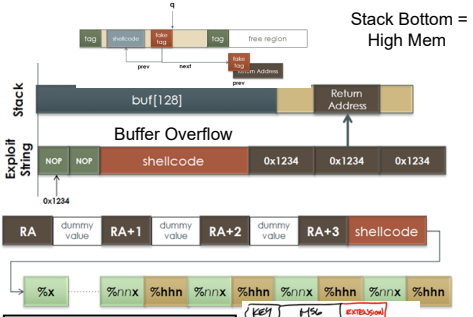


Input Parameters
Return Address
Saved Frame Pointer
Local Variables
Saved Registers

Procedure Linkage Table (PLT) and Global Offset Table (GOT)
GOT: table of pointers to functions. Contains absolute memory location of each dynamically loaded library function
PLT: table of code entries. used to jump to locations in GOT
-Each code entry invokes corresponding function pointer in GOT
Ex. sprintf@PLT code may invoke " jmp GOT[k] where k corresponds to the sprint function index in GOT
- All calls to dynamic libraries jump to PLT
- first time function called, runtime linker invoked to load library
- runtime linker updates GOT entry, based on where library is loaded
- PLT/GOT contain commonly used library functions such as printf
-PLT/GOT always appear at a known location

Needham-Schroeder Protocol:
A -> T : { A, B, N_A }
- A picks a nonce (random number)
T -> A : { N_A, K_AB, B, {K_AB, A}K_B }K_A
- Nonce in reply from server assures A that this is not a replay
- Including B ensures no man-in-the-middle, who A wants to talk to
- Everything is encrypted with A's key, so safe
A -> B : { K_AB, A }K_B
- B will know who they are talking to as A is in it
B -> A : { N_B }K_AB
- Send a new nonce to allow B to make sure this is not a replay, and that their actually talking with A as they share K_AB
A -> B : { (N_B – 1) }K_AB

Double Free:
tag = q - sizeof(chunkTag);
tag->next->prev = tag->prev;
tag->prev->next = tag->next;

tag->next = return addr
tag->prev = shellcode

Stream cipher is faster, no mode required, less compuationally complex, better random access

Stackshield: Put return addresses on a separate stack with no other data buffers there
Stackguard: random canary value is placed just before the return address. Just before the function returns, the code checks the canary value and, if the value has changed, the program is halted
Address Space Layout Randomization (ASLR):
- OS maps stack of each process at randomly selected location with each invocation
- attacker will not be able to easily guess the target address
- ASLR also randomizes location of dynamically loaded libraries, making it harder to perform return into libc attacks or GOT overwrites
Non Executable Pages (NX):
- If stack is made non executable, then shellcode on the stack will not execute
- Page tables have NX protection bit. Requires support from OS

Domain Name System (DNS): IP address discovery (URL->IP)
Cache Poisoning (Response Host):
- Attacker initiates DNS request to victim nameserver, then sends multiple DNS responses with spoofed IP address with guessed queryID (from the outside)
- If spoofed response has correct queryID, victim nameserver will have poisoned cache entry for URI website
Cache Poisoning (Domain nameserver):
- targeting one mesg earlier in the victim nameserver exchange
- attacker performs several DNS requests for wwwN.bankofsteve.com to victim nameserver where N changes with each request
- Then sends several DNS responses with spoofed authoritative nameserver
- If correct, nameserver is cached and attacker controls the entire domain
DNS Rebinding Attack:
- DNS rebinding attack replaces mapping for attacker's domain with victim's IP
- allows bypassing the browser's same origin policy because attacker's IP and victim's IP appear to belong to the same domain
Steps:
- Browser: Attacker gets client to visit attacker's site
- Attacker: Attacker controls DNS of site, and returns DNS response with short TTL. Attacker's site serves a web page with malicious script.
- Browser: script makes request to attacker's web site. In turn, browser makes another DNS query, which reaches the attacker's DNS, since the cached entry had a short TTL.
- Attacker: This time attacker's DNS returns IP of a victim web server
- Browser: Now victim's web server and attacker's web server appear to be in the same origin

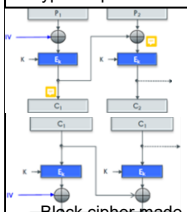
Clock Glitching:
-introduce rapid clock pulse longer than time to increment the PC and shorter than instruction fetch
-get instruction skipping
- PC increments, but instruction not fetched, so previous instruction gets executed.

Confidentiality: data secrecy, provided by ciphers
Integrity: trustworthiness of data, provided by hashes
Authenticity: allows principle to prove identity or origin of data, provided by signatures
message authentication codes (MACs)
Non-Repudiation: prevents principle from denying they performed an action, achieved by trusted third party

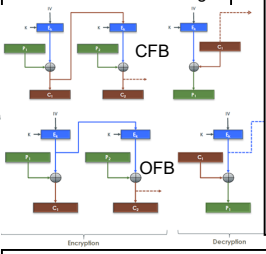
- bootloader, controls startup of application
- processor go into provisioning mode, gen random encryption keys
- processor blows fuses, so can't extract keys
- Secure Hash (bootloader, key) = Digest
- everytime boots, checks digest, recalculates hash and cmps digest match

Modes hide patterns, protect against frequency/pattern attacks. Secure cross blocks and chaining prevent reordering and similar block encrypt to ciphertext

CBC (Cipher block chaining):
- hide patterns
- IV does not have to be secret, but shouldn't be reused and random, so that plaintext doesn't encrypt to same
- No parallelism for encryption (sequential)
- Decryption can be parallelized
- If you wanted to start in the middle of a video file, then you can start a bit before the middle due to the chaining



Block cipher made up by:
- iterated block cipher repeatedly applies these two ciphers in different combinations
Substitution cipher:
- replaces characters in the plaintext with other characters (mapping)
- confusion
Permutation cipher:
- transposes the plaintext characters
- diffusion



Block ciphers:
Confusion
- Obscures relationship between plaintext and ciphertext
- Makes statistical analysis difficult
- Non-linear => E_k(M1 + M2) != E_k(M1) + E_k(M2)
- Each character affected by entire key
- Confusion performs substitution and adds non-linearity. However, still allows attacker to know that affecting bit in ciphertext will affect same portion in plaintext.
Diffusion:
- Spreading the influence of individual plaintext characters over ciphertext
- hides patterns
- many input bit -> to output bit
- repetitive patterns in plaintext spread over ciphertext, thus hides statistical info about plaintext
- each plaintext character adds a bit to each ciphertext character, helps hide patterns
- Diffusion hides this issue by spreading affect of bit (hides substitution mapping)

Diffie-Hellman and Public Key Cryptography (RSA) allow establishing secure communications without a trusted server
- protocol for live key exchange, without trusted server
-for short mesg as very computational expensive
- establish a common (short) secret over an insecure link
- Not very efficient for long messages

Protocol:
- Alice selects n=(a large prime modulus), select number g=(a generator of the field n that lies between between 1 and (n-1)) g^x mod n = y
To establish secret key (can occur publicly)
- Alice selects a random integer x and computes: P = g^x mod n
- Alice sends P, g and n to Bob; x is kept secret
- Bob selects a random integer y and computes: Q = g^y mod n;
- Bob sends Q back to Alice; y is kept secret
- Alice computes Q^x mod n = g^xy mod n
- Bob also computes P^y mod n = g^xy mod n
- both share the secret g^xy mod n

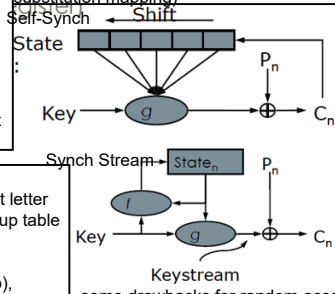
Problems:
- diffe-hellman does not prevent man-in-the-middle attack, can't tell communicating with (no authentication)
- Man-in-the-middle can pretend to be the other person in the conversation and pass through t
own shared key with each party

A->C {P,g,n}, x -> g^xy
C->A {y} -> g^xy
C->B {P, g, n}, g^mn
B->C {n}, g^mn
Hence, C has g^xy and g^mn

PKI-X.509:
- CA signs public key certificate using private key, use CA public key certificate to decrypt and validate signature
- no longer a single point of failure 3rd party server
- changes trust level, not giving 3rd party server private key

SSL: fragment, compress, add MAC to each, encrypt, header describing protocol
- sent to TCP in OS

Self-Synch Stream Cipher:



Random access due to shift register
- With self synchronizing ciphers, an adversary can replay previously sent ciphertext into a stream, and the cipher will resync

some drawbacks for random access, due to state
- ex. hard drive, have to decrypt from beginning till point needed
- efficient for short/traditional messages and not for large encrypted volumns

Cesur Cipher -> Shift Cipher -> Substitution Cipher
- Each plaintext letter is replaced with exactly one ciphertext letter
- The key = mapping between plaintext and ciphertext, lookup table
- Do not hide frequency
Polyalphabetic/period cipher:
- Instead of one mapping, have set of n mappings (n lookup), change mapping with every character (use a different lookup table per each character)
One-Time-Pad (OTP)/Vernam cipher:
- polyalphabetic cipher that never repeats
- random substitution used for every character (ie. infinite lookup tables)
- C = K XOR P
- plaintext bit is flipped when key bit is 1, same when 0
- Each key can be used only once

Three party diffe-hellman:
1) A,B,C generate random values: a, b, c
2) A,B,C calculate g^a, g^b, g^c
3) A->B: {g^a}, B->C: {g^b}, C->A: {g^c}
4) A: g^ac, B: g^ab, C: g^bc
5) A->B: {g^ac}, B->C: {g^ab}, C->A: {g^bc}
6) A: g^abc, B: g^abc, C: g^abc
Shared Key: g^abc

Public-Key Cryptosystems:
To encrypt:
sender encrypts the message with the intended recipient's public key
Only the recipient should have the private key to decrypt message
To provide authentication/signing/non-repudiation:
Sender encrypts with their private key
Anyone can decrypt with senders public key, proves sender was one who created mesg

Border Gateway Protocol (BGP):
- gateway routers communicate using BGP protocol to update routing information. route discovery.
- Routes updated in peer-to-peer manner, asking nieghbours

Attacks on BGP:
- Attackers compromise protocol by advertising "good" routes to destinations
- Traffic then gets directed through attacker's gateway
- Attack similar to APR
- Routers trust each other and don't authenticate

HMAC = H[(K ⊕ opad) + H((K ⊕ ipad) + M)]

ARP Attack
- The attacker makes hacked machine redirect all traffic to itself by responding to all ARP broadcasts
- Every machine starts to think that the hacked machine owns every IP address
- All LAN traffic is redirected to hacked machine which can start faking services, stealing passwords, etc.
- ARP broadcasts are never forwarded outside of a subnet, so the attacker must control a machine on a subnet

Address Resolution Protocol (ARP): used for MAC address discovery (IP->MAC)
- (IP layer): uses IP addresses for routing packets in link layer
Protocol:
- host that wants to send a packet to IP address A, it will perform an ARP broadcast to all devices asking which device owns that IP address
- All hosts ignore the broadcast except the host that has the IP address A, responds with its MAC address
- All packets sent to IP Address A are sent using the given MAC address
- ARP msgs are not sent beyond local network

Certificate Pinning: ensure connecting to correct server (no MITM), can pin a certificate locally and check the certificate hash signature matches that of the certificate provided by server.

1) something you are (bio)
2) something you have (credit card)
3) something you know (password)
4) something you can do (captcha)

rootkit hide systems resources:
1) User mode: intercepts libraries or systems calls at the user level and modifies returned results
2) Kernel mode: intercepts system calls in the kernel
- affect system call table lookup for function

OAuth: used for authentication, but is authorization protocol
- allows third-party services to exchange your information without you having to give away your password.
Authentication: Service wants to determine identity of untrusted user. Login
Authorization: User wants to permit an untrusted service access to sensitive resource
- User sends token to service, so service doesn't know user but trusts by token
OAuth Authorization Code Grant:
- 3 parties in authorization/authentication even
1) Resource Owner (user)
2) Authorization Server (SSO get token)
3) Client (Websit)

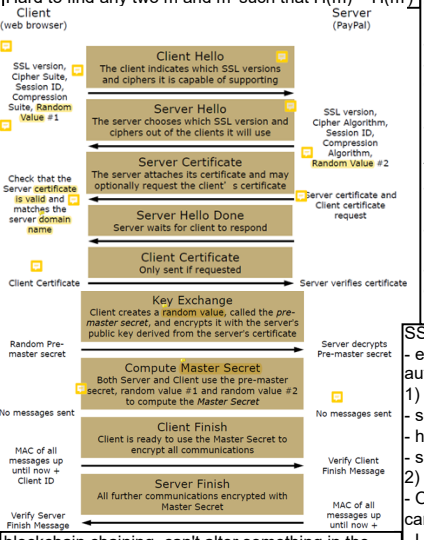
IPSec: same as SSL (confidentiality, integrity, source authentication, protection against replay)
1) Authenticated Headers (AH): all but confidentiality
- Protects IP packet (except IP header fields altered during transit) using MAC stored in AH Header
2) Encapsulating Security Payload (ESP): Provides all properties
- Payload encrypted to protect contents
- can be used even when all routers not IPSec enabled
A) Transport mode: Both endpoints support IPSec, but intermediary routers do not, like SSL
B) Tunnel mode: Endpoints do not support IPSec, but endpoint gateways do
- This mode encrypts/authenticates the packet header and payload and encapsulates it in another regular IP packet
- Similar to SSH tunneling or VPN software

IPSec vs. SSL:
- SSL can provide more security because if one connection is compromised, others are not
- SSL can provide better access control since IPSec allows access to entire network

Cryptographic hash 3 properties:
1) Preimage Resistance:
Given a hash value, hard to find a preimage that will yield the hash value
Given: h, H -> hard to find m such that H(m) = h

2) 2nd preimage resistance:
Given a preimage, hard to find another preimage that hashes to the same hash value
Given: m -> hard to find an m', such that H(m) = H(m')

3) Collison Resistance:
Hard to find collisions (2 preimage values that coincidently hash to the same hash value)
Hard to find any two m and m' such that H(m) = H(m')



Same Origin Policy: same origin policy dictates that scripts from one origin (a web site) cannot access or set the properties of a document/cookies from another origin (another web site)
-same protocol (ex. https), same hostname, same port

Proof of Retrieval (POR):
- customer encrypts file and randomly embeds set of rand blocks called sentinels
- encryption renders sentinels indistinguishable rest
- customer later challenges CSP by asking for a random collection of sentinel blocks
- If CSP has modified or deleted substantial portion of file, then with high probability it will also suppressed a number of sentinels
- Checksums are used to detect the possibility of small changes being made

Provable Data Possession (PDP):
- client pre-computes tags for each block of a file
- Tags computed using homomorphic encryption: means tags computed for multiple arbitrary file blocks can be combined into a single value
- At a later time, client can verify the server possesses the file by generating a challenge against a randomly selected set of file blocks
- server calculates result for the requested blocks, and sends it back as a proof of possession
- client is convinced of data possession, without actually having to retrieve the file blocks.
- deals with homomorphic encryption: allows you to perform operations on encrypted data, work done on encrypted data reflects in the unencrypted data when you decrypt it, like RSA

Cross-site request forgery (CSRF):
- Attacker tricks user into visiting a website that contains a link to a site that the user may have visited previously
- If the user's browser contains a valid authentication cookie, the attacker issues an authenticated request on behalf of the user
- Web site cannot distinguish between requests
<img src="URL?param=value&param=value"> <iframe>, if token two (pasrse page)
Defense:
- Limiting lifetime of authentication cookies
- Checking the HTTP referrer header: When visiting a webpage, the referrer is the URL of the previous webpage from which a link was followed
- Requiring secret token information in GET and POST parameters

CSRF token bypass was accomplished in the lab by performing form post, posting a script, when triggered by user starts attack to perform transfer from user. two-phases to validate request. parsed returned validation page to extract token, passed back

Covert channel: intentional flow of information out of a system by an attacker.
- exists whenever the actions of one process affect the actions of another process in some way, without explicit communication.

Non-interference property allows analyzing covert channels.
- states: a system has non-interference property iff any sequence of inputs to a process will produce the same outputs, regardless of inputs to another process

Cross-Site Scripting = XSS
Reflected:
- Attacker crafts URL that targets a vulnerable site
- website not modified
- part of URL becomes part of web content
Persistent:
- attacker posts script to vulnerable site, blog post
- website modified
Defense:
- input validating
- convert special characters before sending to user
- whitelisting: Allow a small set of safe characters, rather than disallow specific dangerous tags
- HTTP_only cookies: Web site can tag certain cookies as being inaccessible to Javascript. Web browser will then not let any Javascript read the cookie, even if it is from the same site

Cache Timing Exploits:
- attacker run on same processor as victim use shared cache as timing channel to infer information about victim
Prime Phase:
- attacker fills shared cache with data, evicting all victim's data
- attacker lets victim execute, using shared cache
- Loads by victim cause attacker's data to be evicted
Probe Phase:
- Attacker reads their data from cache and times how long each read takes
- Some accesses will take longer because cache miss, so attacker can infer which cache lines victim accessed between prime and probe
Defense:
- Allocate memory such that no overlap in cache lines used by different customers or can't evict from cache thus not affect timing

SSL:
- encrypt data -> confidentiality, integrity, authenticate server machine
1) Key exchange/handshake
- secret key between sender and receiver
- handshake bottleneck - asymmetric key
- shared secret key, symmetric
2) Communication:
- Once keys setup, arbitrary number of messages can be exchanged
- Large amounts of data can be transmitted, efficient, encrypted using symmetric fast
- Hello plaintext, vulnerable (downgrade attack)
- can use certificate pinning to prevent MITM
- random values for live conversation
- pre-master secret encrypted with server key, so client knows only server able decrypt, server knows not replay as contributed random value
- master secret key used for encrypt messages going forward compatible version
- Finish messages with MAC are encrypted
- MAC protects against spoofing, reordering, deletion
- Fragments are numbered - Seq# to protect against reordering, deletion
- Nonce's (both sides) protects against replay
- Certificate protect against MITM

Side Channels: unintentional information flow out of system that can potentially be exploited by an attacker
1)Timing Analysis: execution time of algorithms can sometimes leak important security information Ex. strcmp
- If strings not hashed so that get fixed size, loop terminates early
- Recommended constant or randomized runtime, independent of input
2)Electromagnetic Side-Channel: reconstruct monitor screen remotely or I/O devices
3)Power Analysis: power input to circuit produce unintended output that can leak information
- Monitor power consumption, reconstruct data lines based on power lines
4)Differential power analysis: similar but use capacitor instead of resistor

blockchain chaining, can't alter something in the past without altering every single block that follows it
- Nonce used for fairness in terms miners signing, nonce value sometimes has special rules need to pick a random nonce value that when hash everything in block (including the nonce value), needs to hash to something that has certain number of zeros on the end
- Signature: calculate hash of everything, then encrypt with private key. Anyone with public key can decrypt hash and verify

The physical implementation of a lock is less secure due to manufacturing methods (cost, material) results in parts not exactly precise, material damagable, and no input validation - unexpected inputs

BLP provide confidentiality, a subject can only read an object with the same or less security and can only create objects at the same or higher security.
BLP -> Read Down (less secure), Write Up (more secure)
BIBA integrity policy (trustworthiness/reliability of information). prevent low integrity information from affecting higher information integrity. subject can read higher integrity objects, but cannot create object with higher integrity.
BIBA -> Read Up, Write Down

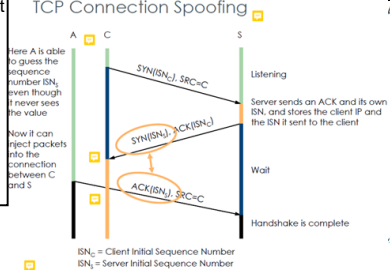
Kerberos model is divided into:
1) authentication server: authentication
2) ticket granting server: access control
3) service server
- trusted 3rd party, symmetric Needham Schroeder key exchange

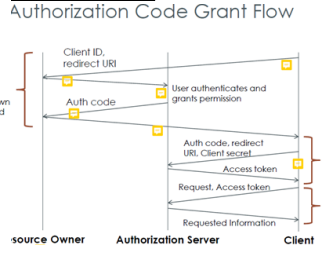
1) The idea is the authentication server is aware of user login credentials and handles authentication. If authentication is valid, the server returns a single use signed ticket that identifies the user and is valid/signed by the authentication server. Knows user passwords, provides authorization ticket
2) The authentication server ticket, after being passed to the client, can then be passed to the ticket granting server, which grants user access to services. The ticket granting server only cares about the ticket provided from the authentication server, and trusts a valid ticket means the user is authenticated. The ticket granting server then uses a database to see if the user has access to the service their requesting, and if so grants access to the service by returning a new single use ticket signed by the ticket granting server.
- provides tickets to user indicating is authorized to use service
3) The client can then provide the ticket from the ticket granting server to the service server to perform service.

Step1: AuthServ->C: TGT: [client Addr,validity,K(C,TGS)]_K(TGS), [K(C,TGS)]_K(C)
Step2: C->TGS: TGS: service, Step1 TGT Ticket_K(TGS)
Authenticator: [C, timestamp]_K(C,TGS)
Step3: TGS->C: Ticket(C,Service): service, [client, client addr, validity, K(C,service)]_K(Service), [K(C,service)]_K(C,TGS)
Step4: C->Service: Ticket(C,Service): from step3
Authenticator: [C, timestamp]_K(C,service)

Virus: avoid detection
- needs host program to infect (not-standalone app), by inserting instructions into existing programs
- could cause problems if same virus keeps infecting the same program, gets large. virus has signature that the author puts to check if already infected
- Beginning of program: Virus overwrites the start of the program, and then inserts some fixup code to replicate the code that it overwrote
- End of program: Virus overwrites first instruction of program with a jump to virus code at end, and then jumps back to the beginning of program. Virus only has to overwrite one instruction at the start of the program, and virus length is not limited.
Virus Detection: Virus scanners look for signatures: instructions found in known viruses, or common blocks of code
Polymorphic Viruses: small decryption engine + encrypted virus to avoid signature detection
- Encryption key changed on propagates, so encrypted body is never the same
- decryption engine constant short, hard to build signature
Defeating Polymorphic Viruses: To detect polymorphic viruses, virus scanners run files inside emulator, decrypt and scan
Metamorphic Viruses:
- change virus code when replicating by rewriting themselves (ie. machine instructions)
Worm: spreads quickly automatically, standalone app
Hit list scanning: Worm builder pre-seeds worms with hosts that are potentially vulnerable
- To Improve scanning speed for vulnerable machines to increase spreading speed
- infecting local hosts faster
- UDP no ACK, no wait for ACK, faster
Morris Worm operation: two parts (main server program, bootstrap/vector program). replicating creates shell on vulnerable target, uploads the vector, compiles on the target, then runs vector program
- vector downloads rest of worm from server, starts the server on newly infected host

TCP/IP protocol: for routing packets, 3-way handshake
TCP Connection Spoofing:
- attacker forge packet source IP set to client and guess ISN_c
- Forging source IP doesn't allow the attacker to receive packets
- However, if server's sequence number can be guessed, then attacker can connect to the server as the victim client
- Attacker pretending to be client, attacker is forging the address of the client, and putting the clients IP address as the return/source address
- with spoofing, attacker can send instructions but will not be able to see response
- unless attacker on same subnet, will not be able to see replies from server, they will be sent to client
- if attacker is able to acknowledge the sequence number that server sent to the client, then will be able to trick server into thinking is live connection, and start accepting whatever traffic that follows
TCP Reset Attack: Requires IP spoofing and guess ISN (terminates TCP connection (DoS) RST packet)
TCP SYN Flooding:
- Attacker sends many connection requests with spoofed IP source addresses
- Victim allocates resources for each request until some timeout
- Typically, OSs have a fixed bound on these half-open socket connections
- Eventually, half-open connection queue resource exhausted and no more requests are accepted (DoS)
SYN-ACK cookie:
- Server responds to client with SYN-ACK cookie
- ISNs = Cookie: HMAC(something unique to client + something unique and secret that server knows)
- ISNs = H(src addr, src port, dest addr, dest port, rand), rand 32-bit time counter
- server pick random sequence number, but not truly random
- server will wait till client comes back for 3rd part of handshake before allocating resources to stop attacker from exhausting resources
- When client performs 3rd part of handshake, client will pass hashed sync number created using a secret only the server knows, server can verify signature. When mesg comes in, server recalculate hash and check if hash matches, then server allocate resources and let the client connect.





Internet Control Message Protocol (ICMP):
- ICMP protocol is designed to help measure and route and deal with some basic utilities for low level management of networks (ie ping), subnet
ICMP Smurf Attack:
- generates lots of traffic by flooding victim with ICMP packets from many different machines. Attacker sends spoofed ICMP ping packet with victim as src and dest as broadcast address and could specify large packet size.
DoS