: Security of Internet Protocols

7.1 Internet Infrastructure

The computers within a local area network use the local and inter-domain routing protocol to communicate with each other. Computers from different ISP networks use TCP/IP protocol to communicate. But in order to decide how to send traffic from host A to host B, which may be in two separate ISP networks, there needs to be routing information which is decided by Border Gateway Protocol or BGP. The domain name system or DNS is a distributed hierarchical database system that provides the mapping between an IP address and a symbol domain name.

7.2 Infrastructure Quiz

tier one network: one that can reach every other network through peering, there are only 17 tier one networks in the world

tier two network: one that peers some of its network access and purchase some of its network access

tier three network: one that purchases all of the transit from other networks

7.3 TCP Protocol Stack: Data Formats

link layer protocol: a group of protocols that only operates on the link the host is physically connected to

network layer protocol: a group of protocols that are used to transport packets from one host to another and may cross network boundaries if necessary, e.g. IP protocol

transport layer protocol: provide host-to-host communication services for applications such as connection-oriented data stream support, reliability, flow control, and multicasting, e.g. TCP protocol

application layer protocol: depend upon the underlying transport layer protocols to establish host-to-host data transfer channels, and manage the data exchange in a client-server or peer-to-peer networking model

The data formats are

application layer: data starts as application message

transport layer: segment = TCP header + message

network layer: packet = IP header + segment

data link layer: frame = link header + packet

7.4 Internet Protocol: IP Routing

At the IP layer the IP protocol routing is connectionless because it is best effort only and unreliable. The ports are not part of the IP header because they are for the transport layer. If a data segment is too large it may be fragmented into multiple IP packets. At the receiving end these fragments will be assembled back together.
If the destination did not receive a packet or fragment, it can send an Internet Control Message Protocol or ICMP packet to the source to report the error. The IP header can also include a Time-to-Live (TTL) field. This field is decremented after every hop and a packet is dropped if its TTL reaches 0. TTL is useful to prevent infinite loops.

7.5 IP Authentication

In the IP header the source and destination IP addresses must be specified. However one can easily override the source IP address using raw sockets, for example by using the libnet library to format raw packets with arbitrary IP header information including the source IP address. This means that there is no guarantee that the source ID address is authentic, and one can send packets with arbitrary source IP addresses. The ability to forge arbitrary source IP addresses enables anonymous or hard-to-traceback attacks such as denial-of-service and malware infection.

7.6 Transmission Control Protocol

The transmission control protocol or TCP is connection-oriented and preserves the order of packets. At the source application data are broken into TCP packets, each of which has a sequence number attached. At the destination each packet upon its receipt will be acknowledged, and any lost packet will be notified so that the source can resend the packet, and then the packets will be reassembled in the original order. The TCP header includes port numbers, sequence number of the packet, and acknowledgment number for a previously received packet. It also has a number of flags used to control the TCP connection.

7.7 Review TCP Handshake

1. the client sends a SYN packet with randomly generated sequence number SN_c and acknowledgment number $AN_c = 0$ to the server

2. the server sends a SYN/ACK packet with randomly generated sequence number SN_s and acknowledgment number $AN_s = SN_c + 1$ to the client

3. the client sends an ACK packet with sequence number $SN_c = AN_s$ and acknowledgment number $SN_s + 1$ to the server

At this point the connection is established. Once a connection is established both sides can expect that their next packet will have the sequence number that is increment from the previous packets. Although packets can arrive out of order, one can expect that the sequence number should not be too far out of the current window. Therefore if a packet arrives with a sequence number that is too far out of the current window, it would be dropped.

7.8 TCP Basic Security Problems

There are three

- packet sniffing via eavesdropping, which is quite easy for an attacker if he can control your router or the WiFi access points
- TCP state can be easily obtained by eavesdropping, which enables spoofing and session hijacking
- denial-of-service attacks

7.9 TCP/IP Security Issues Quiz

Network layer controls can protect the data within packets as well as the IP information for each packet. Data link layer controls cannot protect connections comprised of multiple links.

7.10 Random Initial Sequence Numbers

In TCP handshake the first packet from the client and the first packet from the server have the sequence numbers randomly generated. This randomness is important because if these initial sequence numbers are predictable, then an attacker can forge a source IP address and finish TCP handshake to establish a TCP session. This will break IP-based authentication such as Sender Policy Framework or SPF that is used to authenticate email. From this point on the attacker can send command through the server. For example he can then send a reset packet to terminate a connection and result in a DoS attack.

7.11 Protocols Quiz

address resolution protocol (ARP): maps IP addresses to the hardware addresses used by a data link protocol, necessary because machines on the same local area network are identified by MAC addresses, hence the router sends an ARP request asking for the MAC address of the specified IP address, this request will reach all computers on a network

open shortest path first (OSPF): uses a link state routing algorithm for interior routing

border gateway protocol (BGP): used to exchange routing and reachability information among autonomous systems or AS, each AS talks to a peer to learn the address prefix of the computers within their peer network, this helps ASes work together to determine how to route traffic from one network to the other

7.12 Routing Security: Interdomain Routing

From the perspective of routing the Internet is a collection of domains or autonomous systems. An **autonomous system** is a connected group of computers where their IP addresses share some common prefixes and they are using a single or common routing policy. The routing between these domains is determined by BGP and the routing within each domain is determined by protocols such as OSPF.

ARP: if node A is malicious, it can send an ARP reply to the gateway with its own MAC address, if this reply arrives at the gateway before that of node B, then the gateway will think that node A is node B, which means that node A now can read or inject packets into node B's sessions

BGP: in BGP routing information in particular route updates are not authenticated, therefore through a false advertisement an attacker can cause the traffic to a victim host to instead be routed to the attacker's own address

7.13 BGP

The main security issues of BGP are due to the fact that BGP path information is not authenticated, which means that anyone can inject false advertisements and such advertisements will be propagated everywhere. As a result attackers can shape and route traffic to launch DoS attacks, send spams, and perform eavesdropping.

7.14 BGP Security Issues

Two solutions are

public key infrastructure (PKI): each AS obtains a certificate to certify each route origination authority (ROA) from the regional Internet register (RIR), and then attach the ROA to path advertisement, advertisements without a valid ROA are ignored, this provides defense against a malicious AS but not a network attacker

S-BGP: sign every hop of a path advertisement

7.15 S-BGP

S-BGP uses IPsec to protect point-to-point router communication. It also uses PKI to provide attestations, which consist of address attestation that proves the authorization to advertise certain address blocks and route attestations that prove the validation of the route update information. As such S-BGP requires repositories and tools to manage certificates, the certificate's revocation lists or CRLs, and the address attestations.

address attestation: the issuer is the organization that owns the address prefixes contained in the attestation, and the subject is one or more ASes that are authorized to advertise these prefixes, an AS such as an ISP has to be authorized by the owner of the address blocks to advertise the route to these address blocks
an address attestation includes (1) owner's certificate (2) AS to advertise the address blocks (3) address blocks (4) expiration date, the owner uses his private key to sign the address blocks, address attestation is used to protect BGP from incorrect updates from authenticated but misbehaving or misconfigured BGP speakers

route attestation: the issuer/speaker is an AS and the subject/listener is a transit AS, route attestation allows a BGP speaker that receives a route advertisement to verify that each AS along the route has been authorized by the preceding AS along the path to advertise that route, and that the originating AS has been authorized by the owner of each IP address prefix contained in the update to advertise these prefixes
a route attestation includes (1) the speaker's certificate issued by the owner of the AS (2) the address blocks and the list of ASes in the update (3) the neighbor (4) expiration date, the signature guarantees that the organization owning the IP address space advertised in the update was allocated that address space through a chain of delegation originating at the IANA, route attestation is used to protect BGP from incorrect updates from authenticated but misbehaving or misconfigured BGP speakers

To validate a route an AS needs to perform

- address attestation and address allocation certificate from each organization owning an address block in the network layer reachability information or NLRI

- route attestation and certificate for each AS along the path AS_1 to AS_n where the route attestation for AS_k specifies the NLRI and the path up to that point i.e. AS_1 through AS_{k-1}
- checking all relevant CRLs

8 Lesson 8: Domain Name Systems Security

8.1 DNS: Hierarchical Name Space

DNS or the domain name system is a hierarchical database. There are root servers, top level domains, second level domains, third level domains, and so on. There are 13 DNS root name servers.

8.2 DNS Lookup Example

A DNS Lookup is an iterative or recursive process, querying the hierarchy code database starting on the root or top level domain servers. That is the local DNS server asks the root and top level domain servers what the IP address of a domain name is.
There are several types of DNS records in the response to DNS query.

NS: points to a name server that contains the IP address of a name server, e.g. gatech.edu

A: contains the IP address of the domain name in the original query, e.g. www.cc.gatech.edu

MX: contains the address of the mail server for the domain, e.g. mail.gatech.edu

TXT: contains other useful information about a domain, e.g. it can be used to distribute public keys

8.3 DNS Caching Quiz

Changing a domain name into an IP address involves a large number of steps. To save time the record is cached on a local server for reuse later, so that the DNS server does not have to look it up again because the mapping is already stored in the cache. Each record has a TTL that states how long a record can be kept for future use. TTL is useful because a server may be moved to a new IP address.

8.4 Caching

Not only A records but also NS records are cached. DNS servers can also cache negative results such as a domain that does not exist. All cache data, whether it is positive or negative response, has a TTL.

8.5 Basic DNS Vulnerabilities

First of all we must be able to trust the domain name and address mapping provided by DNS. In fact many security policies depend on this, e.g. the same-origin policy of browsers or URL-based filtering. If the host address mapping provided by DNS can be forged, then the traffic intended for the original legitimate host is now destined to the wrong or malicious host.
There are several ways to forge the host address mapping

- the attacker can compromise the DNS servers, including cache poisoning (see below)
- the attacker can control the access point or gateway to intercept DNS queries and forge a response

A solution is to authenticate each request and response using cryptography, and DNSSEC is such a solution. However few have been using DNSSEC thus far.
The basic idea of DNS server attack in particular cache poisoning is that an attacker provides the local DNS server some false records and get the records cached. The existing defense in DNS is the use of a 16-bit request ID to link a response with a query. That is the attacker's response must have the ID that matches the ID of the original query, and we will discuss how an attacker can overcome this defense. A DNS cache can be easily poisoned if the DNS server does not use the IDs properly or the IDs are predictable.

8.6 DNS Quiz

DNS stores both the IP address and the domain name. All domain names and IP addresses are stored at the central registry. It takes 12 to 36 hours to propagate information to all DNS servers worldwide.

8.7 DNS Packet

There is a usual UDP header because DNS uses UDP, and the UDP payload is the actual DNS data. One of the most important fields in DNS data is the query ID which is a 16-bit random value. A DNS query contains a query ID and its response also carries the ID. A DNS server can use the ID to link a response to a query.
In a DNS query

- QR = 0 means this is query
- OP = 0 means this is a standard query
- RD = 1 means recursive queries on its behalf is desired

In a DNS response

- QR = 1 means this is a response
- AA = 0 means the server doesn't know the IP address of the query, so it responds with a series of NS records that should know how to handle the original query, both the domain names and IP addresses of these name servers are provided, these are called glue records
- RA = 0 means the server is busy thus recursive query is unavailable, the inquirer should contact the name servers in the glue record instead
- AA = 1 means this is the final or authoritative response which contains the IP address of the inquired domain name, caching TTL is provided, NS records that are in the same domain of the original queried domain are also cached

8.8 Poisoning Attacks

In traditional poisoning attack an attacker first sends a query to the local DNS server through a compromised machine within the domain. The local DNS server is then going to perform a recursive query with a query ID. The attacker is going to forge a response claiming it is the authoritative response using the forged IP address. But the attacker does not know the real query ID, so all he can do is send a flood of responses each with a guessed query ID. Hence this is a matter of the attacker being able to guess the correct query ID and reach the local DNS server faster than the legitimate response from the real DNS server. However if the attacker's attempt fails, the legitimate IP will be cached and the

attacker has to wait for TTL to expire before launching the whole attack again.
Kaminsky found an approach that is drastically more effective than the traditional attack. What he discovered is that we can go up one level and hijack the NS records instead. Suppose an attacker wants to forge the IP address of www.google.com, this time the inside help is going to send a query of a random domain within www.google.com instead. Now the legitimate response will say that this random domain does not exist but will provide the IP address of www.google.com. The attacker is attempting to do the same thing. What is new in the Kaminsky's Poisoning Attack is that when the first attempt fails the attacker can start immediately again. He doesn't have to wait for the TTL to expire, because he can simply use a different random domain which will immediately result in another query so that he can flood the DNS server again. It has been reported that such Kaminsky's poisoning attack can succeed in mere ten seconds.

8.9 DNS Defenses

The first few here simply make attackers do a lot more work in order to succeed

- increase query ID size
- randomize the source port (additional 11 bits)
- query twice so that the attacker has to guess correctly twice (32 bits), but DNS system cannot handle the load

More fundamentally we can use cryptography to provide authenticity of DNS records, and that is the idea behind DNSSEC.

8.10 DNSSEC

The goal of DNSSEC is to provide guarantees of the authenticity of DNS servers as well as the integrity of their responses. These guarantees are accomplished by having the DNS servers sign responses every step of the way using public-key cryptography. It is also assumed that the DNS servers themselves can be secured.

8.11 DNSSEC: DNS Signing

Suppose a local DNS server wants to look up wikipedia.org. They first query the root server. The root server provides the IP address of .org and signs it. The signature is based on the private key of the root server. The DNS server performs recursive query sending the request to .org, and the response contains the IP address of wikipedia.org signed with the private key of .org.

8.12 DNS Rebinding Attack

Even DNSSEC cannot prevent all DNS attacks. The DNS rebinding attack is one such example. To launch a DNS rebinding attack the attacker needs to only register a domain name e.g. evil.com and attract web traffic e.g. by running an advertisement in a frame. When evil.com is looked up the attacker answers with the IP address of his own server and use a very short TTL value. The attacker's server, evil.com, also serves the browser a malicious JavaScript. To circumvent the firewall when the malicious JavaScript issues a query to evil.com, the TTL has expired. The attacker then rebinds the host name, evil.com, to an IP address of an internal server so that the firewall thinks that evil.com is internal. Then the malicious script can easily exfiltrate information from the server to evil.com.
To mitigate such attack

browser: should use DNS Pinning—refuse switching to new IP address for domain, however this may break proxies, VPNs, dynamic DNS, and so on, therefore it is not consistently implemented in all browsers

server: should check host header for unrecognized domains and provide authentication of users stronger than IP

firewall: should implement a policy such that external domain names cannot resolve to internal IP addresses and stronger protection of browsers within the network

8.13 DNS Rebinding Quiz

In DNS rebinding attack the attacker exploits the same-origin policy.

9 Lesson 9: Advanced Malware Analysis

9.1 Malware Evolution

Malware keeps evolving fast and some of the traditional defense-in-depth mechanisms are not adequate

network-level protection: we have firewall as the prevention mechanism and IDS as the detection mechanism, but for firewall command-and-control traffic can look just like normal traffic such as visiting a webpage, and for IDS that analyzes the payload of traffic the encrypted or specially encoded malicious contents can evade such analysis

host user access control: most often since the users often do not understand the security implications they will simply say yes

antivirus software: the traditional signature matching approaches are not effective where malware uses obfuscation techniques

Hence we have to continue to develop more complex behavior-based analysis approaches.

9.2 Malware Obfuscation

Since malware contents are encrypted and look random, there is no single signature that matches all the instances. Thus a signature scanner that tries to identify malware by its unique strings would not be effective.
Code that decrypts malware in runtime so that the malware can execute needs to be included. However we cannot simply use the code that is part of the compressed or the encrypted malware as a signature because legitimate programs can contain such instructions.

9.3 Malware Obfuscation: Packing

One of the most widely used obfuscation techniques is packing, a technique whereby parts or all of an executable file are compressed, encrypted, or transformed in some fashion. A packing tool would transform the original malware program so that the transformed code looks random. Because it is encrypted a key is randomly generated, and this happens each time the packing tool is running on a malware program. Hence even for the same malware program each packed instance will look different. Therefore a signature-based approach is not effective in detecting the malware. Furthermore the transformed machine code looks like data. Therefore a network IDS that looks for executables and

email attachments will miss it.
Even antivirus companies can detect and start analyzing a zero-day malware, by the time the malware is de-obfuscated the analysis may become obsolete. Because the attacker's server can continuously send an updated or new malware. This is called **server-side polymorphism** which attacks the heart of the traditional host-based antivirus model by automating mutations. Effectively the defenders have to deal with zero-day malware constantly (e.g. Waledac).

9.4 Obfuscation Quiz

The following obfuscation techniques hide malware from

user: rookits

security such as detection: thoroughly mapping security sites and honeypots

researcher: Using nonce-based encryption schemes would make crypto-analysis more difficult

9.5 Malware Analysis

The benefits of malware analysis include

- if we understand malware's network behaviors, we can create rules to detect and block malware traffic
- if we understand malware's on-host behaviors, we can repair the compromised hosts
- if we have knowledge of the malware's support infrastructure on the Internet and its evolution history, we can analyze its trend and threat scope such as how widespread the malware is and the likely next target etc.

A challenge in malware analysis is that the volume of malware is huge. Available and automated tools make the job of creating obfuscated malware very easy. Since malware creation is already automated, it is imperative that malware analysis has to be automated as well.
In addition to automation, malware analysis also needs to be transparent so that the malware does not know it is being analyzed. Otherwise the malware may refuse to run or purposely alter to its behaviors to fool the analysis. This is the so-called **malware uncertainty principle**. Unfortunately robust and detailed analyzers are typically invasive so that malware can detect them. In fact malware authors already tried to actively detect malware analysis tools. In malware creation toolkits there are standard point-and-click options to add logic to detect various malware analysis methods. To fulfill this transparency requirements

higher privilege: a malware analyzer should be at a higher privilege level than a malware so that the malware cannot directly access or know about the analyzer

no non-privileged side effects: a malware may try to find a side effect of analysis, which should be privileged so that the malware cannot directly get the answers back, in addition since the analyzer is at a higher privilege level, it can actually lie the answers back to the malware

same instruction execution semantics: a malware should get the same correct result of instruction execution as if the analyzer is not present

identical exception handling and notion of time: a malware should see identical signals and instruction handlings as well as time durations

In terms of fulfilling the transparency requirements most interesting tools fall short

in-guest tools: the analyzer has the same privilege as the malware, and some analysis side effects can be detected by the malware without privilege operations, the exceptions trapped by the analyzer may be discovered by the malware

virtual machine tools: e.g. VMW, some analysis side effects can be detected by the malware without privilege operations

emulation tools: e.g. QEMU, the execution semantics of instructions maybe different from real hardware

Emulation-based malware analysis tools are the most widely used. But they have major shortcomings. The main issue here is that these emulation environments do not fully emulate the hardware. There are corner cases where a set of instructions give different results on emulation environments versus on hardware, and there have been attacks based on these corner cases to detect the emulation environments. But the bigger problem is that there is no way to eliminate all these corner cases, as dictated by the so-called EQTM theory—determining whether the languages of two Turing machines are equal is undecidable.

9.6 Identical Notion of Time

The most challenging transparency requirement is the identical notion of time, in particular if a malware uses network timing measurements to infer the presence of analyzer because the analyzer causes delay or the website that the malware connects to is not a real one. There are many direct or indirect measurements that a malware can perform and it is impossible to identify all of them. In fact the problem of identifying all network-based timing measurements is equivalent to the problem of detecting and removing all covert channels. This problem has been proved to be undecidable.

9.7 Analysis Difficulty Quiz

The four basic types of malware analysis in ascending level of difficulty are

fully automated analysis: use fully automated tools to analyze malware, easiest thus should be the first performed

static properties analysis: examine the static properties of malware including metadata, strings embedded, header details etc.

interactive behavior analysis: run the malware in a protected and isolated environment

manual code reversing: use a disassembler and decompiler to recreate the malware code

9.8 Analysis Technique Quiz

The four basic types of malware analysis in ascending level of information that can be revealed are the same as the level of difficulty. Because it is not surprising that the harder an analyst's technique is the more information it can yield.

9.9 Robust and Efficient Malware Analysis

robust: transparent and not easily detected and evaded by malware

efficient: automated and fast

We focus on host-based analysis i.e. learning malware on a machine and analyze its behaviors. The transparency requirements are fulfilled as follows

higher privilege: put the analyzer in hardware or a virtual machine

no non-privileged side-effects: the analyzer will inevitable introduce side effects which can only be learned through privileged instructions, the analyzer, since it is at a high privilege level, can also lie the answers back to the malware

identical basic instruction execution semantics: need to use the real hardware

transparent exception handling: exception handling should be the same as if on real hardware

identical measure of time: need to know the timing of each instruction I/O and exception handling on the real hardware, ensure that the malware can only use privileged instructions to get timing measurements on the host so that the analyzer can lie the answers back

9.10 Ether Malware Analyzer

Ether, built at Georgia Tech, is an example of a transparent malware analyzer. It fulfills the transparency requirements as follows

higher privilege: Ether is built using Intel VT for hardware virtualization, the hypervisor of which has higher privilege over the OS kernel, therefore Ether has higher privilege than the malware, some of the hardware supported traps further guarantee this higher privilege

no non-privileged side-effects: Ether runs outside of the virtual machine where the malware runs, therefore there is minimal side-effects that malware can directly observe

identical basic instruction execution semantics: Ether uses hardware-based virtualization, therefore the instruction execution semantics are the same as on hardware

transparent exception handling: hypervisor pre-empts before OS exception handling

identical measurement of time: the operations come down to the use of read time-stamp counter (RDTSC) instruction which is a privileged instruction, therefore the Hypervisor and Ether can control the return results to the malware

Ether has a component with Xen, the hypervisor, and the rest of Ether is Dom0, a separate privileged virtual machine. The malware runs on a separate user-level virtual machine called DomU. Ether provides a fine-grained instruction by instruction examination of malware, and also a coarse-grained system call by system call examination.
Two tools were created to evaluate Ether

EtherUnpack: used for fine-grained tracing of individual instructions, evaluated by obfuscating a test binary and looking for a known string in the extracted code

EtherTrace: used for fine-grained tracing of system calls in guest OS, evaluated by obfuscating a test binary that executes a set of known operations and then observing if they were logged by the tool

The results show that Ether has much better transparency than other tools.
For each major categories of analysis approaches attackers come up with obfuscation techniques to defeat such analysis

static analysis based approaches: polymorphism, metamorphism, packing, opaque predicates, anti-disassembly

dynamic malware analysis: trigger-based behavior (logic bombs, time bombs, anti-debugging, anti-emulation etc)

Since simple dynamic analysis now becomes inadequate, researchers have come up with various ways to force execution of this malware. Of course this battle between analysis and obfuscation will continue.

9.11 Malware Emulators

A recent trend is that malware authors are starting to use emulator-based obfuscation techniques. This is an instruction-level approach. There are several commercial tools out there, and they can be used for legitimate purposes such as digital rights management.
Emulator-based obfuscation works as follows: suppose the original malware is for x86 set architecture. It is then transformed into a bytecode program of an arbitrary language L. The emulator based on L will emulate this microprogram on x86. That is the obfuscated malware includes both the bytecode program and its emulator. When the obfuscated malware runs on x86, the emulator will emulate this bytecode program and execute the malware logic.
The impact of emulator-based obfuscation on malware analysis is that

- The malware program now is a bytecode of arbitrary language L which is unknown. In fact it can be randomly generated. As a result we cannot perform pure static analysis.
- We can perform analysis on the emulator. But it is only specific to the language L and the malware bytecode program is only one of the possible inputs to the emulator.
- We can perform dynamic analysis including some local code analysis. We call this greybox methods. But such analysis is actually performed on the emulator not the malware code directly. Because the executable is the emulator while the malware bytecode is only the input. Therefore the analysis results that we get are from the emulator not directly from the malware bytecode.
- Manual reverse-engineering cannot scale because each instance can have a different language L and a different emulator. Since the process of creating such confiscated malware is automated, we also need an automated approach to reverse engineer these emulators, which should not require any knowledge about the language L and should be generic enough to work on a large class of emulators. However theoretically the feasibility of automated reverse engineering is an undecidable problem. Nevertheless most of the emulators have a fetch-decode-execute behavior that can be identified at runtime and can be the starting point of our reverse engineering.

9.12 Approach of Emulation

In fetch, decode, and execute, the emulator fetches the next bytecode, decodes to get its opcode, looks up the execution routine for this opcode and then executes it. The execute routine executes real x86 machine code. The virtual program counter or VPC is a maintained pointer to the next bytecode to fetch from.
There are quite a few challenges of reverse engineering emulator-based obfuscation

- we don't know where the bytecode resides in memory

- the code that corresponds to decode, dispatch, and execute phases of the emulator is unknown

The malware author can also make reverse engineering more difficult by changing how the emulator works, e.g.

- context can be maintained in many different ways
- an attacker may complicate VPC identification by maintaining it in different correlated variables
- bytecode program may be stored in non-contiguous memory

We develop a tool to automatically reverse engineer emulator-based malware. The first step is to identify abstract variables in particular pointers, and of course one of the most important pointers we need to identify is VPC. From the fetch, decode, and execute operations of the emulator we can identify the opcode and operands as well as the execution routine of the malware bytecode. We can then construct a control flow graph (CFG) of the original malware, which will tell us the behaviors of the malware.

10 Lesson 10: Mobile Malware

10.1 Mobile Device Quiz

Wikipedia says that a mobile device must be mobile. Therefore a smartphone by itself is not mobile. However a smartphone held by a person can be mobile. We are not going to use this strict definition of mobile device. Instead we are going to use this common definition. That is a smartphone is a mobile device.

10.2 Forensics Quiz

The following characteristics are associated with mobile devices (M) versus stationary computers (C)

- (M) specialized hardware vs. (C) standardized hardware
- (M) many different (versions of) OS vs. (C) usually runs Windows, Mac OS or Linux
- (M) large number of accessories (cameras, GPS) vs. (C) large storage capability

10.3 iOS Malware

Some apps may appear to be providing useful functions but secretly still use confidential information. These apps are considered malware. But malicious apps may still get on victims devices even after it is taken off the app store. One technique is to exploit the design floor in Apple's DRM scheme. Apple allows users to purchase and download iOS apps from their app store through the iTunes client running on their computers. In a FairPlay man-in-the-middle attack attackers purchase an app from the app store and then intercept and save the authorization code. They then develop a program that simulates the iTunes software and install the software on a victim's computer. This fake iTunes software can check the victim's iOS devices to believe the app was purchased by the victim. Therefor the user can install apps they never actually paid for, and the attacker can install malicious apps without the user's knowledge. This attack continues to work even after the malicious app is removed from the app store.

10.4 Android Malware

There is a large increase of the number of Android malware and the majority is still SMS Trojans. We discuss a few here

AccuTrack: turns an Android smartphone into a GPS tracker

Ackposts: steals contact information and uploads them to a remote server

Acknetdoor: opens a backdoor and sends the IP address to a remote server

Steek/Fatakr: fraudulent app advertising an online income solution, some have the capability to steal privacy related information and send SMS messages

Tapsnake/Droisnake: posts the smartphone's current location to a web service

ZertSecurity: steals users' banking account details

Zitmo/Citmo: steals confidential banking authentication codes sent to the infected device

Nonetheless the risk is relatively low, because in general mobile devices are still less powerful than desktops and laptops. Furthermore the app review process and the sandbox space execution model also means that mobile devices are in general more secure.
In addition to protection against malware, a bigger problem is the loss of devices and the loss or the theft of data. Therefore secure companies also try to protect and manage mobile data.

10.5 Lifetime of iOS Malware

The stages include

produce: toolchain attack, risky third-party SDKs, repackaging

distribute: enterprise distribution, app store, FairPlay MITM

do evil: private APIs, hooking, design flaws

make profit: advertisement, accounts, app promotion, user privacy

10.6 Toolchain Attacks

Toolchain attack is one approach to produce malware. For example

XcodeGhost (2015): malware found in unofficial distributions of Xcode targeted at Chinese developers, apps compiled with these versions of Xcode are infected with XcodeGhost, which collect information on devices and upload it to command-and-control center

Jekyll (2013): attacks App store review process by deliberately creating a vulnerable app, once the vulnerability is exploited the Jekyll app can activate new execution paths via return-oriented programming (ROP), and then the app can send SMS, email, tweet, and so on, the App store review process cannot reveal these malicious paths because it doesn't have the correct input, which can generate a new control flow not observable in the app review process

10.7 Toolchain Attacks Quiz

Information that an infected app can obtain about the device include app's name, app's bundle identifier, the device's name and type, device UUID, current time, system's language and country, network type.

10.8 Hardening the Toolchain Quiz

The four areas of the C based toolchain where hardening can occur are configuration, preprocessor, compiler, and linker. Essentially all the main steps of the toolchain can be hardened.

10.9 Mobile Malware Detection

Kirin: a simple system that looks for suspicious combination of permissions based on the definition of 9 manually compiled rules, provides basic support for multi-app analysis

RiskRanker: a simple static analysis tool using manually defined suspicious features such as DVM cod loading from assets, call to crypto-related APIs before loading native code, and sensitive calls without user interaction

DroidRanger: based on both static and dynamic analysis, using manually defined heuristics such as loading of native code from suspicious websites, manifest information, packages, location of used resources, sensitive API tracing, system call level tracing

DREBEN: a lightweight static analysis tool, using a machine learning algorithm called support vector machine (SVM) based on data attributes such as used hardware components, requested permissions, API calls, filtered intents, and network addresses, it runs on real devices

Many malicious apps are in fact repackaged version of legitimate apps. This is actually the most effective way to distribute malware, because a popular or cool app already has a large number of users. As such there are research systems on clone detection, for example

DroidMOSS: uses fuzzy hashing of Java methods to match and detect clone code

DNADroid: performs similarity analysis based on PDGs which are program dependency graphs between methods

PiggyApp: focuses on piggybacked apps

AdRob: investigates AD-related issues

There are a few sandboxes for mobile malware analysis which enable dynamic analysis, e.g. Andrubis, Mobile Sandbox, and Apk Analyzer. Many dynamic analysis and detection tools use system call information, for example PREC, which stands for particle root explore containment, can dynamically identify system calls from high risk components e.g. third-party native libraries, and execute those system calls within isolated threads. Therefore PREC can detect and stop root exploits with high accuracy while imposing very low interference to benign applications.

10.10 Information Leakage Detection

Apps that leak sensitive information can often be considered malware even though many think that they are in a gray area. There are several approaches to detect leakage, for example

PiOS: performs static analysis on iOS apps to detect information leakages, which requires code decryption, analysis on ARM assembly, and dynamic dispatch implemented with indirect jumps

TaintDroid: uses taint tracking to perform information flow analysis, i.e. it analyzes how data from a source, such as address book, flows to a sink, such as the internet, it supports both variable-level and message-level taints

Another approach is to check if an app does what it promises to do. For example WhyPer compares the app's permissions against its description. The analysis is based on natural language processing (NLP) techniques, and is limited to permissions related to camera, microphone, and contact list.

10.11 STAMP Admission System

STAMP is a research system designed to analyze mobile apps and decide if the mobile apps meet security and privacy requirements. The system is intended to be used in an App Store to decide if an app should be admitted. The system uses both static and dynamic analysis approaches because they have pros and cons of their own.

10.12 Data Flow Analysis

One of the most important analysis is data flow analysis. Data flow analysis can be useful for

- malware or greyware analysis to find out what information is being stolen, based on which enterprise specific policies can be improved
- checking external app to make sure that there is no API abuse or data theft
- informing users about potential privacy implications
- discovering vulnerabilities in applications, e.g. accepting data from untrusted sources

Performing data analysis on the whole 3.4M+ lines of code of Android system would take a long time thus is not practical. The STAMP approach is to abstract the Android stack into models. These models include sources and sinks, data structures, callbacks and we will focus on data flows.

10.13 Data Flows

There are more than 30 types of sources (e.g. account data, device ID, calendar) and more than 10 types of sinks (e.g. browser, system logs, file system). Each pair of source and sink is a flow type thus there are close to 400 flow types. The cloud computing industry represents a large ecosystem of many models, vendors, and markets, therefore our definition here is very general.

11 Lesson 11: Cloud Computing: VM Monitoring

11.1 Definition of Cloud Computing

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage, applications, and services. Cloud computing uses resources that can be rapidly provisioned and requires low management overhead.

11.2 Cloud Characteristics Quiz

Cloud computing is characterized by on-demand measured self-service with rapid elasticity.

11.3 Cloud Service Models

There are three cloud service models

software as a service (SaaS): enables a customer to use a provider's applications running on a cloud infrastructure, which can be accessed from various kinds of devices through a thin client interface such as the web browser

platform as a service (PaaS): enables a consumer to deploy onto the cloud infrastructure consumer created applications using programming languages and tools supported by the cloud provider

infrastructure as a service (IaaS): enables a consumer to provision processing, storage networks, and other fundamental computing resources on the cloud, where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications

11.4 Services Quiz

The exemplary products of the three service categories are

SaaS: Knowledge Tree

PaaS: Google Apps, Salesforce

IaaS: AWS, Microsoft Azure

11.5 Cloud Deployment Models

There are several cloud deployment models

private cloud: the cloud infrastructure is operated solely for an organization, may be managed by the organization or a third party and may exist on-premise or off-premise

community cloud: the cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns, may be managed by the organizations or a third party and may exist on-premise or off-premise

public cloud: the cloud infrastructure is made available to the general public or a large industry group, and is owned by an organization selling the cloud services

hybrid cloud: a composition of two or more of these models

11.6 NIST Cloud Definition Framework

Any one of the four deployment models can use any one of the three service models. All cloud environments share some essential characteristics, which include on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. They also share some common characteristics, which include massive scale, homogeneity, virtualization, low-cost software, resilient computing, geographic distribution, service orientation, and advanced security.

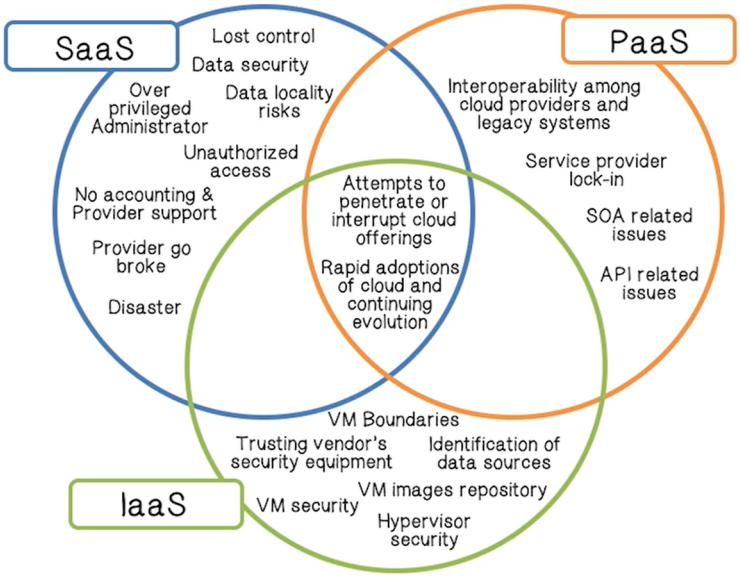

Figure 1: The NIST cloud risk service categories

11.7 NIST Risk Identified Quiz

The cloud risk classification according to the three service models is depicted in Figure 1 below.

11.8 Cloud Security Concerns

Security and data privacy are the critical barriers to adopting cloud computing. In fact according to the IDC Enterprise Panel survey 2008 security is the main concern when people consider moving to the cloud.

11.9 Analyzing Cloud Security

There are several key security issues, for example

- how do we trust the cloud computing environment if we don't manage it
- multiple different organizations may be using the same cloud computing environment
- in terms of data protection, obviously we should use encryption to protect confidentiality, but we also need to consider various compliance issues

These security issues become very challenging because the clouds are massively complex systems. Although the primitives and common functions are simple, the complexity comes from the fact that these primitives and function units are replicated thousands of times.
On the other hand cloud security is a tractable problem because there are both advantages and challenges associated with cloud computing. The advantages include

- public data can be moved from internal network to an external cloud to reduce the exposure of internal sensitive data
- cloud homogeneity makes security auditing/testing simpler, because it is easy to manage testing and security patches when most systems in a cloud computing environment use the same software
- cloud enables automated security management, e.g. disaster recovery is easy and fast because it is easy to set up the same system

However there are also challenges, for example

- customers need to trust vendor's security model
- customers cannot respond to all the audit findings
- customers cannot investigate directly and have to obtain support from vendor
- customers have to rely on cloud providers who are the direct admins
- some software running in the cloud is proprietary thus cannot be examined
- loss of physical control of the computing resources

11.10 Security Relevant Cloud Components

All the core components of a cloud computing environment are relevant to security. These include

cloud provisioning services: the advantages and challenges are

advantage: they are

- rapid reconstitution of services
- greater data availability resulted from provision of multiple data centers
- advanced honeynet capabilities

challenge: the impact is disruptive if the provisioning service is compromised

cloud data storage services: the advantages and challenges are

advantage: they are

- data can be fragmented and dispersed thus are more resilient to attacks
- data can be encrypted at rest and in transit
- automatic backup and replication of data

challenge: they are

- data from multiple organizations may be put in the same storage server thus requires isolation management
- storage server may be in a different country thus exposed to foreign government
- storage controller is a single point of failure

cloud processing infrastructure: the advantages and challenges are

advantage: we can secure the master copy of a program and then replicate that copy throughout the cloud computing environment

challenge: they are

- multiple applications can be running on the same physical machine, and therefore achieving a real isolation is not easy, e.g. cache and memory access are side channels that can leak information
- the assumption that the hypervisors or the virtual monitors are secure may be invalid

cloud support services: the advantages and challenges are

advantage: provide on-demand security controls for customer applications

challenge: cloud providers need to make sure that customer applications will not cause security problems in the cloud environment through certification, accreditation etc, because there is additional risk when integrated with customer applications

cloud network and perimeter security: the advantages and challenges are

advantage: since the computing systems are distributed denial of service attacks are harder to succeed, standard perimeter security measures such as IDS and firewalls are deployed by default

challenge: it is challenging to create security zones where the resources share a common security exposure or security risk

elastic elements: such as storage, processing, and virtual networks

11.11 Cloud Security Advantages

The security advantages a cloud computing environment can provide include

- data fragmentation and dispersal
- dedicated security team
- greater investment in security infrastructure
- fault tolerance and reliability
- greater resiliency
- hypervisor protection against network attacks
- possible reduction of certification & accreditation (C&A) activities
- simplification of compliance analysis
- data can be held by unbiased party (cloud vendor assertion)
- low-cost disaster recovery and data storage solutions
- on-demand security controls
- real-time detection of system tampering
- easy and rapid reconstitution of services
- an advanced honeynet can be deployed

11.12 Cloud Security Challenges

The challenges associated with a cloud computing environment include

- have to consider various international privacy laws (e.g. EU Data Protection Directive, US Safe Harbor Program), data can be subject to subpoenas from government including a foreign government
- need very strong isolation protection since multiple organizations may be using the same cloud computing environment

- logging is challenging since computation and data can be distributed
- it is challenging to provide consistent quality of service since the cloud is a distributed computing environment
- there could be potential data ownership issues since the customer does not own the computing environment
- have to depend on a secure hypervisor
- the cloud is an attractive target to attackers
- still need the security of the operating systems running in the virtual machines
- the potential impact of a successful attack on the cloud can be massive
- for organizations that use public clouds it is challenging to reconcile the security policies of the internal network versus the public cloud
- for public cloud that uses SaaS model a customer cannot control what software or what version to use
- need to encrypt not only the data at rest but also access to the cloud resource control interface, administrative access to OS instances, and applications, because otherwise every service can learn about the computing tasks being performed

11.13 Cloud Security Quiz

Close to 90% of the service providers encrypt data in transit. In contrast Only 10% of the providers encrypt data at rest. Nevertheless there's plenty of data that does not require security protection, and therefore such data does not need to be encrypted at rest.

11.14 Cloud Security Additional Issues

There are some additional issues associated with cloud security, in particular

- need to consider the privacy impact when personal identifiable information or PII data is moved to the cloud
- customers should be aware that they need to negotiate with cloud providers to obtain a security service level agreement (SLA) and make sure that they have proof that the SLA has been satisfied
- a cloud provider needs to consider a number of compliance issues (e.g. HIPAA for healthcare data and PCI for payment data)

11.15 Cloud Security Architectures

The primary technologies cloud computing is built upon include virtualization, grid technology, service-oriented architectures, distributed computing, broadband networks, browser as a platform (BaaP), and free and open source software. Other technologies include autonomic systems, web 2.0, web application frameworks, and service level agreements or SLAs.

11.16 Virtualization Quiz

Virtualization requires at least one instance of an application or resource that is to be shared by different organizations. Sharing between organizations is accomplished by assigning a logical name to the resource and then giving each request a pointer to the resource.

11.17 Virtualization Quiz 2

Virtualization involves creating a virtual machine using existing hardware and operating systems. The virtual machine is logically isolated from the host hardware.

11.18 Virtualization Quiz 3

type 1 hypervisor: installed on a bare system, the operating system runs in a virtual machine on top of the hypervisor

type 2 hypervisor: emulates the device with which a system normally interact, the host operating system is installed

11.19 Platform Virtualization

The most important technology for cloud computing is virtualization, because cloud computing relies on separating applications from the underlying infrastructure. The key to virtualization is the hypervisor or the virtual machine monitor. In addition to increased utilization (e.g. running many virtual machines on a single hardware) and portability, virtualization can also improve security.

11.20 Kernel-Level Attacks and Security Tools

A kernel-level security tool has high privilege than user level applications, and therefore can detect and remove a malware at user level. However since the security tool has the same privilege as attacks in the kernel such as a rootkit, it can be compromised. In other words the security tool cannot be isolated or protected from a kernel-level attacker.

11.21 Hypervisor Based Approaches

With virtualization we can put the security tool in a separate virtual machine so that the security tool is isolated from the virtual machine that has the malware. With the help of the hypervisor the security tool can perform introspection of the other virtual machine.

11.22 VirtualBox Security Quiz

It is not safe to

- set the clipboard sharing between the VM and the host to be bidirectional
- allow the VM to read/write files on the host machine with the same privilege as the host machine, because this is similar to allowing an attacker to write files on your machine

It is safe to disconnect the VM from the internet when opening questionable files, because this prevents potential malware from contacting its command & control server.

11.23 Monitoring Memory

One of the most essential tasks of virtual machine security is memory analysis, because memory is the only reliable source of the current state of a computer. Using memory analysis we can find out

- a list of the running and (some) killed processes
- the encryption keys being used and the decrypted data
- network socket and data
- OS-level accounting information
- user input such as key strokes and mouse movements
- screen captures and the graphical elements of an application

11.24 Production Level Systems

The security in a controlled virtual machine is smaller because it can run a stripped down OS. The security machine gets a raw memory view meaning that it has seized the physical memory from other virtual machines. However if we want anything useful from the raw view we need to rebuild the abstraction levels on our own.
There are several types of monitoring

passive monitoring: the security virtual machine takes a view of the memory of another virtual machine periodically without any timing synchronization between the two virtual machines

active monitoring: the security virtual machine takes a view of the memory of another virtual machine when there is an event notification sent from that virtual machine

An important goal of memory monitoring is to locate important data structures in memory for security analysis e.g. the process list.

11.25 Passive Monitoring

Also known as performing virtual machine introspection, the main steps of passive monitoring are

1. the security tool in the security virtual machine performs an API call to access a kernel symbol
2. the address of the kernel symbol is looked up
3. the page tables for the memory of the user virtual machines are traversed to locate the kernel data
4. the pointer to the memory is returned to the security monitor

We use a virtual machine introspection library called libVMI to convert raw memory view into something meaningful, such as virtual addresses, kernel symbols etc.

11.26 libVMI

First released as XenAccess in Spring 2006, libVMI is open source, providing access and analysis of virtual addresses, kernel symbols etc. With libVMI you don't need to write a lot of code in order to obtain useful runtime information of the other virtual machine. libVMI features include

- perform virtual memory translation for kernel symbol, DTB, PTD etc.
- place monitoring hooks in a guest virtual machine to trap exceptions and page faults, and events like memory read-write-execute, register read-write, interrupt, and single stepping of instructions

11.27 Active Monitoring

Active monitoring is event driven which allows for enforcing security policy and stopping attacks before they happen. The security application receives event notification from the guest virtual machine when one of the hooks executes. The hook invokes trampoline which transfer the control to the security application. The security application then performs virtual machine inspection of the guest virtual machine.

11.28 Active Monitoring Challenge

There are several challenges emerged in active monitoring. The first is high overhead which comes from several sources

invocation cost: switching from a virtual machine to the hypervisor and to another virtual machine is very expensive, especially for fine-grained monitoring

introspection cost: accessing the memory of untrusted virtual machines requires calls to the hypervisor for mapping pages to the security virtual machine

11.29 Secure In-VM Monitoring

Can we have both the security benefit provided by virtual machine monitoring which means out-of-VM monitoring and the efficiency of traditional monitoring which means in-VM monitoring? We have developed a secure in-VM monitoring approach called SIM which stands for secure in-VM monitoring. It provides the same security as the out-of-VM approach, and its performance is similar to the traditional in-VM approach. The main idea is to use hardware virtualization features to minimize the need to switch to hypervisor. In other words we can read or write the memory of an untrusted virtual machine at native speed.

11.30 SIM Monitoring Requirements

We want

fast invocation: invocation of monitor happens without privilege change, there is no need to switch to the hypervisor

data read/write at native speed: native instructions should be able to read/write data directly without going through the hypervisor

11.31 SIM Security Requirements

They are

- the code and data of the security tool need to be isolated from the untrusted virtual machine
- the invocation for event-handling should be secure
- the security tool should not rely on untrusted code and data

11.32 SIM Design

In operating systems paging-based virtual memory is generated by creating page tables that map virtual addresses to physical addresses. An operating system creates a separate page table for each process so that it can have its own virtual memory address space hence necessary isolation can be achieved. The SIM address space contains all kernel code data and also the SIM data in its own address space. Therefore the instruction as part of the security monitor can access in native speed. The guest operating system has it's own virtual address space. Thus although we put SIM into the same virtual machine as the guest operating system, they have their own separate and different virtual address spaces. Security monitoring is done through the entry and exit gate since this requires switching of address spaces thus we need to modify the CR3 register contents directly. Intel VT contains a feature that doesn't trigger a VM exit to the hypervisor if the CR3 is switched to an address space that's predefined and maintained by the hypervisor. Therefore by predefining the SIM address space and the system address space we can achieve fast switching without exiting to the hypervisor.
We also use the hypervisor memory protection to protect the security of the SIM address space and the entry and exit gates. The entry and exit gates are the only ways to switch between the SIM address space and the system address space. We use an Intel hardware feature called the *last branch recording* in the invocation checker. With last branch recording we know the last few basic blocks leading to the entry gate.
The SIM security tool is self-contained, meaning that it does not call any code or use any data from the kernel of the untrusted region.

11.33 Protected Address Space Generation

The SIM code and data cannot be viewed by the untrusted system. The entry gate and exit gate are executable in both spaces. Their security is provided by memory protection in the hypervisor and the invocation checker in SIM. The kernel code and data from the untrusted system can be read directly by SIM, but SIM does not execute such untrusted code and data.

11.34 Monitor Invocation Overhead

We perform experiments to measure the overhead of SIM and compare it with the out-of-VM monitoring approach. We use null handlers so that only switching time is measured. The result shows that SIM is faster than the out-of-VM approach by an order of magnitude.

12 Lesson 12: Property-Preserving Encryption: Oblivious RAM

12.1 How Do Data Breaches Happen

We want to protect the data by encrypting the data so that the adversary cannot read our data, but at the same time we also want the application to continue to work.

12.2 Encryption Quiz

property preserving encryption: some selective properties of the alternate data are preserved such as the order

searchable encryption: the encrypted data can be searched using the encrypted keywords

secure computation: multiple parties can compute a function using inputs that are kept private

homomorphic encryption: computations performed directly on encrypted data match the result of the computations on the plaintext

functional encryption: the possession of a secret key allows someone to learn the function that is being encrypted

12.3 Property Preserving Encryption

Property Preserving Encryption or PPE is one way to protect data while allowing applications to continue to work. This approach is widely deployed in various environments including the cloud. PPE has the following advantages:

deployability: no need to change application and database servers

expressiveness: supports common data retrieval methods including SQL queries

efficiency: reasonably efficient with around 25% overhead

security: widely believed to be secure but really???

12.4 PPE Leak Quiz One

In standard encryption there is no preserving of properties. In PPE equality is preserved, but what is leaked is frequency. Because we now know that there's one value that appears N times.

12.5 PPE Leak Quiz Two

In order-preserving PPE not only the frequency but also the order is leaked.

12.6 PPE Leakage

Attributes in electronic medical records are typically used in equality queries or ordering. Hence to ensure that the applications will continue to work these attributes will be encrypted to preserve equality or order. These attributes are sensitive to either the hospital or the patient or both.

12.7 Data for Attributes

The data as we can obtain from the hospitals is encrypted. On the other hand there is information or auxiliary data that is public. We will show that using both the encrypted data and auxiliary data, an attacker can launch inference attack to obtain plaintext data.

12.8 Encryption Attacks

An attacker can use frequency analysis to defeat equality-preserving encryption.

12.9 Frequency Analysis Attack

We first sort the number of days a patient stays in hospital, and then sort and record the frequency of the auxiliary data—the public information of how frequent a patient will stay for one day versus two days and so on. By matching these two histograms we can link a ciphertext value to a plaintext value. Therefore with frequency analysis we can uncover the plaintext data without the encryption key.

12.10 l_p-Optimization Attack

This attack can be generalized and optimized. The basic idea is to find an assignment from ciphertext to plaintext that minimizes a given cost function which measures the distance between the frequency histogram of the encrypted data and that of the auxiliary data. This minimizes the total mismatch in frequencies across all plaintext-ciphertext pairs. There is algorithm that can find the assignment that obtains the minimum cost, with which we can find the optimal assignment from ciphertext to plaintext and then decipher the original encrypted data.

12.11 Optimization Attack Analysis

At least x fraction of records was recovered for y fraction of hospitals. It is (100%, 95%) for sex, (40%, 50%) for disease severity, (40%, 28%) for major diagnostic category, (10%, 85%) for age, and (83%, 50%) for length of stay.

12.12 Cumulative Attack

We can use sorting attack for order-preserving encryption. But a more effective attack is cumulative attack. Combining ordering with frequencies the attacker can tell for each ciphertext C what fraction of the encrypted values are less than C. More formally it is known as the empirical cumulative distribution function of the dataset or CDF. In cumulative attack an attacker leverages the CDF to improve the ability to match plaintext with ciphertext. Intuitively if a given ciphertext is greater than 90% of the ciphertext in the encrypted data, then we shall match it to a plaintext that is greater than about 90% of the auxiliary data. This problem belongs to a category of Linear Sum Assignment Problem (LSAP). Hence we can use an algorithm to find the mapping from plaintext to ciphertext that minimizes the total sum of mismatch in frequency + mismatch in CDF across all plaintext-ciphertext pairs.

12.13 Cumulative Attack Analysis

With frequency + CDF matching, the (x, y) is improved to (100%, 100%) for disease severity, admission month, mortality risk, and length of stay and (83%, 99%) for age.

12.14 Attack Recap

The result of attacks on electronic medical records shows that the confidentiality of many attributes is compromised.

12.15 Suppose We Don't Trust the Cloud

Suppose we don't trust the cloud provider thus encrypt our data on the cloud storage and keep the keys to ourselves. When we need to use the data we can fetch the encrypted data to our environment, decrypt the data, then use the application in our local environment. The data access patterns can still leak information such as what kind of computing tasks are being performed.

12.16 Oblivious RAM

A promising approach to eliminate this kind of leakage is to use oblivious RAM or ORAM. ORAM can hide access patterns. The main idea is that with ORAM for any fixed size request sequence the associated storage accesses observed by the cloud are statistically independent of the requests, i.e. while the cloud provider can still observe data access, the access patterns are independent of the actual data requests. Some of the main techniques include

- O data access
- operate on fixed size data blocks
- encrypt blocks with ciphertext indistinguishably, i.e. not using property-preserving encryption
- dummy data access (both read and write regardless whether the original request is read or write), re-encryption, shuffling etc.

12.17 ORAM Quiz

ORAM clients must have a private source of randomness, because they must generate random access patterns. Each access to the remote storage must have a read and a write, because we want to hide from cloud providers the fact that we are only reading or writing data therefore we will include dummy reads and writes.

13 Lesson 13: Botnet Detection

13.1 Network Monitoring

Attack traffic used to be obvious. For example the payload of a packet may contain an exploit to a known vulnerability and therefore a signature can be used to detect such attack, or a network monitor can detect a denial-of-service attack or spam activity by analyzing the volume and rate of network traffic. The typical network monitoring systems are firewalls and network intrusion detection systems (IDS).
Increasingly the traditional firewalls and network IDS are becoming less effective. First of all mobile devices are now widely used. A mobile device can be compromised when an employee is on travel. When the employee brings the mobile device back into the company's network it effectively has bypassed the perimeter defense. In addition attack traffic now is very subtle and they often look like normal traffic, e.g. a botnet's HTTP-based command-and-control traffic would look like normal legitimate web traffic. Therefore we need more advanced network monitoring systems to detect this new generation of attacks.

13.2 Bot Quiz

A bot is often called a zombie as it is a compromised computer controlled by malware without consent and knowledge of the user.

13.3 Botnet Quiz

A Botnet is a network of bots controlled by a bot master or an attacker. More precisely a botnet is a coordinated group of malware instances that are controlled via command-and-control (C&C) channels. C&C can use centralized architectures or distributed architectures. Botnet is a key platform for fraud and other for-profit exploits.

13.4 Botnet Tasks Quiz

Botnets commonly perform

- more than 95% of all spam
- all distributed denial of service (DDoS) attacks
- click fraud
- phishing & pharming attacks
- key logging & data/identity theft
- distributing other malware e.g. spyware
- anonymized terrorist & criminal communication

Other than spam and DDoS these other attacks can look a lot like normal traffic.

13.5 Why Traditional Security Measures Fail

traditional antivirus: Traditional signature-based antivirus systems are not effective, because bot codes are typically packed and they can use rootkit to hide. They also use frequent updates to defeat antivirus tools.

traditional IDS/IPS: The traditional IDS and IPS are not effective because they typically look at the specific aspect of an attack e.g. a specific exploit method, whereas botnet typically performs multiple kinds of activities because they are for long-term use. As a result although we can detect that a host has been compromised by an exploit we do not know whether it belongs to a botnet because we would need to analyze its command-and-control traffic and daily malicious activities.

honeypot: honeypots and honeynets are also not effective, because

- since they only passively waiting for incoming connections, they have to be lucky to capture botnet activities
- sophisticated bot malware can detect a honeypot due to the lack of realistic user activities
- since a honeypot is a single host, it cannot detect a network of bots

13.6 Botnet Detection

The challenges in botnet detection include

- bots try to hide themselves
- bots are also involved in multiple activities over a period of time
- bot malware can also get updates frequently
- botnets can have many different command-and-control methods, in fact a bot can be programmed to select one of several C&C methods at run time

To detect botnet we need to first focus on the characteristics that botnets are different from normal traffic.

- a bot is not a human, i.e. the activities by bots may look different from the activities by human
- the fact that botnet is a network means that the bots are connected and their activities are somehow coordinated

We can also distinguish botnets from other traditional attacks.

- botnets are for profits, and most likely are going to use compromised computers as resources
- botnets are for long-term use, and therefore there will be frequent updates to the bot malware
- there must be coordination among the bots to form a botnet

To detect botnets in the enterprise network we can deploy a botnet detection system at a gateway or router. This is how we deploy firewall and IDS. There are several detection approaches

vertical correlation: look for correlated events across a time horizon even if a bot has multiple activities in its life cycle

horizontal correlation: look for similar or coordinated behaviors across multiple bots

cause-and-effect correlation: inject traffic to play with the bot to confirm that the traffic is generated by bot versus human

In this lesson we are going to discuss two systems, one is BotHunter and the other is BotMiner.

13.7 BotHunter

BotHunter is a system that performs vertical correlation which is also called dialog correlation. That is BotHunter correlates multiple events that belong to the life cycle of a bot. The main idea behind BotHunter is to analyze network traffic to detect patterns that suggest any of these activities belonging to the botnet life cycle. These observations do not have to follow a strict order, but they do have to appear within the same period of time.
BotHunter uses a table to keep track of the evidence that it collects for each host. BotHunter keeps track of the specific activities that belong to each steps of the botnet life cycle, which are valid as long as a timer has not expired. The intuition is that within a period of time before the timer expired, if we see multiple evidences belonging to the botnet life cycle then we can determine that this host is a bot. We give more weight to evidences which suggest that an internal machine has been compromised and it is participating in botnet activities such as egg downloading, outbound scanning, and outbound spamming.

13.8 BotHunter Architecture

BotHunter has a number of detection engines, each of which is resposible for detecting certain activities of the botnet life cycle. The correlator correlates evidences of these activities and makes detection that an internal machine has become a bot and produces a bot infection profile.
The first of BotHunter detection engines is Statistical Scan Anomaly Detection Engine or SCADE. It is for scan detection. Recall that in botnet life cycle inbound scan is a first event. SCADE uses different ways for different inbound scan connections, in particular it gives a higher weight to vulnerable ports i.e./ failed connection to vulnerable port = high weight vs. failed connection to other port = low weight. SCADE has bounded memory usage to the number of inside hosts thus is less vulnerable to DoS attacks. SCADE also detects outbound scan. It looks at (1) the rate of outbound connections (2) scan failed connection rate (3) scan target entropy—the distribution of the destinations of these outbound connections (low revisit rate implies bot search). Collectively this can suggest outbound scan.
Another BotHunter detection engine is SLADE. It can detect anomalies in network payloads efficiently. The main idea is that we can establish the normal profile of a network service by looking at the n-gram byte distribution of the traffic payload of this network service. That is an attack such as an exploit or egg download will cause deviation from this normal profile, because the n-gram byte distribution of the attack traffic will be different from the normal traffic.
BotHunter also includes a signature engine to detect known exploits and known patterns of command control. The signature rules come from multiple open sources.

13.9 BotMiner

Botnets can have different infection life cycles and they can change the protocols and structures of their command-and-control. Our goal is to have a Botnet detection system that is independent of the command-and-control protocol and structure. To achieve this we need to focus on the intrinsic properties of botnet. In particular bots are for long-term use, and bots within the same botnet have similar or coordinated communication and activities. Therefore we need to perform both vertical and horizontal correlations.
A botnet is a coordinated group of malware instances, and they are controlled via command-and-control channels. Therefore the architecture of BotMiner consists of

A-plane monitor: for monitoring malicious activities

C-plane monitor: for monitoring command-and-control traffic

On both planes we perform clustering to detect groups that are in correlated or similar ways. Then we use cross-plane correlation to detect a group of machines that perform similarly in both command-and-control activities and malicious activities, and these are parts of the same botnet.
In C-plane clustering a flow record for connection between a local host and a remote service is defined by ⟨protocol, source IP, destination IP, destination port, time, number of bytes⟩. All flow records go through a number of steps that include filtering, white listing, aggregation, feature extraction, and clustering. BotMiner performs clustering in two steps—in step one we group C-flow records into course-grained clusters, in step two within each of these course-grained clusters we further position them into finer-grained groups. The main idea behind multi-step clustering is that we can use a small subset of features to perform course-grain clustering. Because the number of features that we use is small this step is very efficient. Then within each course-grained cluster we can afford to use the full feature space to perform fine-grained clustering.
In A-plane clustering we first cluster based on activity type e.g. scan, spam, binary downloading or exploit. Within each activity we use the traffic features to perform further clustering. For example for

scan we can use destination subnets and ports while for spam we can use a spam template. The main idea of A-plane clustering is to capture similar activity patterns among the hosts.
In cross-plane correlation we are looking for the intersection between A-plane and C-plane clusters. Intuitively hosts in the intersections have similar malicious activities and similar C&C patterns. In particular if two machines appear in the same activity clusters and in at least one common C-cluster, that means they should be clustered together because they are in the same botnet.

13.10 Botnet Detection Quiz

The following behaviors are indicative of botnets

- linking to an established C&C server
- generating Internet Relay Chat (IRC) traffic using a specific range of ports
- generating SMTP emails/traffic
- reducing workstation performance/Internet access to the level that is noticeable to users

Generating DNS requests by itself is not indicative of Botnet activities. However if multiple machines look up the same domains at the same time and the domain is not on a whitelist, then it is suspicious.

13.11 BotMiner Limitations Quiz

Botnets can do the following to evade C-plane clustering

- manipulate C&C communication patterns
- introduce noise (in the form of random packets) to reduce similarity between C&C flows

Botnets can vary activities to evade A-plane clustering, e.g.

- perform slow spamming
- use undetectable activities (spam spent with Gmail, download .exe from HTTPS server)

13.12 Botnet Detection on the Internet

A botnet must use Internet protocols and services to maintain a network infrastructure, e.g.

lookup services: to find C&C servers or peers

hosting services: to store and distribute/exchange attack data and malware download

transport services: to route or hide its attack traffic

Therefore by identifying the abnormal use of internet services, we can detect botnet activities on the Internet. In this lesson let us focus on DNS, and the reason is that most bots use DNS to locate command-and-control and hosting services.

13.13 Botnet Use of Dynamic DNS Services 1

Many botnets use DNS for command control. A key advantage of DNS is that it is used whenever a machine on the Internet needs to talk to another, that is DNS is always allowed in a network and using DNS for command control will not stand out easily. The bot malware has instructions to connect to its command-and-control server. But to connect to the command-and-control server it will perform a DNS lookup first. With the IP address the bot can connect to the command-and-control server and that is how the bot becomes part of a botnet. The DNS service providers preferred by botnets are dynamic DNS providers, because they allow frequent changes of the mapping between DNS domain names and IP addresses. That is the bot master can change to use another machine on the Internet for command-and-control, and all he needs to do is log into his dynamic DNS provider and make changes. If we can detect that a domain is used for botnet command-and-control, then we can detect which machine connects to this domain and this machine is a bot. On the other hand it turns out that the way the bots look up the domain is different from how machines look up Internet domains—a botnet C&C is looked up by hundreds of thousands of machines across the Internet which is however so-called unknown according to Google search and that is an anomaly. We can use anomaly detection at the dynamic DNS service provider by examining queries in DNS domains to identify botnet C&C domains. Once we identify that a domain is used for botnet C&C we have the following options:

- for the provider to disable a domain by responding to a query with "no such domain" reply
- for the provider to set a mapping of a domain to a single address, so that instead of connecting to the botnet command-and-control server the bots are now connected to a sinkhole, the sinkhole, in addition to disabling the botnet, allows security researchers to learn where the bots are by looking at the origins of the connections to the sinkhole

13.14 Botnet Use of Dynamic DNS Services 2

To detect that a domain is used for C&C there are a number of heuristics based on observations:

- Domain name purchases leave traceable financial information. To limit such traceable information a bot master may do the so-called package deal, where for one second-level domain he is going to use it for multiple three-level domains for botnets, so that a single financial transaction can support multiple botnets. Hence we can cluster the three-level domains under the same second-level domain that looks similar in their names or they resolved to similar subnets of IPs because they are likely to be botnets that are related. Summing up the lookups to all of these domains within a cluster and comparing it to the lookup patterns of legitimate domains will reveal that it has larger lookup volumes—recall that domains are in dynamic DNS providers tend to be small and medium sized businesses, therefore legitimate domains tend not to have a very large lookup volume.
- Bots must maintain communication with the command-and-control server. Since they have no control over when the host machines connect to the Internet, they must take their first opportunity to look up their command-and-control service as soon as their host machines are connected to the Internet. The consequence is that there is an exponential arrival of DNS requests from the bots, whereas legitimate DNS lookups by normal user activities is a lot smoother.

There are other detection heuristics, e.g.

source IP dispersion: the lookups are from all over the Internet

resolved IP dispersion: a C&C server is resolved to many different IP addresses across the Internet

resolved IP rate of change: resolved IP address changes frequently

Any of these observations alone cannot effectively detect a botnet command-and-control domain. Thus we combine them in a detection system.

13.15 Botnet Use of Dynamic DNS Services 3

To detect botnets in large networks such an Internet Service Provider (ISP), we can analyze the Internet traffic from the internal host to the recursive DNS server. In particular we can detect any abnormal growth of the popularity of a domain name. This can suggest that this domain name is used for botnet command-and-control. Because a botnet will grow as more machines become infected and become bots, therefore the growth pattern of a botnet corresponds to the propagation pattern of an infection. According to studies

exploit-based infection: grows exponentially

email-based infection: grows exponentially or linearly

drive-by-download infection: growth is likely to be sublinear

In a large ISP there are many domain names being looked up every day. But we can focus on a few anomalous domain names. Since many regularly spelled easily sounding domain names have been taken up by businesses and individuals already, botnets are forced to use very random looking domain names. Then we need to analyze the growth pattern of these suspicious domain names. In particular we look for the exponential or linear growth of their popularities. For example we can

1. Train a Bloom filter, which is a very efficient representation of set, to record domain names being looked up in an N-day time window. We also use Markov model to model all the domain name strings. On the $N + 1$ day if we observe a new domain that is not in the Bloom filter and does not fit the Markov model, we consider it suspicious.

2. Treat the sequence of lookups to each new and suspicous domain as a time series. Apply linear and exponential regression to analyze the growth of the number of lookups. If the growth is linear or exponential then we know that this domain is used for botnets.

The latest threats are more targeted and more advanced. For example they use custom-built malware on zero-day exploits, their activities are low-and-slow, and they move within network and covering their tracks. Even the existing botnet detection systems are not effective against these targeted advanced threats. To counteract these targeted and advanced threats we need multifaceted monitoring and analysis. That is we need malware analysis, host-based monitoring, forensics, recovery, network monitoring, Internet monitoring, threat analysis, and attribution.

13.16 APT Quiz

We need the following to identify the source (perpetrator) of an advanced persistent threat (APT) attack:

- source IP address of TCP-based attack packets
- coding style of malware
- inclusion of special libraries with know authors
- motives of the attack
- language encoding

14 Lesson 14: Internet Scale Threat Analysis

14.1 Attacker Intelligence Quiz

Attackers have three phases of intelligence gathering:

footprinting: the attacker gathers information about target, the kind of information gathered is DNS, email servers, and the IP address range

scanning: the attacker uses the internet to obtain information on specific IP addresses, the kind of information gathered is operating system, services, and architecture of the target system.

enumeration: the attacker gathers information on network user and group names, routing tables and simple network management protocol

14.2 Internet-Wide Security Scanning

There are a lot of benefits from internet wide security scanning such as discovering new vulnerabilities and tracking adoption of a defensive mechanism. On the other hand internet-dwide security scanning means that we have to probe the entire address space. With existing tools this is very difficult and slow task. However some of the security studies does require the scanning of the entire address space. For example if we can obtain the cryptographic keys in the entire address space, then we can understand the ecosystem of public key infrastructure and its vulnerabilities.

14.3 Internet-Wide Network Studies

There have been quite a few influential internet-wide network studies

- University of Michigan (2012) found vulnerabilities in more than 5% of HTTPS hosts and 10% of SSH hosts, which took 25 hours across 25 Amazon EC2 instances (625 CPU-hours)
- Electronic Frontier Foundation SSL observatory (2010) took a glimpse at the public key certificates ecosystem, which took 3 months on 3 Linux desktop machines (6500 CPU-hours)
- Census and Survey of the Visible Internet (2008) which took 3 months to complete ICMP census (2200 CPU-hours)

These scans require massive parallelization and typically they also require an extended period of time.

14.4 ZMap

Given that existing tools are so slow, researchers at the University of Michigan wrote ZMap, which is a fully-functional open source network scanner that can scan a single port on the entire IPv4 address space. On a gigabit network it can cover 98% of the IPv4 address space at 97% of the speed of gigabit Ethernet from a single machine in under 45 minutes.

14.5 ZMap Architecture

While there are many excellent multi-purpose network scanners, they were never intended or optimized for scanning the entire Internet. ZMap, on the other hand, was designed from the ground up for the specific purpose of completely scanning the whole Internet. Let's compare existing network scanners with ZMap.

existing network scanners: need quite immense amount of resources to keep track of state of the entire Internet, reduce state by scanning in batches which leads to time loss due to blocking and result loss due to timeouts

ZMap: eliminates local per-connection state to allow fully asynchronous components and no blocking except for network

existing network scanners: spend a lot of resources keeping track of which host is being scanned and which host have responded for retransmission, albeit most hosts will not respond

ZMap: uses shotgun scanning approach by always sending n probes per host and keeps minimum state information

existing network scanners: attempting to be more polite by slowing the scans and avoiding flooding through timing, resulting in time loss to waiting

ZMap: distributes the scan across the Internet to scan as fast as network allows without impacting the destination networks

existing network scanners: utilize existing OS network stack, not optimized for immense number of connections

ZMap: bypasses the inefficient network stack in the operating system and generate outgoing packets directly

14.6 Address Probes

If we simply poll network actresses in numerical order we will risk overloading the destination networks. We also don't want to have to track what host we have scanned or need to scan. So how do we randomly scan addresses without excessive state? We do this by selecting addresses according to a cyclic group. Specifically we iterate over a multiplicative group of integers modulo a prime number that is slightly larger than 2^{32}. As an example consider a cyclic group generated by primitive root 5. We start randomly with 2. The iteration is carried out by multiplying current number with 5 modulo 7 to produce the next number, i.e. $2 \to 3 \to 1 \to 5 \to 4 \to 6 \to 2$. In this way all numbers will be iterated.
Following the above example, we treat the address space as a multiplicative group and each scan is essentially a random permutation of the address space. We first decide on a primitive root or generator of the multiplicative group and then choose a random starting address. With this approach we can guarantee that all addresses in the address space can be iterated. More importantly there is very little state information we need to keep track of—for each scan we only need to keep track of the primitive root or generator and the current address.

14.7 TCP IP Quiz

TCP is used to break data into packets, then IP is used to move the packets from router to router, after that TCP is used again to reassemble the packets into the original data.

14.8 Validating Responses

To validate the scan responses we can encode secrets into the fields of the probe packets similar to the SYN cookies, by encoding a scan invocation and host-specific secret into mutable fields that will have a recognizable effect on the probe responses. Specifically for each scan host ZMap computes a hash of the destination address keyed by the scan's specific secret. This MAC value or Message Authentication

Code is then spread across any appropriate and available fields. In the case of using a TCP SYN packet for scanning we encode the secret into the source port and sequence number, because we know that the destination host will have to include the port and sequence number to send a response. As a result in response packet the receiver port is the sender port in the scanning packet and the acknowledgment number is the sequence number of the scanning packet.

14.9 Packet Transmission and Receipt

Here are the high level workflows of ZMap. To send scanning packets ZMap will be configured of a random permutation of the address space. Then the probing packets will be sent to these addresses according to the random permutation. Since the majority of fields in a probe packet never change ZMap performs all network operations at the Ethernet layer using a raw socket in order to cache values. This eliminates time lost to kernel operations such as route lookups, and allows ZMap to send probing packets at a very high speed and validate the responses by checking the MAC value, analyzing the probing results etc. This configuration allows ZMap to send probes at approximately 1.4 million packets per second on a gigabit network.
ZMap is an extensible framework allowing any type of single packet scan such as a TCP SYN scan, ICMP echo requests, or application-specific UDP scan. The ZMap framework abstracts out details such as configuration, timing, addressing, and validation.

14.10 Scan Rate

The first major question is whether ZMap or our network can handle scanning at gigabit speed. To answer this question researchers at the University of Michigan performed 10 trials scanning 1% of the IPv4 address space at different rates, ranging from 1000 packets per second to the maximum 1.4 million packets per second limited by the NIC configuration. The result shows that there is no correlation between scanning rate and the number of hosts that respond, which implies that slower scanning rates do not reveal additional hosts.

14.11 Coverage

Another question is, is one probe packet sufficient or is sending multiple packets beneficial. To estimate ZMap coverage multiples in packets were sent to 1% samples of the IPv4 address space, and the number of responses exhibits a plateau after 8 SYN packets (an eventual plateau is expected as we keep on increasing the number of probing packets sent). Analyzing these results we can estimate that with a single packet the coverage is about 97.9% and with three packets the coverage is 99.4%.

14.12 Comparison with Nmap

Many previous projects have used the popular Nmap network scanner to perform scans thus we should compare ZMap with Nmap. The researchers performed several experiments focusing on the total time spent for scan and scan coverage. In these experiments one million addresses were scanned and Nmap used the most aggressive scan template called insane with a minimal scan rate of 10000 packets per second. The results show that ZMap is capable of scanning the IPv4 address space more than 1300 times faster than Nmap, and has higher coverage than Nmap even when Nmap sends multiple probes. ZMap have higher coverage than Nmap because ZMap does not timeout hosts but Nmap does. In fact if Nmap sends one packet it times out after 250 milliseconds; if it uses two packets it times out after 500 milliseconds, and some responses arrive after Nmap has timed out. In summary stateless scanning increases both the performance and the coverage of ZMap.

14.13 Entropy Quiz

With regards to computing **entropy** is the randomness for use in cryptography or other applications that require random data. The two sources of entropy are hardware sources and randomness generators. A lack of entropy will have a negative impact on performance and security.

14.14 Cryptographic Keys

As one internet-wide security study using ZMap the researchers scanned the HTTPS servers and the SSH servers to collect their public keys, and found out that 5.6% of TLS hosts and 9.6% of SSH hosts share keys in a vulnerable manner, vulnerable in the sense that they simply use the default keys without creating their own keys, and there is an apparent entropy problem that caused them to have the same keys.
The consequence of not having sufficient entropy is as follows. As a quick review in RSA the public key n is the product of two large random prime numbers p and q, $n = p \cdot q$. The security of RSA is based on the fact that factoring n back to p and q is very inefficient. On the other hand if two public keys n_1 and n_2 share the same prime number say p, $n_1 = p \cdot q_1$ and $n_2 = p \cdot q_2$, then it is trivial to compute p because p is the greatest common divisor of n_1 and n_2, and there is a very efficient algorithm due to D.J.Bernstein to compute the greatest common divisor $p = \text{GCD}(n_1, n_2)$. With this efficient approach the researchers were able to find 2134 distinct primes, and compute the RSA private keys for 64,081 (0.50%) of TLS hosts and 10,728 (1.03%) of SSH hosts. The majority of these compromised keys are generated by headless or embedded network devices from > 40 manufacturers.

14.15 Embedded Systems

The reason why these embedded systems or network devices generate broken keys is two-fold. First they all run Linux and use Linux's /dev/urandom to generate cryptographic keys. Linux maintains several entropy pools. As entropy is gathered it is stored in the input pool and is eventually mixed into the non-blocking pool which feeds /dev/urandom. However many of these sources of entropy are not present on an embedded system. For example the system may not have keyboard/mouse and many of them don't have spindle-based disks (data access timing). In fact most of them don't even support real-time clocks (time of boot). Therefore with all of these random sources removed we end up with a deterministic source of randomness.
The second reason is that, entropy isn't mixed from the input pool to the non-blocking pool until there are more than 192 bits of entropy that have been collected. Therefore even if a small amount of entropy has been collected on a device, none of it is available until the pool reaches this threshold, which would not happen until fairly late in the boot process. This is the so-called **Boot-Time Entropy Hole** problem, which means that /dev/urandom may be predictable for a period after boot. Unfortunately cryptographic keys may be based on this predictable /dev/urandom.

14.16 Certificate Authorization

Another internet-wide security study using ZMap is on the certificates issued by certificate authorities. The problem is that there are many certificate authorities and they can sign for any website. In fact we don't even know all the certificate authorities until we see them.

14.17 Certificate Chains

Browsers such as Firefox decide who they trust. These certificates are stored as part of the browser or the operating system. Browsers typically support a few hundred root certificates. These root certificate

authorities then agree to sign other certificates. This can go on involving multiple intermediates and we end up having a certificate chain with the top certificate being a self-signed certificate. This entire chain is presented by the browser to the client, and the client can verify the signatures by traversing the certificate chain up to the browser trusted certificate which is self-signed.
The researchers at the University of Michigan performed > 200 scans in one year to track the use of certificates. They observed 3700 browser trusted certificates, and collected 42 million unique certificates from 109 million unique hosts. They also discovered two cases of misused certificates. In one case a signing certificate was accidentally issued to a Turkish transit provider. In another case 1300 certificates were issued by the Korean government.

14.18 Identifying Certificate Authorities

The researchers found 1800 signed certificates belonging to 683 organizations. More than 80% of the organizations aren't commercial certificate authorities, and more than half of the certificates were provided by the German National Research and Education Network (DFN). The most worrisome finding is that all major browser roots are selling intermediates to third parties without any constraints. 90% of the certificates are signed by 5 organizations which are descendants of 4 roots and signed by 40 intermediates, and there are only a few big players—Symantec, GoDaddy, and Comodo control 75% of the market through acquisitions. Another worrisome fact is that 26% of the trusted sites are signed by only a single intermediate certificate.

14.19 CA Risks

there are several worrisome observations made by the scan:

- CAs are ignoring foundational principles such as defense in depth and the principle of least privilege
- CAs are offering services that put the ecosystem as a whole at risk
- CAs are using some weak cryptographic keys
- CAs deploying HTTPS remains difficult

ignoring foundational principles: For defense in depth there are several technical practices already in place for limiting the scope of a signing certificate, including setting name or path length constraints and distributing leaf certificates among a large number of intermediate certificates. However all but 7 of the 1800 CA certificates do not utilize these restrictions. As an example local domain names (e.g. localhost, mail, exchange) are not fully qualified and intended resource is ambiguous and there is no identifiable owner. Nonetheless these local domain names appear on almost 5% of the certificates.

weak cryptographic keys: 90% of the certificates use a 2048- or 4096-bit RSA key, but 50% of the certificates are rooted in a 1024-bit key. More importantly more than 70% of these will expire after 2016 and many still use signs using MD5.

14.20 HTTPS Adoption

Here is an example of how do we use internet-wide scan to check adoption of a technology. From June 2012 to May 2013 the researchers found out that the number of HTTPS servers only increased by 10%, and there is a 23% increase in the number of Alexa top 1 million sites using the HTTPS and an 11% increase in the raw number of browser trusted certificates.

14.21 ZMap Open Source

The researchers at the University of Michigan have released ZMap as an open source project (`https://zmap.io`). There is also a repository of data from previous scans (`https://scans.io`).

15 Lesson 15: Domain and Network Reputation

15.1 DNSBL Quiz

DNSBL stands for DNS black list. The various DNSBL levels are defined as follows:

white: complete trust in this IP address

gray: this IP address is not directly involved in spamming but is associated with spam-like behaviors

yellow: this IP address is known to produce spam and non-spam e-mail

black: no trust in this IP address

NoBL: this IP address does not send spam and should not be blacklisted but it is not fully trustworthy

15.2 Motivation for Reputation

Traditionally when an IP address is discovered to have been used to send spam, it is added to a blacklist. This is called a **static blacklist**. If an IP address is in a blacklist then emails coming from this IP address can be blocked. This is a great idea except that attackers also know about the blacklist thus can circumvent the blacklist by using new IP addresses. In addition static blacklist model follows the philosophy of being innocent until proven guilty. We need to change this model to be in line with our philosophy about network security, that is everyone is suspect until proven innocent.

15.3 New Blacklist Model Criteria

motivation: need a dynamic reputation system that outputs the reputation score or the trustworthiness of a domain

intuition: legitimate uses of domains/sites are different from malicious uses, and such differences can be observed in DNS query traffic, e.g. the patterns of requests and the reputation of the requester, the resolved IPs, and the network providers for these domains

approach: analyze DNS traffic to extract temporal and statistical features, and then apply machine learning algorithms to learn models that can provide the dynamic score of a domain

15.4 DNS Quiz

botnet: typically have a set of domains at his disposal thus each domain is only used for short period of time

spyware: used to steal information thus needs a site to upload information which is typically registered anonymously

adware: uses domains that are not associated with legitimate businesses thus are disposable

15.5 Notos

Notos is a system that dynamically assigns reputation score to a domain name based on the features that capture the characteristics of resource provisioning, usages, and management of DNS domains. Given examples of legitimate and malicious domains Notos uses machine learning algorithm to compute a scoring function based on these features. Our study shows that Notos can correctly classify new domains with a low FP (0.3846%) and a high TP (96.8%). Hence Notos can detect a fraudulent or malicious domain weeks or days before it is widely used and appears on static blacklists.
Notos can be applied to a large network e.g. an ISP, whereas another system called Kopis has internet-wide visibility because it performs monitoring at the upper levels of the DNS hierarchy. That is Kopis uses DNS traffics at authoritative DNS (AuthNS) or top-level domain (TLD) DNS servers. Similar to Notos, Kopis also extracts a set of statistical and temporal features from the traffic and uses machine learning algorithm to train a scoring function. Similar to Notos, Kopis also has a low FP (0.3%) and a high TP (98.4%), and can detect a fraudulent or malicious domain weeks before it is widely used and appears on static blacklists. Since Kopis has the internet-wide visibility, it can detect that a domain is being used by malware within one country months or weeks before the malware begin to spread to other countries (e.g. a DDoS botnet rising in networks within China).

15.6 Kopis vs. Notos

Notos can be deployed at a large local network such as an ISP, in other words Notos uses traffic to recursive DNS servers, whereas Kopis is deployed at the upper level of the DNS hierarchy in particular the AuthNS servers or the TLD servers. In other words Notos has local views whereas Kopis can have a global internet view.

15.7 Malicious Domain Names Quiz

The types of characters a malicious domain name detection program should look for in a domain name include

number of characters: malicious domain names tend to be long

number of hyphens: malicious domain names tend to have a lot more hyphens

number of digits: malicious domain names tend to have a lot more digits

number of digits: malicious domain names tend to have a lot more numbers

15.8 Notation and Terminology

resource record (RR): a tuple of domain name and its reserved IP address, e.g. (www.exam.com, 192.0.32.10)

2nd and 3rd level domains (2LD, 3LD): e.g. for domain www.example.com, 2LD is example.com and 3LD is www.example.com

related historic IP (RHIP): all the routable IPs historically mapped with this domain name and any other domain name within the same 2LD and 3LD

related historic domains (RHDN): all the fully qualified domain names (FQDN) that historically had been linked with this IP in the RR, and all the IPs within its corresponding CIDR and AS

authoritative domain name tuple (ADNT): the requester e.g. the requester DNS (RDNS), the domain name and the RDATA that includes all the information about this domain

Notos uses passive DNS (pDNS) data collected at a recursive DNS server e.g. your ISP. pDNS data collection is the harvesting of successful DNS resolutions that can be observed in a given network. In our study we use data from multiple ISPs and data repositories. For Kopis we use data from two large authoritative DNS servers and the Canadian top-level domain server.

15.9 Local Notos

Given a resource record Notos uses

pDNS data: to extract network-based features which are based on related historic IPs

pDNS query: to extracts zone-based features which are based on related historic domains

blacklist and honeypot data: to extract evidence-based features which are evidence associated with these IPs and domains

These features are then combined and forwarded to the reputation engine which computes a reputation score for this domain.

15.10 DNS Database Quiz

Examples of the above three categories of information extractable from the pDNS database are

network-based features: the total number of IPs historically associated with the domain, the diversity of their geographical locations, the number of distinct autonomous systems in which they reside

zone-based features: the average length of domain names, the occurrence frequency of different characters

evidence-based features: the number of distinct malware samples that are connected to any of the IPs

15.11 Notos Statistical Features

network-based features: extracted from the set of RHIPs, including the total number of IPs historically associated with the domain, the diversity of their geographical locations, the number of distinct autonomous systems in which they reside

zone-based features: extracted from the set of RHDNs, including the average length of domain names in RHDNs, the number of distinct TLDs, the occurrence frequency of different characters

evidence-based features: including the number of distinct malware samples that contacted the domain, and the same for any of its resolved IPs

The result of clustering 250,000 domains using the Notos features do not overlap, which implies that the Notos features are very good at separating domains of different types into different clusters.

15.12 Notos Reputation Function

Given a set of domains known to be legitimate or malicious Notos extracts features from each of these domains and gives label 1 to malicious domains and label 0 to legitimate domains. Then given this training data set Notos can use a machine learning algorithm to learn a function that given the Notos feature vector of a domain it will output a label 0 or 1. The reputation score of this domain is simply the confidence level of this label i.e. the probability that this domain is malicious. This function is very accurate, that is it has a very high TP rate and a very low FP rate. Notos can detect many malicious domains days, weeks, or even months before they show up in any blacklist. This is true for all the malicious domains and also for different types of malicious domains.

15.13 Dynamic Detection Quiz

A dynamic malware-related domain detection system should

- have global visibility into DNS request and response messages
- not require data from other networks
- be able to detect malware-related domains even if there's no reputation data
- be able to detect malware domains before the infection reaches a local network

15.14 Kopis Statistical Features

Kopis is deployed at the upper level of the DNS hierarchy, therefore the main difference between Kopis and Notos lies in their features. In particular Kopis analyzes

requester diversity (RD): characterize if the machines (e.g. RDNS servers) that query a given domain name are localized or are globally distributed (based on BGP prefixes, AS numbers, country codes etc.)

requester profile (RP): distinguish requests from a small network versus a large network, and assign a higher weight to RDNS servers that serve a larger client population because a larger network would have a larger number of infected machines

resolved-IPs reputation (IPR): look at whether and to what extent the IP address space pointed to by a given domain name has been historically linked with known malicious activities or known legitimate services

15.15 Kopis Detection Performance

Similar to Notos, given examples of malicious and legitimate domains Kopis can use machine learning algorithm to learn a scoring function based on the features. We show that if we use more data meaning using a longer time window the accuracy is higher. In particular with a five-day time window we can achieve very high detection rate and very low false positive rate.

15.16 Predictability

Here, we show that Kopis can detect many malicious domains days, weeks, or even months before they show up in any blacklist.

15.17 Study of Mobile Malware Prevalence

We study mobile malware prevalence using domain reputation

motivation: most research has been focusing on analyzing malicious mobile apps, but the question remains how prevalent are the infections on mobile devices

intuition: mobile web is actually part of the regular web and therefore mobile malware will use similar command&control infrastructure

approach: analyze DNS traffic obtained from cellular network providers and identify domains looked up by mobile apps, analyze the reputation of the internet hosts pointed by these domains

15.18 Key Data and Findings

We used three months of data from a major US cellular provider and a major US non-cellular ISP. We find that at least in the US

- the number of machines that are infected with known malware remains very small (6586 out of 380,537,128 or 0.002%)
- iOS and Android devices are equally likely to look up these suspicious domains

15.19 Methodology

In our study we first identify the mobile devices and attribute each DNS query to a device. We then analyze the reputation of the resource records associated with these DNS query. We use the Notos features to analyze the hosting infrastructures of mobile domains and compute reputation scores. The difference is that we use different features. In particular for mobile domain we first find the IP address of the machine that hosts this domain, then for each of these IP addresses we extract the following features:

- the related historic non-cellular domains
- the related historic mobile domains
- any evidence of malware association with these domains
- URLs for phishing and drive-by download associated with these domains
- evidence that these domains have been blacklisted before

With these features and examples of malicious and legitimate mobile domains Notos can then use machine learning algorithm to learn a scoring function.
When a machine on Internet has bad reputation we call this machine tainted. Some mobile domains are hosted on these tainted machines. Our study shows that the iOS and Android devices are equally likely to connect to these tainted hosts (8.8% vs. 8.2%). We also measure the number of devices that looked up domains associated with known malware families. We found out that the number of devices with known malware infection is small.

15.20 Botnet Takedown Quiz

Typically it is not possible to enumerate all bots in a botnet. A proven method to stop botnets require isolating the C&C domain from the botnet, because if we take down the command&control infrastructure the botnet will cease to function. As such with regard to takedowns P2P-based networks are much harder than centralized C&C networks.

15.21 Botnet Takedowns

Currently botnet takedowns are ad hoc, performed without oversight, and sometimes are not successful. The high-level idea of our study is that we need to have as complete knowledge of the botnet infrastructure as possible. To expand our knowledge of the botnet infrastructure we use

network side: passive DNS analysis

malware side: malware backup infrastructure analysis

In our system called RZA we start with a set of seed domains which are known to be associated with botnet infrastructure. Then we use passive DNS data to analyze domains related to these seed domains including the reputation and malware samples associated with these domains. For the malware samples we perform malware analysis (interrogation) to find out more domains. That is how our system now has a much more complete knowledge of the botnet infrastructure. Based on this knowledge we can perform analysis of previous takedowns, and recommend takedown strategies of a current botnet.
To illustrate how we interrogate a malware, suppose foo.bar.com is a domain that malware would use by default. The malware is going to perform a lookup of this domain and then connect with its C&C server. To force the malware to tell us more domains that it may use, we can intercept the DNS query and respond with no such domain. Then if there is a backup domain the malware is going to look it up. Hense by analyzing malware DNS traffic we can find out more domains that a malware might use.

15.22 RZA Malware Interrogation

We run malware in a virtual machine and have a gateway between the malware and internet. Therefore we can control how the malware can connect to its infrastructure. For example we can play with DNS no such domain, or TCP reset to force the malware to exhibit backup behaviors, which can include using additional domains or switching to a peer-to-peer. We observe that a malware typically would

- retry hardcoded IPs/domains
- when these domains are not available, try the next finite set of IPs/domains as a backup
- when these domains are not available, switch to peer-to-peer or a domain generation algorithm (DGA) to try an increasing number of randomly generated IPs/domains

The intuition behind our approach is that malware will use an increasing number of IPs/domains when the infrastructure is unavailable. The easiest manipulation is via DNS and TCP and we can easily add more protocols.

15.23 RZA Takedown

An ideal takedown procedure is as follows:

1. start with a set of seed domains which are known to be associated with botnet infrastructure
2. enumerate the infrastructure using passive DNS
3. get the malware associated with the domains and interrogate a malware
    - if the malware tells us additional domains, then we loop back to step 2
    - if there is no additional domain, then we take the domains that we know so far and revoke all of them, that is how we take down the botnet

- if we discover that it is using dynamically-generated domains (DGA), then we need to first reverse engineer DGA and then tell the TLD operators and DNS registrars to stop hosting or registering these new domains, we will also revoke the set of known C&C domains
- if the malware exhibits behavior of using peer-to-peer network, then we need to take down the peer-to-peer protocol by e.g. penetrating into the peers, we also need to revoke the set of known C&C domains

15.24 RZA Studies

As postmortem studies we analyzed Kelihos, Zues, and 3322.org/Nitol takedowns.

Kelihos: lookups to the seed domains, the interrogated domains, and malware domains stop immediately at takedown or soon after, implying that this takedown is very effective

Zues: although there are multiple groups including Microsoft and several research groups taking place in takedowns their efforts were not coordinated, more importantly none of them had complete knowledge of the botnet infrastructure, therefore even after the takedown effort the botnet continue to connect to its infrastructure

3322.org/Nitol: takedown was accomplished by transferring the entire 3322.org name server authority to Microsoft and the malicious domains resolved to a set of known simple IP addresses, that is domains were sunk on the day of takedown and were limited to the 3322.org domain names, unfortunately this only accounts for a fraction of the domains associated with the malware, therefore lookups to this malware associated domains remain frequent

15.25 RSA Takedown Study

We also analyzed how to take down the 45 active botnet C&Cs. Among them we found out that 2 had DGA-based lookup mechanism, 1 had P2P-based backup mechanism, and 42 are subject to DNS-only takedown.
To summarize the drawbacks of current Takedown efforts, they are

- ad hoc, meaning that they don't have complete knowledge of the botnet infrastructure
- little oversight, meaning that different groups may step on each other
- the success rate is not high

Therefore we believe that a central authority to coordinate takedown efforts is necessary. ICANN's UDRP/URS policy can provide an exemplary framework. For example there is policy for domain name dispute resolution, and there is also a system for rapid suspension. Therefore we should study

- policy criteria for takedowns
- have the community review such criteria (more eyes = more successes)
- test the policy with the newly created top-level domains (much like with URS)

16 Lesson 16: Machine Learning for Security

16.1 Data Analysis Quiz

The types of analytics are

anomaly detection: model normal network and system behavior and identify deviations from the norm, have the ability to detect zero-day attacks

misuse detection: detect known attacks using signatures of those attacks, can detect known types of attacks without generating a lot of false positives

hybrid detection: combination of misuse and anomaly detection

16.2 Machine Learning Review

The task of machine learning is that given training examples x as input, we would learn a function f that can predict the output $y = f(x)$. Once we learn this function we can apply it to testing examples and this function can then produce the output.
Machine learning consists of the following steps:

training: given a training set of labeled examples $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ estimate the prediction function f by minimizing the prediction error on the training set

testing: apply f to a never seen before test example x and output the predicted value $y = f(x)$

Ideally data used in machine learning should reflect the real world. Therefore we use a process to draw data from the real-world application. Then that data is randomly split into training dataset and test dataset. We can then apply a machine learning algorithm to training dataset to learn the particular function that can be applied to test dataset.

16.3 Machine Learning Example

For machine learning purpose objects are represented as feature vectors of their images. Machine learning algorithms are applied to the feature vectors of training dataset to train a predictive function. This predictive function is the learned model. In testing again feature vector is extracted from the object and fed into the predictive function to produce a prediction.

16.4 Machine Learning Features

For images the useful features are: the raw pixels, the histogram, and GIST which is a representation of the scene.

16.5 Generalization

A good machine learning model should be able to generalize from the data that it was trained on to new test dataset. Generalization is the most important property of machine learning.

16.6 ML Types Quiz

The types of machine learning are

supervised: the task is to find the function or model that explains the data

unsupervised: the main task is to find patterns, structures, or knowledge in unlabeled data

semi-supervised: some of the data is labeled during acquisition

16.7 Performance Measures

error rate: fraction of false predictions

accuracy: fraction of correct predictions

precision: fraction of correct predictions among all examples predicted to be positive, can be generalized to multi-class cases

recall: fraction of correct predictions among all real positive examples, can be generalized to multi-class cases

16.8 Classification Problems

In a classification problem the training dataset contains records with attributes or features and class labels. Then the machine learning algorithm is to produce a model that would output the class label based on a set of attributes or features.

16.9 Decision Tree Learning

In decision tree learning the training data is repeatedly partitioned until all examples in each partition belong to one class. The decision tree can also be thought of as a set of rules that describe the decision logic. To build a decision tree we first find the best attribute that can serve as the root. Then we break up the training dataset into subsets based on the values of this root attribute, that is, we grow from the root into branches. Then for each subset we test the remaining attributes to see which is best at partitioning this subset. We continue this process until all examples in a subset belong to one class, or there is no more attribute left to further partition the subset in which case the default class label is the majority.
We use entropy and information gain to determine which attribute is the best at partitioning a set. The entropy of a set is the minimal number of bits needed to represent examples in this set according to their class labels. Roughly speaking the entropy of a set represents how pure the examples are in this set—if the examples of this set are evenly distributed into different classes then the entropy is the maximum; if the examples are all in a single class then the entropy is the minimum. To determine which attribute is the best at partitioning a set we compute its information gain measured in terms of entropy reduction. A high information gain value for attribute A means that A is better at separating samples in a set S according to their classification, that is S can be partitioned into purer subsets by using A. Hence we should use the attribute that has the highest information gain to partition a set.

16.10 Decision Tree Quiz

Decision tree intrusion detection models can

- supplement honeypot analysis

- supplement penetration testing
- highlight malicious traffic
- characterize known scanning activity
- detect previously unknown network anomalies

16.11 Clustering

Clustering can also be used for classification. In clustering we assign training examples into different clusters based on some distance measure, and typically we define the distance function to measure similarity. That is examples in the same cluster are more similar to each other than examples from different clusters. Typically we predetermine the number of clusters.
We start with seed clusters each with one element, that is each seed is a representative example of a cluster. Then we assign samples to these seed clusters based on distance measure. After that we find a new centroid for each cluster. A centroid is the center of a cluster according to some distance measure. Then based on the new centroids we adjust cluster membership so that the samples are assigned to the best fit cluster, where cluster membership is based on the distance to the centroids of the clusters. We continue this process of finding new centroids and adjusting cluster membership until the clusters converge. That is there is no more changes in the cluster membership. Once the clusters are finalized each cluster can be represented by its centroid. Given a test data it is assigned to a cluster if its distance to the centroid of that cluster is the shortest.

16.12 Define the ML Problem Quiz

Suppose we want to recognize the attributes of C&C communication on networks using supervised learning, we need known C&C communication data as labeled examples.

16.13 Classifiers

We should always try simple classifiers first and it is better to have smart features + simple classifiers than simple features + smart classifiers because smart features capture the domain knowledge. On the other hand if we have a lot of data or wide variety of data, we can use increasingly powerful classifiers. Intrusion detection is about classifying each network flow into normal or different attack types. Hence it is natural to apply machine learning algorithm to learn a classifier to detect intrusions. Traffic data has higher entropy because it is mixed with intrusions and normal traffic. We can build a decision tree to partition the data into subsets so that each has a pure class—normal or one of the attack types.

16.14 Audit Data Processing

We start with raw data captured from the network tab e.g. tcpdump packet data which can be summarized into connection records. Having smart features is the key to building effective classification models. This is known as the feature construction problem. In the context of intrusion detection the features of the connection records are not very useful. For example normal data can also have S0 flag meaning only the initial SYN packet is seen, which can be due to the loss of connection request or response. On the other hand if we look at the percentage of S0 flag connections to a particular host, we can see that this feature can distinguish intrusion from normal traffic. In other words this feature has high information gain. Note that this feature is not in the original feature set, which means we need to construct features that have high information gains. To construct features that are useful to

distinguish normal versus intrusions, our approach is to use temporal and statistical patterns associated with intrusions, e.g. a lot of S0 connections to the same service or host within a short period of time.

16.15 Building ID Models

We start with raw audit data. We summarize the data into connection records. Then we find frequent patterns and compare the patterns to determine the unique patterns associated with intrusions. We then construct features according to these patterns. With these features we can learn classification models. This process is iterative. In fact each step can iterate over and over to improve performance.

16.16 Mining Patterns

We use data mining algorithms to find patterns. We first find the associations among the features. We then find the patterns that describe how the associations tend to appear in a sequence. For example if you see the first two S0 connections then there is 80% chance that you will see the third one within a 2-second time window.

16.17 Basic Algorithms

The basic algorithms are the association rules and frequent episodes. These algorithms can produce many useless patterns, e.g. associations and frequent sequences involving source bytes and destination that do not involve the more important attributes such as service and flag. Therefore we modify the basic algorithms so that they only produce relevant or useful patterns. The intuition is that the relevant patterns must describe these essential features: time, source, destination, source port, service. More specifically we use some of these essential features as the so-called axis attributes or reference attributes to constrain how the patterns can be computed.

16.18 Axis Algorithms

An axis attribute is typically the most important attribute e.g. the service. An association must contain an axis attribute. Once we compute associations involving axis attributes, we can then compute sequential patterns involving these associations. We then use reference attribute to compute a sequence of related associations. These associations in a sequence all refer to the same reference subject because they are essentially a sequence of related actions, e.g. all associations referring to the same destination host. Again reference attribute can be used as a constraint to filter out patterns that are not useful to intrusion detection.

16.19 Patterns

After we compute the frequent patterns from the normal data and from the intrusion data, we can then use these patterns to construct futures to build classifiers. We call the axis and reference attributes the *anatomy* of an attack, and the *invariant* attributes are those that are independent of the axis and reference attributes. Since our patterns are frequent and sequential in nature, we apply count, percent, and average operations to add the corresponding temporal and statistical features.
To summarize we mine patterns from both the intrusion and normal datasets and compare the patterns to identify the patterns associated with only the intrusion dataset. We construct futures according to these patterns and add these features into the training dataset. We can then apply a machine learning algorithm to learn a classifier that can detect intrusions.

16.20 Dataset Selection Quiz

Some considerations when selecting a dataset for training are

- there is no perfect way of labeling data and therefore there is really no perfect IDS dataset
- select a correct baseline dataset for your network
- select a dataset that has a range of intrusion attacks

16.21 Feature Construction Example

Using service as the axis attribute and destination host as the reference attribute, we can compute a frequent pattern. Based on this pattern we can construct these following features

- count the connections to the same destination host in the past two seconds
- among these connections the percentage with the same service and the percentage that has the S0 flag

16.22 DARPA Evaluation

There are 38 attack types in the DAPRA evaluation dataset which can be classified into four categories

DoS: e.g. SYN flood

probing: gather information, e.g. port scan

remote-to-local (r2l): remote intruder illegally gaining access to local systems, e.g. guess password

user-to-root (u2r): user illegally gaining root privilege, e.g. buffer overflow

Among them 40% are in test dataset only.

17 Lesson 17: Data Poisoning & Model Evasion

17.1 Introduction

Attackers try to compromise the process of machine learning through

data poisoning: cause machine learning to produce wrong model

model evasion: learn about the decision boundary of a model and then bypass it

17.2 Machine Learning for Security: History

Machine learning for security has a long history, at least 20 years. In the early days

- Getting labeled training data was very hard because our understanding of attacks was limited, and there were very few attack examples. Creating a standard or reference dataset so that every researcher could use was even harder.
- Feature construction in the early days was mostly manual, which typically involves encoding domain knowledge and extract the features from raw data.

- Security experts were typically very skeptical about the approach of applied machine learning for security, because

    1. the perceived high rate of false positives from these models
    2. if we had to use domain knowledge to extract features, then why don't we just hand code expert rules?

17.3 Machine Learning for Security: Recent Work

Things have changed quite a bit over the last 10 years.

- there is a lot more data now available for research, because

    1. it is very easy to obtain malware samples and run these samples to obtain data
    2. there is a lot of data sharing going on
    3. organizations are more supportive of data logging e.g. allowing network traffic to be recorded (approval from the Institutional Review Board (IRB) is needed for a research institution)

- it is still hard to find standard or reference datasets, because

    1. it is hard to share normal training data because of privacy concerns
    2. some malware samples are considered sensitive, that is they are not to be shared

We have seen many papers and even products in applying ML for security. Basically everyone now realizes the need to analyze a huge amount of data. In fact the security sensors themselves also produce a huge amount of data e.g. alerts. Therefore the huge number of alerts also need to be processed. There are many benefits of automatic data analysis, for instance we can

- perform a link or correlation analysis on alerts
- extract interesting or previously unknown patterns from data
- validate if our so-called expert knowledge or intuition is correct

17.4 Machine Learning for Security: Future

For security the bar is much higher because machine learning has to work not only in the normal cases but also in cases where there are attackers around. We know from history that attackers would try to defeat machine learning-based security mechanisms.

17.5 Adversarial Machine Learning

Adversarial machine learning refers to machine learning in a setting where there are attackers around. Early research in this field dated back to early 2000s. For example researchers showed that

- by inserting normal system calls into attack sequences, an attack can evade an anomaly detection model
- by including normal contents in the attack payload, an attack can evade a network anomaly detection model
- by inserting incorrect network data, attackers can force the machine learning process to produce incorrect worm signatures

More recent research has been focusing on the conceptual framework of adversarial machine learning, and the limitation of emerging machine learning paradigms such as deep learning. There are also demonstrations of attacks on new machine learning applications such as web services. For example attackers can inject fake user actions and signals to influence such results.

17.6 Attacks on Machine Learning

Conceptually there are two types of attacks on machine learning:

exploratory attack: an attacker uses examples to find out the decision boundary of a machine learning model and then crafts his attack to evade this model, this is also called **evasion attack**

causative attack: an attacker injects malicious examples into the training data so that wrong model is produced, this is also called **data poisoning attack**

17.7 Evasion Tactics Quiz

environmental awareness: allows malware samples to detect the underlying runtime environment of the system it is trying to infect

confusing automated tools: allows malware to avoid detection by technologies such as signature-based antivirus software

timing-based evasion: used by malware to run at certain times or follow certain actions taken by the user

obfuscating internal data: uses a number of tricks to run code that cannot be detected by the analysis system

17.8 Dyre Wolf Attack

1. an employee within the targeted organization receives an email with the Upatre malware
2. upon opening the attachment the Upatre malware is installed
3. Upatre establishes communication to the attacker and downloads Dyre
4. Dyre alters the response from the bank's website, tricking the victim to call an illegitimate number
5. to overcome measures by the bank to protect against fraud, the Dyre Wolf social engineers critical information from the victim through the phone call
6. up to $1.5 million USD is quickly transferred from the victim's account to several offshore accounts
7. immediately after the theft a high volume DDoS against the victim starts in order to distract or hinder investigation

17.9 PAYL: Payload-Based Anomaly Detection System

We use a simple anomaly detection system as an example. This system is called the Payload-Based Anomaly Detection System or PAYL. It measures and models the frequency distribution of the n-grams in traffic payloads. If $n = 1$ then it is measuring the byte frequency distribution. The intuition is that each service such as web or email that is delivered to a host or an enterprise network has its own unique characteristics. For example a particular user normally gets certain types of web or email contents. On the other hand attacks embedded in network traffic such as malware in an email attachment are very different kind of contents.
We use $n = 1$ as an example to explain PAYL. In this case PAYL computes for each character its relative frequency and a standard deviation. Then a set of frequencies and standard deviations for all the characters in normal traffic data form the normal traffic profile. We can then compute the anomaly score of a packet which is essentially the sum of fraction of the deviations across all bytes in the packet

$$score(P) = \sum_i \frac{f_P(x_i) - f(x_i)}{\sigma(x_i) + \alpha}$$

where α is a smoothing factor to prevent division by zero. Thus PAYL is simple and therefore it is also very efficient in runtime. It can detect zero-day attacks and polymorphic attacks.

17.10 Polymorphic Attacks vs. Detection

Polymorphic attack has three main components

attack vector: used for exploiting vulnerability, some parts can be modified but there is always a set of invariant parts typically the starting point of execution, if the invariant parts are small and exist in normal traffic then detection can be evaded

attack body: contains the attacker's shell code and is typically transformed or encrypted

polymorphic decryptor: decrypts the shell code (since the attack body is encrypted it needs to be decrypted in order to execute) and can be transformed

Because in a polymorphic attack the transformed attack body and decryptor tend to have different byte frequencies than a normal traffic, the polymorphic attack can in theory be detected by anomaly detection system such as PAYL.

17.11 Evading Detection: Polymorphic Blending Attacks

To defeat anomaly detection system an attacker can use **polymorphic blending attack** instead of the simple polymorphic attack, in which each polymorphic instance will try to blend in with normal traffic by matching the normal profile of the legitimate traffic. For example each polymorphic instance can have its byte frequency distribution match that of the normal profile, albeit attackers must carefully choose encryption key and pad its payload to replicate the desired byte frequency.

17.12 Polymorphic Blending Attack Quiz

- the process should not result in an abnormally large attack size, because otherwise the size by itself can raise an alert
- the blending needs to be economical in time and space, because otherwise it will not only slow down the attack but also increase the chance of detection by the IDS
- attacks of this type need not collect a lot of data to learn normal statistics

17.13 Polymorphic Attack Scenario

Suppose the attack is already in a network A that has a relationship with the target network B. It is reasonable to assume that the attacker has knowledge of the IDS of network B, because the attacker can read the product white paper. In fact the attacker can send normal traffic from network A to network B and observe which traffic is accepted or rejected by the IDS, and can then estimate the false positive rate of the IDS.

17.14 Blending Steps

Here are the steps of a polymorphic blending attack:

1. compromise a host X in network A
2. observe the normal traffic going from host X to host Y in network B
3. use the traffic and the IDS modeling algorithm to produce an estimated or artificial normal profile
4. craft an attack to match the artificial normal profile using shellcode encryption and padding
5. send the attack traffic which should be able to evade the detection of the IDS

The original attack packet includes the attack vector and the attack body/attack code. In blending the attacker can encrypt the attack code using only the normal characters and add the decryptor. He can also transform the attack vector and the decryptor using as many normal characters as possible. The resulted packet may not match the normal frequency distribution, but the attacker can add more padding of normal characters so that the byte frequency distribution of the whole packet matches the artificial normal profile.

17.15 Polymorphic Blending Quiz

After blending the attack should not match the normal profile perfectly. Nonetheless attackers only need to blend the profile that approximates the real profile, because as long as the packet matches the normal profile within variance it will be accepted.

17.16 Blending Attacks Requirements

In a polymorphic blending attack the most important step is to transform the attack code into normal data, that is each byte is transformed to a byte in a legitimate traffic so that the desired byte frequency is matched. We can use a substitution table for this purpose because the attack vector and decryptor cannot be randomly transformed. We may need to add more padding of normal characters so that the entire packet matches the desired byte frequency.

17.17 Encrypting Attack Contents

Finding an optimum substitution table that only requires minimum padding is a very hard problem. We can instead use a greedy method, that is we can map the most frequent byte value in attack traffic to the most frequent byte value in normal traffic, the second most frequent byte value in attack traffic to the second most frequent byte value in normal traffic and so on until all byte values in attack traffic have been mapped.

17.18 Decryptor

The job of the decryptor is to remove the padding and reverse the substitution step. The decryptor code cannot be blended but can be transformed into equivalent instructions that contain mostly normal byte values. The substitution table can also be represented in such a way that it contains only normal byte values, e.g. the ith entry is the byte value in normal traffic that was used to substitute the byte value i in attack traffic.

17.19 Evaluation

First we verify that PAYL can detect polymorphic attacks without blending using CLET. Next we create polymorphic blending attacks and demonstrate that they can evade detection by PAYL. We also evaluate how easy it is to construct such blending attacks.

17.20 Evaluation Result

One measure on how easy it is to construct such blending attack is the number of packets required to train an artificial normal profile. Training of the artificial profile is stopped when there is no significant improvement over existing profile (measured using Manhattan distance) with two more packets. For 2-gram PAYL we need more packets for convergence than 1-gram PAYL but still no more than 100 packets. This result shows that it is quite easy to learn the artificial normal profile. Before blending the byte frequency distribution of the attack data packet looks very different from the byte frequency distribution of the normal traffic, whereas after blending the byte frequency distribution of the attack data packet matches the normal profile.

17.21 Countermeasures

PAYL is a very simple model that uses only statistical features. We should use more complex models. A more sophisticated model can

- use syntactic and semantic information such as the syntax and meanings of email or web contents
- combine multiple IDS models that use simple but different features
- introduce randomness into the IDS model, e.g. instead of using two consecutive bytes for 2-gram, model byte pairs that are ν characters apart where ν is chosen at random and fixed for a given IDS model, then combine multiple such systems

17.22 Poisoning Attack

An invasion attack is possible because the attacker is able to learn the decision boundary of a machine learning model, i.e. in general there is no protection of confidentiality of a machine learning model. In a poisoning attack the attacker injects wrong or malicious examples into the training dataset. Therefore this is an attack about data integrity.

17.23 Poisoning Attack Goals Quiz

The goals of a successful poison attack are

- create a wrong model so that the attack is undetected
- the attack continues for a while and no one knows that the data has been poisoned

- cause permanent damage to the training data so that it cannot be repaired

As a real-life example of poisoning attack, residents of a neighborhood of LA used the navigation app Waze to falsely report their streets as congested, hoping to direct traffic away from the neighborhood. Nonetheless there were enough Waze users driving through the local streets to offset the poisoned data. In this case the signal far outweighs the noise therefore the noise could not affect the outcome.

17.24 Syntactic Worm Signatures

Researchers have proposed to use machine learning algorithms to produce signatures for worms including polymorphic worms. In particular by examining the network packets related to a worm we can extract a regular expression that describes all variants of the worm, and the worm signatures are based on this regular expression. Our studies show that if the attacker can inject noise into the training data, then the machine learning algorithm cannot generate effective worm signatures.

17.25 Syntactic Worm Signature Generators

The workflow of worm signature generation is as follows

1. capture normal traffic and worm traffic
2. the network traffic is clustered and classified into normal vs. worm
3. a set of worm flows is used to generate signatures
4. the signatures is installed at a firewall or NIDS to stop traffic of worm

17.26 Traffic Based Flow Classifiers

simulated honeynet: A simulated honeynet simulates hosts with individual networking personalities. They intercept traffic sent to nonexistent hosts and use a similar system to respond to this traffic. Any flow that is sent to this honeynet is inserted into the suspicious flow pool.

double honeynet: The first layer honeynet is made of real hosts. Whenever a first layer honeypot host is infected by a worm, its outgoing traffic is redirected to the second layer simulated honeynet and the flow is inserted into the suspicious flow pool.

port-scanning detection: Traffic flows that perform port scanning are collected into the suspicious flow pool.

anomaly IDS: We can also use an anomaly detection system to capture suspicious worm flows. Any anomalous flow can be automatically put into the suspicious flow pool.

17.27 Fake Anomalous Flows

Worm propagates by sending out worm traffic. Worm can also send out fake anomalous flows which need not exploit any vulnerability but just need to appear anomalous or look like worms. With fake anonymous flows injected the flow classifier cannot detect or identify these fake anonymous flows, thus these fake anonymous flows will be included in the suspicious flow pool. As a result the machine learning algorithm cannot generate useful worm signatures, because these signatures will have too many false positives and false negatives.

simulated honeynet: since both the real worm and the fake anonymous flow are sent to the same destination at the same time, they will both be consider suspicious and therefore will both be included into the suspicious flow pool

double honeynet: the real worm will infect a first-layer honeypot whereas the fake anomalous flows will not and will be disregarded, however the real worm infection will again send a real worm flow and a fake anomalous flow, therefore both the real worm flow and the fake anomalous flow will be included in the suspicious flow pool

port-scanning detection: since the real worm flow and the fake anomalous flow both behave like scanning traffic, they will both be included in a suspicious flow pool

anomaly IDS: the fake anomalous flow will be detected as an anomaly, and therefore it will also be included to the suspicious flow pool

17.28 Case Study: Polygraph

Polygraph can use any one of the flow classification techniques that we have discussed. It can generate signatures for polymorphic worms. The system was designed to handle noise in the training dataset because no flow classifier is perfect. It was believed to be able to generate high-quality signatures even in the presence of noise. But we show that if the attacker crafts and injects fake anonymous flows then these flows will be included in the suspicious flow pool. As a result the machine learning algorithm will generate signatures that have too many false positives and false negatives.
Polygraph generates three types of signatures—conjunction, token-subsequence, and Bayes. Each worm flow includes protocol framework, worm body, and true invariants. First the strings that are common to all suspicious flows are extracted which are called *tokens*. A conjunction signature is an ordered set of tokens while a token-subsequence signature is a sequence of tokens. We will show that they are not resilient to noise in the suspicious flow pool.

17.29 Hierarchical Clustering

For clustering if the suspicious flow pool happens to include innocuous flows which is a noise, then the signature will only include the protocol framework which will result in many false positives. Such signatures are useless. Therefore before generating signatures Polygraph first performs clustering so that the worm flows will be clustered together and the innocuous flows will be in its own cluster.

17.30 Misleading Conjunction

The fake anomalous flows inserted by the attacker need to first defeat clustering so that only useless signatures will be produced. The attacker can permute the worm body and then create fake invariants. These fake invariants are set in such a way that they are even less likely to appear in innocuous flows than true invariants. This means that signatures that are based on fake invariants will be selected over signatures based on true invariants.

17.31 Misleading Hierarchical Clustering

Each cluster has a real worm flow and a corresponding crafted fake anomalous flow. This guarantees that all the signatures are based on the fake invariants.

17.32 Polygraph

The third type of Polygraph signatures, Bayes signatures, are generated as follows. First all the tokens common to at least K out of the total N suspicious flows are extracted. Then for each such token compute the probability that it appears in a suspicious flow and the probability that it appears in an innocuous flow. Then we compute the log ratio of these two probabilities.

$$\left.\begin{array}{l} P_{sf} = P(t_j|\text{suspicious flow}) \\ P_{if} = P(t_j|\text{innocuous flow}) \end{array}\right\} \implies \lambda_j = \log(P_{sf}/P_{if})$$

We can call this the score of the token. Then a Bayes signature is a set of tokens and their tokens scores. When we examine a flow we sum up the token scores according to the signature. If the total score is above a threshold then we say this is a worm flow, otherwise it is an innocuous flow. The detection threshold is computed during training to yield high true positive rate and low false positive rate.

17.33 Misleading Bayes Signatures

Consider a string ν in a normal HTTP traffic. Suppose this string shows up in 10% of the innocuous flows. Suppose the attacker injects substrings of ν into all fake anonymous flows so that roughly half of the suspicious flows have these substring tokens. The Bayes signature will then include these substring tokens and their scores. Now an innocuous flow that happens to contain this string ν is going to match all these substring tokens, and the scores will be added together (score multiplier) so that its signature score is much greater than before the substring tokens are injected. Since now the innocuous flows have very high signature scores, they will likely be classified as worm flows. That is by injecting the normal substrings into the fake anonymous flows, the attacker can force the machine learning algorithm to produce Bayes signatures that have too many false positives or false negatives.

17.34 Crafting the Noise

Putting all the above together, the attacker can permute the bytes of the worm body and inject fake invariants which are used to defeat conjunction and token-subsequence signatures. Then the attacker can select some normal string in normal traffic samples and inject substrings of this normal string into fake anomalous flows. These normal substrings are used to defeat Bayes signatures. The fake invariants are specific to each worm and its fake anomalous flow, whereas the normal substrings or the score multiplier strings are common to all fake anomalous flows.

17.35 Experimental Results

We use Apache-Knacker HTTP worm.

training dataset: suspicious flow pool = 10 worm variants, innocuous flow pool = 100,459 HTTP requests with 0.007% FP

test dataset: suspicious flow pool = 100 worm variants, innocuous flow pool = 217,164 HTTP requests with 0.0% FP

attacker's dataset: 300 candidate score multiplier strings extracted from 5000 normal flows

17.36 Results with Bayes Signatures

With noise injected by the attacker Polygraph was unable to produce signatures that have low false positive and false negative at the same time. We sent each real worm flow 20 times and compared two scenarios—in the first case one fake anonymous flow is created per one real worm flow, while in the second case two fake anonymous flows are created per one real worm flow. The result shows that if we use more fake anonymous flows then the attack success rate is higher. It also shows that if all three signatures are used to detect worms then the detection rate is higher. However even if Polygraph uses all three types of signatures, the attacker can simply use more fake anonymous flows to achieve a high chance of success.

17.37 Conclusion

Our study shows that attackers can inject noise to mislead worm signature generators. To mitigate the noise injection attack we need a precise flow classifier which remains an open problem.

17.38 Misleading Worm Signature Quiz

If we can completely control the process of generating or collecting the training data and ascertain the authenticity and integrity of the data set, then we don't need to worry about data poisoning attacks. However if the training data is obtained in an open environment such as the web, then there is always the potential of poisoning attacks, because in an open environment the attacker can inject poisonous data into the environment.

18 Lesson 18: Basics of Blockchain & BitCoin

18.1 Hash Function Quiz

Cryptographic hash functions do not have a key and are primarily used for message integrity.

18.2 Review of Hash Functions

There are several important properties of cryptographic hash functions

- they are easy to compute
- we can compute the hash of a message of any size
- for a given hash function it produces fixed length output (128–512 bits)
- they are one-way functions, that is given the hash of the message it is not easy to find the original message
- they are collision resistant, that is given m_1 it is computationally infeasible to find $m_2 \neq m_1$ such that $H(m_2) = H(m_1)$, this is called the **weak collision resistant** property, note that the **strong collision resistant** property states that it is computationally infeasible to find $m_1 \neq m_2$ such that $H(m_1) = H(m_2)$

18.3 Hash Pointers and Data Structures

A hash pointer contains not only a pointer to where the data is stored but also the hash of that data. We are going to use $H(\cdot)$ to represent a hash pointer. With hash pointer we can not only retrieve the data but also check whether the data has been modified or not. We can use hash pointers to build very useful data structures. In particular we can build **blockchain** which is a linked list of hash pointers. Suppose we have a hash pointer pointing to a block. This block contains not only data value but also the hash pointer of the previous block, and likewise the previous block contains not only the data value but also a hash pointer to the previous block. This is called a blockchain. To construct a blockchain we append a block at the end, include a hash pointer in the block, and then compute a hash and store the hash pointer. This hash pointer is the root of this blockchain.
A use case for blockchain is a tamper-evident log. That is we want to build a log data structure that stores a bunch of data and allows us to append data onto the end of the log. If an attacker changes data in this block then the hash will not match the hash pointer store in the next data block. Of course the attacker can then change the hash value stored in this hash pointer. But the result is that the hash value of this block will not match the hash value stored in the hash pointer of the next block. Therefore tampering a data item will cause the difference of the computed hash of the whole chain to be different from the hash value stored in the hash pointer to the whole chain. That is by just remembering this single hash pointer we can easily detect data tampering.

18.4 Digital Signatures

What we want from signatures is that only the owner of the private key can sign, but anyone can verify because they can retrieve the owner's public key. Further a signature is tied to a particular document and cannot be cut and pasted to another document. An API for digital signatures contains a secret signing key sk and a public verification key pk. The key pair can be created by specifying the key size. Then for a particular message we can create a signature by encrypting the message using the private signing key. Given the message and the signature any person who has the public key can verify the signature. The verification is performed by using the public key to decrypt the signature and matching the result with the message.

$$\begin{aligned}(sk, pk) &:= \text{generateKeys(keysize)}\\ sig &:= \text{sign}(sk, \text{message})\\ isValid &:= \text{verify}(pk, \text{message}, sig)\end{aligned}$$

Requirements for signatures include

- valid signatures can be verified
- no one can forge signatures without the private key which is only known to the owner

18.5 Public Keys as Identities

Public key can be used as an identity. To make a new identity create a new random key pair (sk, pk). The pk is the public name or ID you can use, and the sk is what you can use to prove your identity. There are two advantages of this approach:

identity theft protection: since only you have sk nobody can forge your identity, moreover if the public key looks very random nobody really knows who you are

decentralized identity management: anybody can create a new identity at anytime, in fact you can make as many identities as you want, there is no central point of control or coordination

These public key-based identities are called "addresses" in Bitcoin.

18.6 Cryptocurrency Quiz

- the security of cryptocurrency ledgers depends on the honesty of its miners
- most cryptocurrencies are designed to reduce production mimicking the market of precious metals
- since cryptocurrencies are pseudo-anonymous they are less susceptible to law enforcement seizure

18.7 Simple Cryptocurrencies

GoofyCoin is about the simplest cryptocurrency we can imagine. There are just two rules of GoofyCoin. The first rule is that Goofy can create new coins any time he wants, and the second rule is that the newly created coins belong to Goofy. To create a coin Goofy generates a uniqueCoinID, an ID that he never used before. Then he constructs a string, CreateCoin[uniqueCoinID]. He then computes the signature on this string using his private signing key. The string together with his signature is a coin. Another rule of GoofyCoin is that whoever owns a coin can transfer the coin to someone else. Let's say Goofy wants to transfer a coin that he created to Alice. To do this he creates a new statement that says "Pay this to Alice" where "this" is a hash pointer which references the coin that he created and wants to transfer to Alice. As we discussed earlier identities are just public keys, therefore Alice here is really the public key of Alice. Goofy also signs the statement because he owns the coin thus he has to sign any transaction that spends the coin. Once Alice owns the coin she can spend it in turn. To do this she creates a statement that says "Pay this to Bob's public key" where "this" is a hash pointer to the coin that was owned by her and of course she signs the statement. Now Bob owns the coin and anyone can verify that Bob is indeed the new owner. Because anyone can follow the chain of hash pointers to the coin's creation, and verify that at each step the rightful owner signed a statement that says "Pay this coin to a new owner".

18.8 Double Spending Attack

Let's say Alice passes a coin to Bob by sending her signed statement to Bob but didn't tell anyone else. She could create another sign statement that pays the very same coin to Chuck. To Chuck this would appear to be a perfectly valid transaction. Bob and Chuck would both have valid-looking claims to be the owner of this coin. This is called a **double spending attack**. That is Alice spent the same coin twice. Double spending attack is one of the key problems any cryptocurrency has to solve. GoofyCoin does not solve the double spending attack and therefore it is not secure. GoofyCoin's mechanism for transferring coins is very similar to Bitcoin. But because it is insecure it is not a cryptocurrency.

18.9 Scrooge Coin

To solve the double spending problem we will design another cryptocurrency which we will call Scrooge Coin. Scrooge Coin is build of Goofy Coin. The first key idea is that a designated entity called Scrooge publishes an append-only ledger that contains the history of all the transactions that have happened. This append-only property ensures that any data written to this ledger will remain forever. We can use it to defend against double-spending by requiring that all transactions to be written to the ledger before they accepted.
To implement this append-only feature Scrooge can build a blockchain that he will sign. Each block contains the IDs of the transactions, the contents of the transactions, and a hash pointer to the previous block. Scrooge signs the final hash pointer which binds all the data in the entire blockchain, and then publishes the signature along with the blockchain. In Scrooge Coin a transaction is only valid if it is in the blockchain signed by Scrooge. Scrooge makes sure that it doesn't endorse a transaction that attempts to double spend on already spent coin. The hash pointers ensure the append-only property,

because if Scrooge tries to add or remove a transaction to the history or change an existing transaction, it will affect all of the following blocks because of the hash pointers.
In a system where Scrooge signs blocks individually, you will have to keep track of every single signature Scrooge ever issued. A blockchain makes it easy for any two individuals to verify that they have observed the exactly same history of transactions signed by Scrooge.

18.10 Scrooge Transactions

For Scrooge coin each transaction has a transaction ID and there are two kinds of transactions:

CreateCoins: A CreateCoins transaction can create multiple coins, each of which has a serial number within the transaction ID, a value, and the recipient's public key. By definition a CreateCoins transaction is always valid if it is signed by Scrooge.

PayCoins: A PayCoins transaction consumes/destroys some coins and creates new coins of the same total value. The new coins may belong to different people. A PayCoins transaction is valid if

1. the consumed coins are valid
2. the consumed coins were not already consumed in some previous transaction i.e. not double spent
3. total value of the coins consumed = total value of the coins created, i.e. only Scrooge can create new value
4. the transaction is correctly signed by owners of all the consumed coins

18.11 Centralization Problem

ScroogeCoin will work in a sense that people can see which coins are valid. It prevents double spending because everyone can look into the blockchain and see that all the transactions are valid and that every coin is consumed only once. But the problem is that Scrooge has too much influence:

- he could stop endorsing transactions for some users denying their service and making their coins unspendable
- he could be greedy and refuse to publish transactions unless he gets a fee
- he could create as many new coins for himself as he wants

The problem here is centralization. Cryptocurrencies with a central authority have largely failed to take off, and the main reason is that it is difficult for people to accept a cryptocurrency with a centralized authority. Therefore a key technical challenge is that can we create a cryptocurrency without a central authority. To achieve this we need to decentralize

- agreement on a single published blockchain with a history of transactions
- agreement on which transactions are valid
- agreement on which transactions have occurred
- decentralized ID assignment
- decentralized mining of new coins

If we can solve all of these problems then we can build a currency that will be like ScroogeCoin but without a centralized authority. In fact this will be a system very much like Bitcoin.

18.12 Top Cryptocurrency Failures Quiz

Spacebit: provide a globally accessible blockchain through the use of nanosatellites

GetGems: a social networking platform that uses cryptocurrency to members that view ads in the app, popular in Uzbekistan

Dogecoin: a decentralized peer-to-peer digital currency, the community is friendly and vibrant and known for charitable acts such as sending the 2014 Jamaican bobsled team to the Olympics

Paycoin: according to a published white paper, it used new variations on the blockchain that would result in new breed of cryptocurrency

decentralized autonomous org (DAO): the largest crowdfund in history, an attacker exploited a vulnerability in its smart contract with losses totaling $50 million

This quiz highlights some important points about cryptocurrency:

- There is a strong market for it, but it is also a volatile field.
- It is not an easy problem to solve.

18.13 Bitcoins and Decentralization

The key technical problems we need to solve in Bitcoins are:

- who maintains the ledger or the blockchain
- who can decide which transactions are valid
- who creates new Bitcoins
- who determines how the rules of the system can change
- how do Bitcoins acquire exchange values

In addition we need infrastructure support for Bitcoins which includes exchange, wallet software, service providers etc.

18.14 Distributed Consensus

In Bitcoin there is no centralized authority thus it is important to achieve distributed consensus. There are two basic requirements for distributed consensus:

- the consensus protocol terminates when all correct nodes decide on a same value
- the value must have been proposed by some correct node

Bitcoin is a peer-to-peer system, and achieving consensus in a peer-to-peer system is very hard because

- some nodes may crash
- some nodes may be malicious
- the network is not perfect—not all pairs of nodes are connected and there are network faults and latency

When Alice wants to pay Bob she broadcasts the transaction to all Bitcoin nodes. In fact Alice doesn't know where Bob's computer really is and the computer may not be online.
The trick Bitcoin uses to achieve distributed consensus is to use incentive which is possible only because Bitcoin is a currency, and we embrace randomness or the imperfect nature of the peer-to-peer network by doing away with the exact termination point of the consensus protocol. As a result the protocol achieves consensus over a long timescale e.g. in about an hour.
The key idea here is implicit consensus. In each round a random node is picked, which then proposes the next block in the block chain. Other nodes implicitly accept or reject this block. If they accept they will extend the block chain from this block; if they reject they will ignore this block and instead extend the block chain from an earlier block. Every block here contains not only the transactions but also a hash pointer to the block that it extends.
Here is a high level overview of the consensus algorithm:

1. new transactions are broadcast to all nodes
2. each node collects new transactions into a block
3. in each round a random node gets a chance to broadcast its block
4. other nodes would accept a block if all of the transactions in it are valid, i.e. the signatures are valid and it is not double spent
5. if the nodes accept a block they will include a hash pointer to this block in the next block that they create

18.15 Bitcoin Safeguards

Alice cannot steal Bitcoins that belong to another user because there is no way Alice can create a valid signature that spends that coin as long as the underlying cryptography is solid.
Alice cannot launch a double spending attack, because only one of these transactions can be included in the blockchain, and to protect himself from this attack the new owner should wait until the transaction has multiple confirmations, that is there are multiple blocks extended from this block in the blockchain. The most common heuristics is to wait until you hear six confirmations.
To summarize the safeguards in Bitcoin:

- protection against invalid transactions is done through cryptography and also enforced by consensus
- protection against double spending is purely by consensus, there is no 100% guarantee that a transaction is in the consensus branch thus the guarantee is probabilistic, but the more confirmation you see the higher probability that a transaction is valid

18.16 Proof of Work Quiz

With regards to Bitcoin

- proof of work is costly and time consuming to produce, but not time consuming to verify
- to earn a coin miners of Bitcoins must complete all the work in the block
- changing a block requires regenerating all the successors and redoing the work that they contain, in fact this is a tamper resistant property

18.17 Incentives and Proof of Work

We want to reward nodes that extend from valid transaction blocks. Since we are dealing with Bitcoin which has value, we should be able to provide incentive to nodes that behave honestly.

18.18 Bitcoin Incentive #1

The first incentive is that, for a node to create a block the node can create a special transaction and can choose the recipient of this transaction. Typically the recipient is itself and the value of the created coin is currently fixed at 25 Bitcoins halved every 4 years. The block creator can collect this reward only if the block is accepted by the nodes. That means this block ends up in the long-term consensus branch. Note that there are a finite supply of Bitcoins (21 million). Unless we change the rules Bitcoins can run out in the year 2040.

18.19 Bitcoin Incentive #2

Another incentive that Bitcoin can provide is the so-called transaction fees. A block contains transactions and the creator of the transactions can choose to make output value less than input value. The difference is the fee paid to the block creator. Currently this is like a tip. But in the future when Bitcoins run out, this may become mandatory.

18.20 Sybil Attack Quiz

With regards to Sybil attacks

- The attacker creates a lot of fake identities and use them to change voting outcomes and control the network.
- Sybil attack is designed to attack reputation system in a peer-to-peer network.
- Sybil attack can be stopped if users are willing to give up anonymity.

18.21 Bitcoin Remaining Problems

- How do we pick a random node?
- The system can become unstable if everybody wants to run a Bitcoin code in order to capture some of these rewards.
- How can we prevent an adversary from creating a large number of Sybil nodes to try to subvert the consensus protocol?

18.22 Proof-of-Work

To select a random node we can select nodes

- in proportion to a resource that no one can monopolize
- in proportion to computing power using proof-of-work
- in proportion to ownership using proof-of-stake

18.23 Hash Puzzles

In Bitcoin we use proof-of-work. To create a block a node finds a nonce such that the hash of the nonce with a hash pointer to the previous block and all the transactions in this block is very small. Specifically this hash value has to be smaller than a target value.

$$H(\text{nonce}\|\text{previous hash}\|tx\|\ldots\|tx) < \text{target}$$

With this proof-of-work if Alice has 100 times more computing power than Bob, then she has about 99% chance of winning. In the long run Bob can still create 1% of the blocks.

18.24 PoW Properties

difficult to compute: Because the target hash value is very small it is very hard to find a correct nonce. In fact only the nodes with a lot of computing power, which are called the miners, would do this proof-of-work.

parameterizable cost: We can adjust the cost of proof-of-work. For example if the goal is to be able to create one block every 10 minutes then the nodes can automatically recalculate the target hash value every two weeks.

It is obvious that the probability that Alice will win the next block depends on the fraction of global computing hash power she controls. Therefore the key security assumption is that attacks are infeasible if the majority of the miners weighted by hash power follow the protocol.

trivial to verify: The nonce value has to be published as part of the block. Therefore other miners can simply verify that the hash value is smaller than target hash value.

Proof-of-work can be used as incentive when the mining reward (block reward + transaction fee) is greater than the mining cost (hardware + electricity cost), albeit there are complications such as fixed vs. variable cost and the dependence of reward on global hash rate.

18.25 Bitcoin Summary

What an attacker that controls 51% of the computing power can do:

steal coins from existing address: no, because digital signatures can prevent this

suppress valid transactions from the blockchain: yes, because the attacker can refuse to extend from that block

suppress valid transactions from peer-to-peer network: no, because the transitions are broadcast to the entire network

change the block reward: no, because that is determined by the whole system

destroy confidence in Bitcoin: yes, because for example by refusing to extend from the valid block

19 Lesson 19: New & Alternative Cryptocurrencies

19.1 Bitcoin Operations

in Bitcoin a valid transaction has information stored in a public blockchain but also the transactional information has to be signed by the owner's secret signing key—the public key-private key pair. To

keep private key secret and secure the simplest approach is to store the key in a file, on your computer, or on your phone. However if the device is lost or wiped then your key is lost and all your Bitcoins are lost; if your device is compromised then your key can be stolen and all you Bitcoins can be stolen.

19.2 Bitcoin Wallet Quiz

With regards to Bitcoin wallet

	hot storage	cold storage
location	online	offline
convenient	yes	no
security	risky	archival but safe

We can put the top secret or the less frequently used master secret in a cold storage and the most frequently used keys in hot storage. In fact we can have the best of both worlds with both hot storage and cold storage by moving coins back and forth between the hot side and the cold side using separate keys. We need separate keys because otherwise the coins in the cold storage will be vulnerable if the hot storage is compromised. But with separate keys each side will need to know the others' addresses or public keys.

19.3 Hierarchical Wallet

Suppose for privacy or other reasons we want to be able to receive each Bitcoin at a separate address, i.e. whenever we transfer coin from the hot side to the cold side we would like to use a fresh cold address for that purpose. The blunt solution for the hot side to find out the cold addresses is for the cold side to generate a big batch of addresses and send those over to the hot side. The drawback is that the hot side need to reconnect with the cold side periodically in order to transfer more addresses.
A better solution is to use the so-called *hierarchical wallet*. In a hierarchical wallet instead of generating a single public key/address we generate what we called address generation information, and instead of generating a single private key we generate what we call private key generation information. Given the address generation information we can generate a sequence of addresses. Likewise given the private key generation information we can generate a sequence of private keys. The cryptographic magic that makes this useful is that for every i the ith private key is paired up with the ith address. In other words we have a sequence of regular key pairs.
The other important cryptographic property here is security, that is the address generation information here does not leak any information about the private keys. On the other hand not all digital signature schemes can be modified to support hierarchical key generation. But the good news is that the digital signature scheme used by Bitcoin, elliptic curve, does support hierarchical key generation.
To summarize the cold side creates and saves private key generation information and address generation information. It does a one-time transfer of the address generation information to the hot side. The hot side generates a new address sequentially every time it wants to send a coin to the cold side. When the cold side reconnects it generates addresses sequentially and checks the blockchain for transfers to these addresses. It can also generate private keys sequentially if it wants to send some coins back to the hot side.

19.4 Cold Storage

There are multiple options to keep the cold storage safe:

brain wallet: encrypt information under passphrase that user remembers

paper wallet: print information on paper and lock up the paper

tamperproof device: use hardware to protect the keys

19.5 Online Wallets and Exchanges

Bitcoin exchange accepts deposits of Bitcoins and fiat currency (dollars, pounds, euros) and promises to pay back on demand. These exchanges let customers

- make and receive Bitcoin payments
- buy and sell Bitcoins using fiat currency
- match a Bitcoin buyer with a Bitcoin seller

Therefore Bitcoin exchanges are similar to banks. As a result Bitcoin exchanges need to meet a number of minimal requirements or expectations e.g. meeting the minimum reserve requirements. That is they must hold some fraction of deposits in reserve. A Bitcoin exchange needs to provide a proof of reserve to give customers some comfort about the money they have deposited, from 35% to even 100% of the deposits that people have made.
We can break the proof-of-reserve problem into two pieces

- To prove how much reserve you are holding, the exchange can simply publish a valid payment-to-self transaction of the claimed reserve amount. For instance if it claims to have 100,000 Bitcoins it creates a transaction in which it pays 100,000 Bitcoins to itself and show that the transaction is valid. Strictly speaking this is not a proof that the exchange really owns this amount, but only that whoever does own this amount is willing to support the exchange.
- To prove that how many demand deposits you hold, the exchange can use Merkle tree

19.6 Merkle Trees

A Merkle tree is a binary tree that is built with hash pointers so that each of the pointers not only sets where we can get a piece of information but also what a cryptographic hash of that information is. A Bitcoin exchange executes a proof of how many demand deposits it holds by constructing Merkle tree. In this Merkle tree each hash pointer also includes an attribute—a number that represents the total value in Bitcoins of all the deposits that are in a subtree underneath this hash pointer. For this to be true the attribute value of each hash pointer has to be the sum of the values of the two hash pointers beneath it. By constructing this tree and publishing its root value, the exchange is making the claim that all users are represented in the leaves of the tree and that the deposit values are represented correctly. Moreover the deposited values are propagated correctly up the tree so that the root value is the sum of all the users' deposit amounts, the total liabilities of the exchange. Now each customer can go to the exchange and ask for the proof of correct inclusion of his deposit. The exchange must then show the customer the partial tree from the user's leaf up to the root. The customer then verifies that

- the root hash pointer and root value are the same as to what exchange has signed and published
- the hash pointers are consistent all the way down, that is each hash value is indeed the cryptographic hash of the node it points to
- the leaves contains correct user account information e.g. user ID and the deposit amount
- each value is the sum of the values of the two values beneath it

It is very important to note that the exchange cannot present different values in any part of a tree to different customers, because that will imply that either the exchange can find a hash collision or it can present different root values to different customers.

19.7 Proof of Reserve

To provide a proof of reserve the exchange proves that

- they have at least X amount of reserve currency by doing a self transaction of X amount
- the customers have at most amount of Y deposited

This shows that their reserve fraction is at least X/Y. Notice that this proof is independently verifiable by anybody and no central regulator is required.

19.8 Anonymity

Literally anonymity means without a name. In Bitcoin an ID is an address which is the hash of a public key. In computer science we call this **pseudonymity**. In computer science

$$\text{anonymity} = \text{pseudonymity} + \text{unlinkability}$$

We know that Bitcoins provides pseudonymity. Now the question is does Bitcoin provide unlinkability, that is in Bitcoin different transactions of the same users cannot be linked together. More precisely unlinkability in Bitcoin means that

- it is hard to link different addresses of the same user
- it is hard to link different transactions of the same user
- it is hard to link the sender of a payment to its recipient

19.9 Bitcoin Anonymity Quiz

With regards to Bitcoins

- a time-stamping service prevents people from double spending Bitcoins
- the expenditure of individual coins can be tracked

19.10 De-Anonymize Bitcoin

In Bitcoin it is trivial to create a new address. All you need to do is to create a new private key-public key pair and the hash of the new public key becomes a new address. Hence a single user can own many addresses or many IDs. In fact the best practice is to use a fresh new address for the recipient in any new transaction. Thus the question is, can the addresses and the transactions be linked.
Suppose that Alice has some Bitcoins in her different addresses. In order to pay for a good Alice has to create a single transaction that includes input from two addresses. But by doing so Alice has revealed that these two addresses belong to the same user. This example shows that if multiple addresses are used in the same spending transaction, then these addresses are under joint control. In other words we can link these addresses together.
Furthermore address linkability is transitive and we can propagate such linkability. Suppose Alice creates a transaction where she would transfer Bitcoins from two of her addresses to pay for a good

but also send the change to another address of hers. This may leave an attacker to suspect that one of these output addresses also belongs to the same user. Therefore the attacker can know that these three addresses belong to the same user.
We can not only link the addresses that belong to the same user but also see if any of these addresses reveals the real identity of a real user. For example an address may be used in transaction through a provider that is legally required to know the true user behind the address, or a user may have posted an address in a public forum because he is making a donation.

19.11 Decentralized Mixing

One mechanism to make transaction link analysis less effective to protect unlinkability is to use mixing. The main proposal for decentralized mixing for Bitcoin is a decentralized system called *Coinjoin*. In this protocol

- Different users jointly create a single Bitcoin transaction that combines all inputs. Furthermore in a transaction that has multiple inputs coming from different addresses, the signatures corresponding to each input are separated from and independent of each other. This allows a group of users to mix their coins with a single transaction.
- Each user supplies an input address and an output address and forms a transaction with these addresses.
- The order of input and output addresses is randomized so that an outsider would be unable to determine the mapping between inputs and outputs.
- All the participants check that its outgoing address has been included in the transaction and that it receives the same amount of Bitcoin that they are inputting. Once they have confirmed this they sign the transaction.

Here is the procedure to create Coinjoin:

1. a participant needs to find peers who want to mix
2. they exchange the input and output addresses to be included in the transaction
3. they construct the transaction by mixing the orders
4. the transaction is sent around and each peer can check that his or her input and output have been included before signing, the signatures are then collected
5. this Coinjoin transaction is broadcast to the Bitcoin system

It is important for the participants to exchange these input and output addresses in such a way that even the other members of the peer group do not know the mapping between input and output addresses. To exchange these addresses in unlinkable way we need an anonymous communication protocol. For example the peers can use Tor to exchange the input addresses so that no one knows what input address each other is using. Once that is accomplished it is not necessary to communicate the output addresses in a secure way because there is no way to link an output address with an input address.

19.12 Bitcoin Append-Only Log

The Bitcoin system can be used for other applications:

secure timestamping: We can prove knowledge of x or we know the value of x at a time t. But we can do so without revealing the actual value or knowledge of x at time t.

hash commitment: Hash function is a one-way function and resistant to collision. Therefore if we publish the hash value of x, we essentially announce the commitment to x and we don't have to reveal x to anyone. In other words everyone know that we actually know the true value of x.

19.13 Timestamping

Secure timestamping has many applications, e.g. proof of knowledge that we discussed above, proof of receipt etc. The simplest solution to secure timestamping in Bitcoin system is that, instead of sending money to the hash of a public key send it to the hash of your data and then announce the hash of your data. The caveat here is that the coin you sent can be lost forever because you don't know who happens to own the address that corresponds to the hash of your data. Thus the pro and con are

pro: simple and compatible

con: have to burn coins, more importantly the miners don't know that the coin you send to this address is lost forever, thus they might check it forever just like any other legitimate Bitcoins

A more sophisticated approach is called *CommitCoin*, which allows you to encode your data into the private key. A property of elliptic curve is that if you use bad randomness in making signature it will leak the private key. CommitCoin exploits this property. In CmomitCoin we generate a new private key that encodes our commitment and derive its corresponding public key. We then send a tiny transaction to the address that corresponds to the public key, and send it back two chunks of Bitcoins. We use the same randomness to sign both chunks. This allows anyone looking at the blockchain to compute the private key using these two signatures and the private key contains the commitment. The pro and con of this approach are

pro: avoids the need to burn coins and the miners don't have to keep track of unspendable coin forever

con: quite complex

As of 2014 the preferred way to do Bitcoin timestamping is with the OP_RETURN transaction. The OP_RETURN instruction immediately returns an error so that this script can never be run successfully, and the data that is encoded in the transaction is ignored. This can be used to encode arbitrary data. As of 2015 OP_RETURN allows 80 bytes of data to be pushed, which is more than enough for a hash function output (the output length of SHA256 is 32 bytes). As of late 2014 there are already several websites that help with this. They collect a bunch of commitments from different users and combine them into a large Merkle tree. Then they publish one unspendable output containing the Merkle tree root. This acts like a commitment for all the data that users wanted to timestamp that day.

19.14 Overlay Currencies

Because we can write whatever data we want into Bitcoin, we can use Bitcoin as it exists today as an append-only log and write all the data that we need for our new currency system directly into the Bitcoin blockchain. We call this approach an **overlay currency**. That is Bitcoin serves as the underlying substrate and the data of the overlay currency is written into the Bitcoin blockchain using unspendable transaction outputs. However

- Bitcoin miners will not validate what you're writing into the blockchain because they don't know the new currency system.
- Anyone can write anything as long as they're willing to pay the transaction fees.
- We must develop our own logic for validating transactions, which must reside in each end-user client that participates in sending or receiving overlay currency.

19.15 Mastercoin

An example of overlay currency is Mastercoin. In an overlay currency system such as Mastercoin

pro: There's no need to develop a new consensus algorithm, and therefore developers can instead focus on developing interesting features such as smart contracts, user-defined currencies etc.

con: It still depends on Bitcoin, which can be inefficient because overlay currency may need to process a lot of data since Bitcoin nodes don't filter these transactions for you.

20 Lesson 20: Attack Tolerant Systems

20.1 WWW Robustness Quiz

The internet is a scale-free network. We can infer that it has a high degree of tolerance towards random failure and a low degree of tolerance against attacks. The most successful attacks target the nodes that are the most connected. When a highly connected node is attacked the internet begins to fracture and splinter into unconnected networks. It is this characteristic of the internet that makes attacks so successful.

20.2 Node Connectedness Quiz

Determining which nodes are the most connected is an interesting problem. There are basically three different methods of determining node connectivities:

average node degree: look at nodes with the largest number of nodes connected to them

node persistence: look at nodes that are most likely to appear in a snapshot of the traffic

temporal closeness: look at nodes that interact with the largest number of nodes.

20.3 Defense in Depth

In the principle of defense in depth, the first layer of defense is prevention, the second layer is detection, and the third layer is about surviving the attack. The best way to survive an attack is to able to tolerate the attack, that is an attack will not be able to render our network or system ineffective. When we say our network and systems should tolerate an attack, we mean that

- the confidentiality, availability and integrity of our data remain intact
- the availability and integrity of our system services remain intact

Since data is stored in one place, all the attacker needs to do is to compromise that data storage server. Now if we replicate the data and store the copies in n different servers, for confidentiality protection this is actually a weaker scheme because the attacker can get data from any of these n servers. On the other hand if we use a majority voting scheme then integrity and availability are better protected, because now the attacker needs to compromise the majority of these n servers.
So the question is can we have a better confidentiality protection for our data?

20.4 Naive Secret Sharing Quiz

One approach is by secret sharing. Cryptographic secret sharing involves giving each party a share of a secret, which can only be reconstructed if all parties participate and share their portion. The major weakness of this naive secret sharing scheme is that, the more shares you have of the secret the less work you have to do to guess the secret. In a secure scheme it should not matter how many shares of a secret a party has. It should take the same amount of guess work as a party with no share.

20.5 Secret Sharing

In cryptography **secret sharing** refers to a method of distributing a secret among a group of participants, each of which is allocated a share of the secret. The secret can only be reconstructed when the shares are combined together; individual shares are of no use on their own. With secure sharing we can give a tighter control and remove any single point of vulnerability with regard to confidentiality, because even if an attacker has compromised an individual key shareholder he still cannot access or change the data. We can also improve integrity and availability by replicating each share among a group, i.e. each key shareholder is a group of members where each member stores a replica of the share.

20.6 Mathematical Definition

The goal of secret sharing is to divide some data D into n pieces D_1, D_2, ..., D_n in such a way that

- knowledge of any k or more pieces of D makes D easily computable
- knowledge of any $k-1$ or fewer pieces of D leaves D completely undetermined i.e. all possible values are equally likely

This scheme is called (k, n) threshold scheme. If $k = n$ then all participants are required together to reconstruct the secret.

20.7 Shamir's Secret Sharing

Suppose we want to use (k, n) threshold scheme to share our secret S where $k < n$. Choose at random $k-1$ coefficients a_1, a_2, ..., a_{k-1} and let the secret S be a_0, which together define a random $k-1$ degree polynomial

$$q(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1}$$

All arithmetic operations are done modulo a prime number p that is greater than both S and n, and the coefficients a_1, a_2, ..., and a_{k-1} are randomly chosen from a uniform distribution over the integers in $[0, p)$. Then construct n points on this polynomial which read $(i, S_i = q(i))$ where i = 1,2,...,n. These S_i's are called the shares of S. Given any subset of k of these points $(i, q(i))$ we can find the coefficients of the polynomial $q(x)$ by interpolation. This is guaranteed by the mathematical property that any k points uniquely determine a polynomial of degree $k-1$. Once $q(x)$ is determined evaluate $S = q(0) = a_0$ which is the secret. That is given any k shares we can reconstruct the secret.

20.8 Shamir's Scheme Example

Suppose $k = 2$ i.e. the polynomial is a linear function $q(x) = a_0 + a_1 x$. With a single share/point we cannot determine $q(x)$ because there are infinite many lines that go through this point, which implies that the secret can be any value with equal probability. Obviously we need two points to determine a line and any two points on the line can determine the same line, meaning any two of the $n = 2$ shares can reconstruct the original secret S.
In general to determine $q(x)$ given (x_i, y_i) where $y_i = q(x_i)$ we should use the Lagrange interpolation algorithm.

20.9 Sharmir's Scheme Example 2

Suppose $(k, n) = (3, 5)$, $S = 7$, $a_1 = 3$, $a_2 = 5$, and $p = 11$, that is $q(x) = 5x^2 + 3x + 7 \pmod{11}$ and the 5 shares are $S_1 = q(1) \pmod{11} = 4$, $S_2 = q(2) \pmod{11} = 0$, $S_3 = q(3) \pmod{11} = 6$, $S_4 = q(4) \pmod{11} = 2$, $S_5 = q(5) \pmod{11} = 4$. Then we can use the following Lagrange interpolation formula to reconstruct $q(x)$

$$q(x) = \sum_{i=1}^{k} y_i \prod_{j=1, j \neq i} \frac{x - x_j}{x_i - x_j}$$

20.10 Shamir's Scheme Summation

In practice using $n = 2k-1$ requires attackers to acquire more than $\lfloor n/2 \rfloor = k-1$ shares to reconstruct the original secret, and in general we typically assume that an attacker cannot compromise the majority of our systems.
In summary the security properties of Shamir's secret sharing scheme are

- shares can be dynamically added or deleted without affecting other shares
- it is easy to create new shares thus enhance the security without changing the secret
- it is easy to create hierarchical scheme for organizations where hierarchy is important, e.g. supply participants with different number of shares according to their importance
- it is information theoretically secure, i.e. this scheme cannot be broken even if the adversary has unlimited computing power

20.11 Practical Byzantine Fault Tolerance

A **fault** in the system is the cause of an error that leads to a system failure. A system error is because of a fault and error leads to a failure (fault $\rightarrow$ error $\rightarrow$ failure). Fault tolerance can be achieved through failure masking and failure masking can be achieved through redundancy.

20.12 Redundancy of System Services

The goal of a redundant system is that a set of non-faulty services can reach a consensus even in the presence of some corrupted or faulty services. This consensus is then the correct service that a system will provide.

20.13 Redundancy Quiz

availability: probability the system operates correctly at any given moment

reliability: system ability to run correctly for a long interval of time

safety: failure to operate correctly does not lead to catastrophic events

maintainability: ability to easily repair a failed system

20.14 Byzantine Generals Problem

Getting a set of non-faulty services to reach consensus is actually quite challenging. The Byzantine generals Problem refers to a hypothetical situation in which a group of generals, each commanding a portion of an army plan to attack a city. The generals must decide to attack or wait. Every general must agree on a common decision, anything less will result in a defeat. The problem is compounded by the fact that there may be traitorous generals that do not vote for the best strategy, and of course there can be disagreement among the generals.

20.15 System Models

The Byzantine generals problem is an example of asynchronous distributed systems where nodes are connected by a network. In such systems we typically assume

network: the network is unreliable, it may fail to deliver, delay, duplicate, or deliver out of order

Byzantine failure model: a faulty node may behave arbitrarily i.e. its behavior is not defined, and the nodes fail independently i.e. there is no correlation of failures among the nodes

adversary: attackers cannot delay correct nodes indefinitely and cannot subvert cryptographic techniques

20.16 System Properties

When we say an asynchronous system achieves fault tolerance we typically mean that it has both safety and liveness.

safety: even if the system fails nothing serious happens

liveness: clients of the system can eventually receive replies to their requests

In order for an asynchronous system to provide safety and liveness it needs to have a minimum $3f + 1$ replicas where f is the maximum number of faulty replicas. $n = 3f + 1$ replicas are needed because it must be able to proceed after communicating with $n - f$ replicas since f replicas might be faulty and not responding. However it is possible that the f replicas that did not respond are non-faulty and therefore f of those responded may be faulty. Even so there must be enough responses that those from the non-faulty replicas outnumber those from the faulty ones, that is $(n - f) - f > f$ therefore $n > 3f$.

20.17 System Algorithm

A client sends a request to invoke a service operation in the primary replica. The primary multicasts the request to the backups i.e. the group of replicas. The replicas execute the request and send a reply to the client. The client waits for $f + 1$ replies from different replicas with the same result, and that is the result of the operation.

20.18 Attack Tolerance

We cannot apply fault tolerance techniques for system services to achieve attack tolerance, because in fault tolerance all replicas runs the same program thus the same attack can compromise all replicas. Therefore redundancy is not a solution for attack tolerance.
To achieve attack tolerance we need to use diversification. That is each instance should use a different implementation and this applies to all layers of the stack, including network and application protocols, programming languages, operating systems etc. and each instance should use a different security protection mechanism or apply a different security mechanism to different parts of the program. This can achieve efficiency by reducing security overhead because not all operations are checked all the time. It can also help us identify the attacks based on which protection mechanism works and which fails.
We can take the idea of diversification one step further to use what we call moving target technique to achieve attack tolerance. That is we can dynamically change our network and system configurations. For example we can have many instances of the system and network services. Each instance can have a different implementation, and these instances can be composed on-the-fly.