# CS 161: Computer Security Lecture Notes

## Week 1: Lecture 1 Introduction (8/25)

Learning Objectives

- How to think adversarially about computer systems
- How to assess threats for their significance
- How to build computer systems with robust security properties
- How to gauge the protections and limitations provided by today's tech
- How attacks work in practice

Memory Safety

Cryptography

Web Security

Network Security

Prisoner's Dilemma

- One round: always defect
- Multi Round: tit for tat

What is security?

- Enforcing a desired property in presence of an attacker
- Data confidentiality, user privacy, data and computation integrity

## Week 1: Lecture 2 Security Principles (8/27)

Threat model:

Threat Model: Common Assumptions for Attackers

- Can interact with systems without notice
- Knows general information about systems

Trusted Computing Base (TCB)

- Def: Components of  system that security relies upon
- Properties of TCB:
    - Completeness
- Small TCB

Consider Human Factors

Users

- Users like convenience (ease of use)
- Users subvert security systems if it makes their lives easier

Programmers

- Programmers make mistakes, use tools that make mistakes

Everyone else

- Social engineering attacks exploit other people's trust and access for personal gain
- Consider tools presented to users and make

Warning Dialogs

- Make it user friendly

<u>Security is Economics</u>

- Cost/Benefit analysis often appear in security
    - Cost of defense should be less than the cost of attacks happening
    - More security usually costs more
- Burglar Alarms
    - Too many false alarms

<u>Detect If You Can't Prevent</u>

Deterrence: Stop the attack before it happens

Prevention: Stop the attack as it happens

Detection: Learn that there was an attack (after it happened)

Reponse: mitigation and recovery

Have offsite backups for ransomware attacks

Defense in Depth

- Multiple types of defenses should be layered together
- Should have to breach all defenses to successfully attack a system

Example: Theodosian Walls of Constantinople

Consider the minimum permissions to do their job securely

- Access a specific folder
- Time of check to time of use

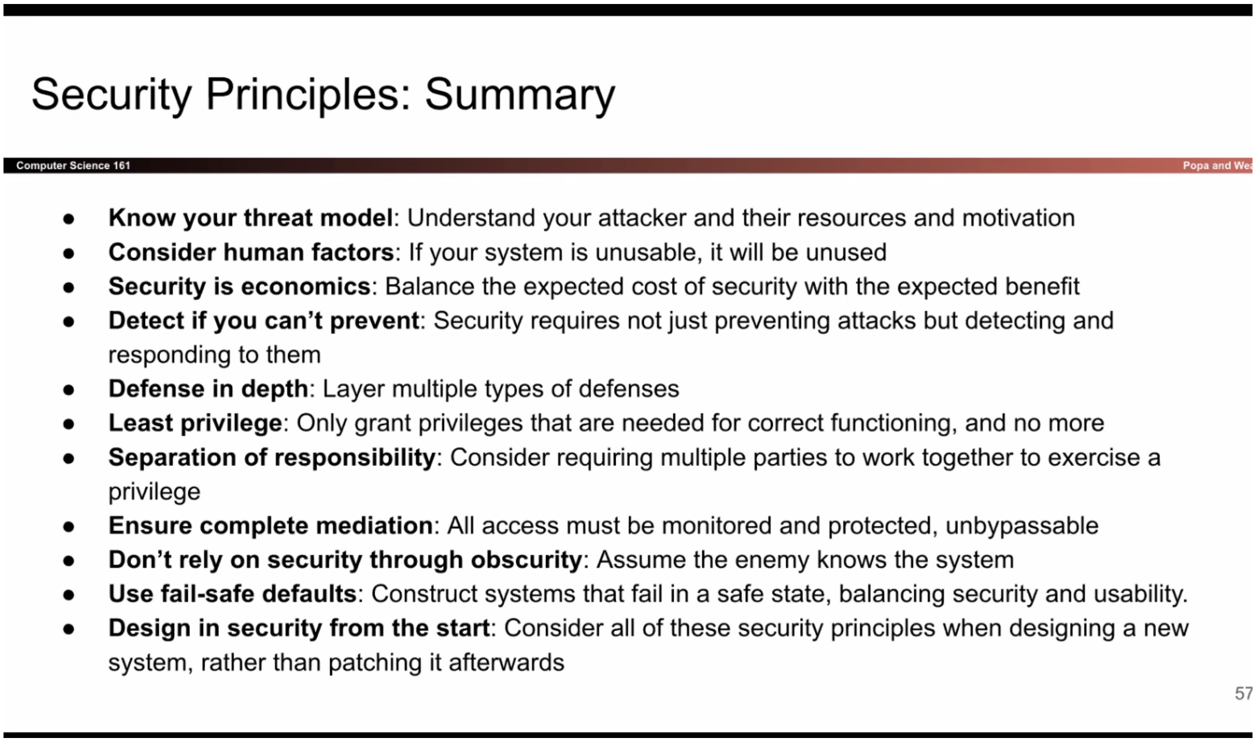
Don't rely on security through obscurity

What happens when the power is out

- Fail safe defaults

<u>Design in Security from the Start</u>

Include security as part of the design considerations rather than patching it after the fact



## Security Principles: Summary

Computer Science 161 Popa and Wea

- **Know your threat model**: Understand your attacker and their resources and motivation
- **Consider human factors**: If your system is unusable, it will be unused
- **Security is economics**: Balance the expected cost of security with the expected benefit
- **Detect if you can't prevent**: Security requires not just preventing attacks but detecting and responding to them
- **Defense in depth**: Layer multiple types of defenses
- **Least privilege**: Only grant privileges that are needed for correct functioning, and no more
- **Separation of responsibility**: Consider requiring multiple parties to work together to exercise a privilege
- **Ensure complete mediation**: All access must be monitored and protected, unbypassable
- **Don't rely on security through obscurity**: Assume the enemy knows the system
- **Use fail-safe defaults**: Construct systems that fail in a safe state, balancing security and usability.
- **Design in security from the start**: Consider all of these security principles when designing a new system, rather than patching it afterwards

57

## Week 2: Lecture 3 x86 Assembly and Call Stack (8/30)

Units of Measurement

- Bit: binary digit
- 4 bits = 1 nibble
- 8 bits = 1 byte

CALL (Compiler, Assembler, Linker, Loader)

- C Code -> Assembly code (RISC-V, x86) -> Machine code (raw bits)
- Compiler: Converts C code into assembly
- Assembler: Converts assembly code into machine code
- Linker, Loader

C Memory layout

- At runtime, loader tells your OS to give your program a big blob of memory
- Stack (grows downward), Heap (grows upward), data, code

X86 architecture

- Little Endian
    - Least significant byte or multi byte numbers is placed at the

x86 Registers

- Storage units as part of the CPU architecture
- General purpose registers
- Esp: stack pointer
- Ebp: base pointer

x86 Assembly

- Opcode, source destination

Stack Frames

- 0 local variables are always allocated on the stack
- Individual variables within a stack frame are stored with the first variable at the highest address
- Struct places on stack the other way around

Function calls

x86 Calling Convention

- Arguments pushed onto the stack in reverse order, val3 placed at the highest memory address, val2, then val1
- How to receive return values
    - Return values are passed in eax
    - Similar to risc-v which passes return values in a0-a1
    - Which registers are caller-saved or callee-saved
        - Callee-saved: callee must
- Caller creates new part on stack

Steps of an x86 function call

1. Push arguments on the stack
2. Push old EIP on the stack
3. Move eip
4. Push old ebp on the stack
5. Move ebp
6. Move esp
7. Execute the function
8. Move esp
9. Move rip
10. Remove arguments from stack
    a. Increment esp to delete the arguments from the stack

## Week 2: Lecture 4 Memory Safety Vulnerabilities (9/1)

Review Instructions

- Push src:
    - Esp moves one word down
    - Puts value in src at the current esp
- Pop dst
    - Copies lowest value on the stack into dst
    - Esp moves one word up
- Mov src dst
    - Copies src into dst

Saved Values on the stack

- Sfp (saved frame pointer) : When callee saves the value of ebp on the stack,
- Rip (returned instruction pointer): when callee saves the value of eip on the stack

Buffer Overflow

- Attackers can exploit lack of two boundaries to control areas (memory) that they aren't supposed to control
- Buffer overflow to set authenticated variable to 1

Out of bounds writes

- One of most dangerous exploits still

Stack Smashing

- Most common kind of buffer overflow

Python Syntax

- Raw bytes \x interpret it in hex
    - len('\xff') == 1
- Characters can be represented as bytes too
    - '\x41' == 'A'

Overwriting the rip

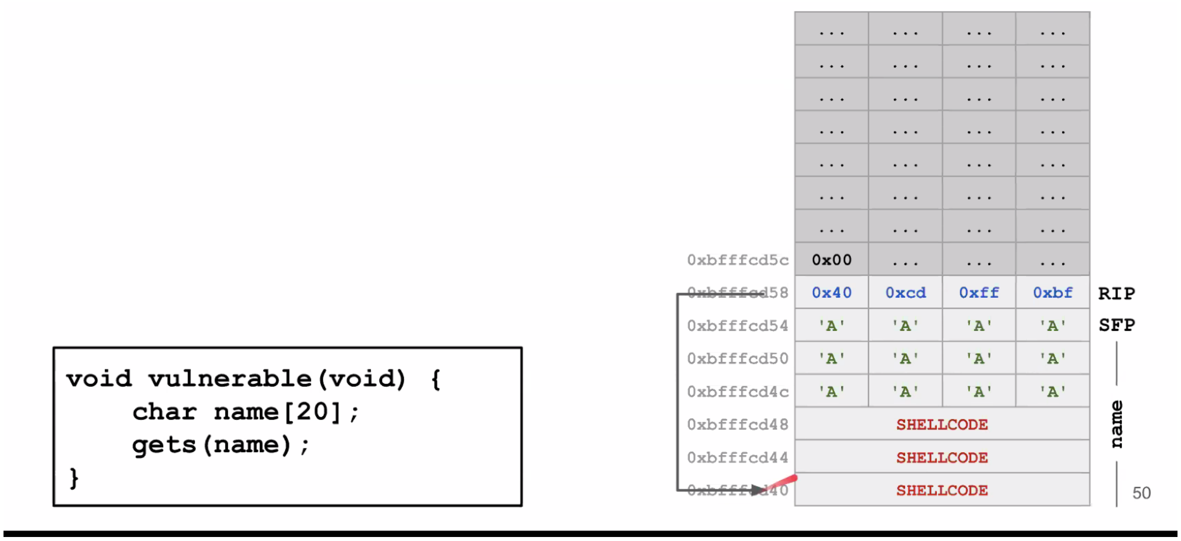
- Gets starts writing and can overwrite anything above name, including the rip

Writing malicious Code

- Place code in memory yourself
- Shellcode: malicious code

Putting together an attack

1. Find a memory safety vulnerability
2. Write malicious shellcode at a known memory address
3. Overwrite ther ip with the address of the shellcode
4. Return front eh function
5. Begin executing malicious shellcode



```
void vulnerable(void) {
    char name[20];
    gets(name);
}
```

| | | | | | |
|---|---|---|---|---|---|
| | ... | ... | ... | ... | |
| | ... | ... | ... | ... | |
| | ... | ... | ... | ... | |
| | ... | ... | ... | ... | |
| | ... | ... | ... | ... | |
| | ... | ... | ... | ... | |
| | ... | ... | ... | ... | |
| 0xbfffcd5c | 0x00 | ... | ... | ... | |
| 0xbfffcd58 | 0x40 | 0xcd | 0xff | 0xbf | RIP |
| 0xbfffcd54 | 'A' | 'A' | 'A' | 'A' | SFP |
| 0xbfffcd50 | 'A' | 'A' | 'A' | 'A' | name |
| 0xbfffcd4c | 'A' | 'A' | 'A' | 'A' | |
| 0xbfffcd48 | SHELLCODE | | | | |
| 0xbfffcd44 | SHELLCODE | | | | |
| 0xbfffcd40 | SHELLCODE | | | | |

Shellcode: spawns a shell that you can use

Can also put code above shellcode

SUmmary

- Buffer overflows: overwrite unintended parts of memory
- Stack smashing: an attacker overwrites saved registers on the stack

## Week 2: Lecture 5 Memory Safety Vulnerabilities Cont. (9/3)

Solution

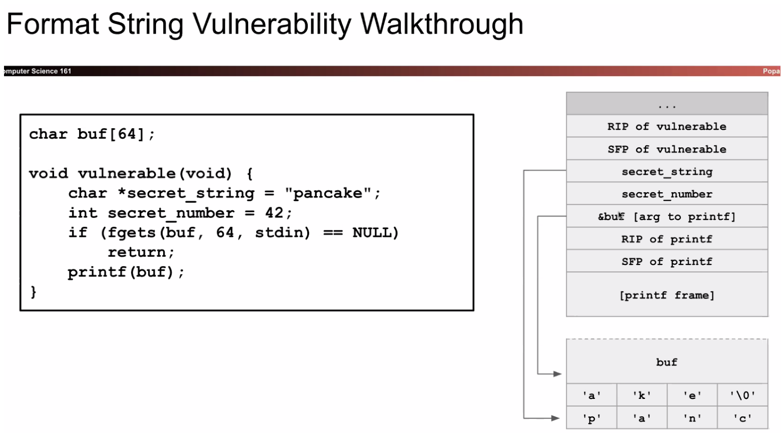
- Specify the size of gets, strcpy
- Use
    - fgets instead of gets
    - Strncpy or strlcpy instead of strcpy
    - strnlen instead of strlen

Possible safety issue

- Is input len and do len + 2, it can overflow
- Have a checker to make sure (len > SIZE_MAX - 2)

Printf behavior

- Printf looks at the argument above it
- Attacker can specify any format string they want and view what the rip is
- Example
    - Input: %d%s
    - Can print out the secret number and secret string



Format String Vulnerability Walkthrough

```
char buf[64];

void vulnerable(void) {
    char *secret_string = "pancake";
    int secret_number = 42;
    if (fgets(buf, 64, stdin) == NULL)
        return;
    printf(buf);
}
```

- %n treats next argument as a pointer and writes the number of bytes printed to that address
- printf("000%n) writes 3 into the pointer above

Printf Defense

- Never use an untrusted input into the first argument of printf

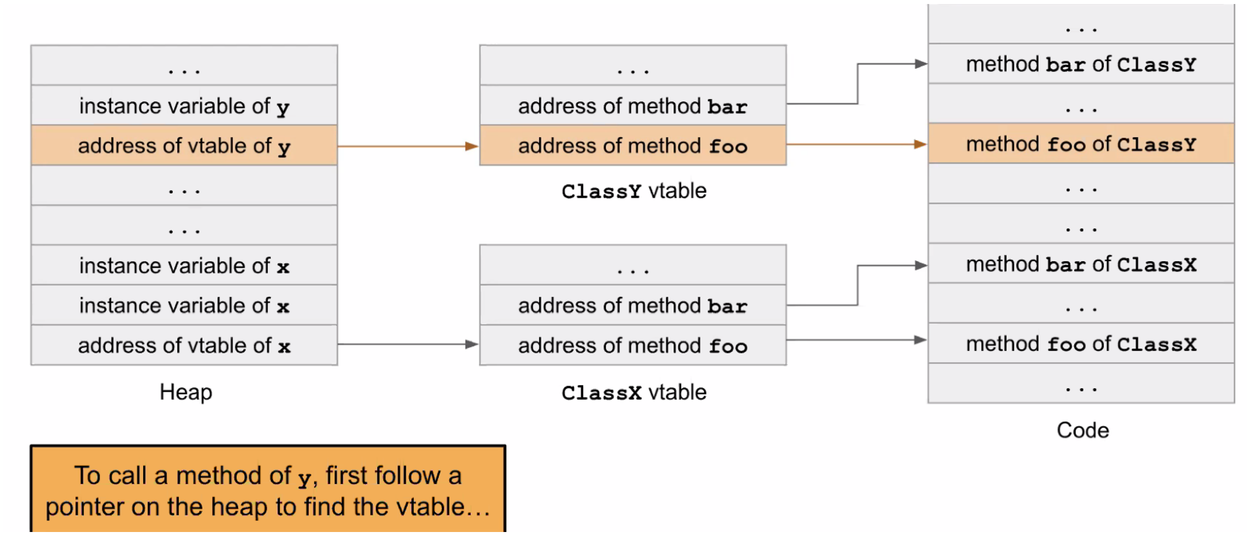
## Week 3: Lecture 6 Memory Safety Vulnerabilities Cont. and Mitigations (9/8)

Review: Memory Safety Vulnerability

- Format string Vulnerabilities
- Read format string from user but attacker can provide malicious format string

Heap Vulnerabilities

- C++ is Object oriented language
- Each class has a vtable (table of function pointers)



. . .
instance variable of **y**
address of vtable of **y**
. . .
. . .
instance variable of **x**
instance variable of **x**
address of vtable of **x**
Heap

. . .
address of method **bar**
address of method **foo**
**ClassY** vtable

. . .
address of method **bar**
address of method **foo**
**ClassX** vtable

. . .
method **bar** of **ClassY**
. . .
method **foo** of **ClassY**
. . .
. . .
method **bar** of **ClassX**
. . .
method **foo** of **ClassX**
. . .
Code

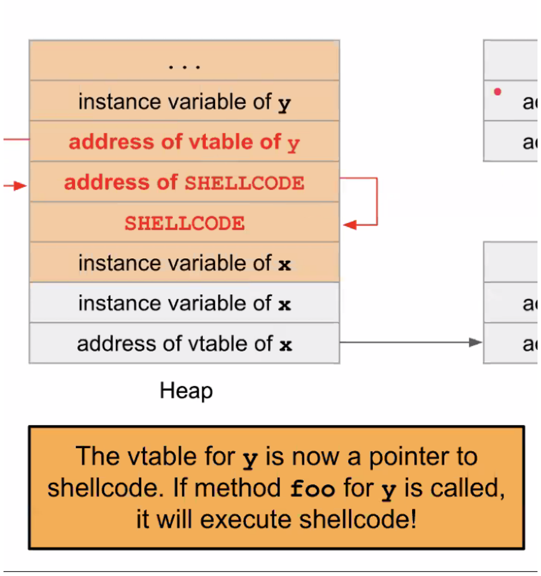
To call a method of **y**, first follow a pointer on the heap to find the vtable…

Heap overflow

- Objects allocated in the heap
- A write to a buffer in heap is not checked
- Attacker overflows the buffer and overwrites the vtable pointer of the next object to point to a malicious vtable, with pointers to malicious code

Use-after-free

- Object deallocated too early
    - Attacker allocates memory, which restaurant eh memory freed by the struct



. . .
instance variable of **y**
address of vtable of **y**
address of SHELLCODE
SHELLCODE
instance variable of **x**
instance variable of **x**
address of vtable of **x**
Heap

The vtable for **y** is now a pointer to shellcode. If method **foo** for **y** is called, it will execute shellcode!

Variance in Exploits

- Exploits can often be very brittle
    - Depends on OS
    - Needs to align to the right place otherwise crashes the computer

NOP Sleds

- Have a lot of nops so that landing anywhere in the sled will bring you to your shellcode
- Allow you to just get close enough to your code

Memory Safety Mitigations

- Memorys afe languages
- Writing memory safe code
- Building secure software

Defending against emm roy safety vulnerabilities

- Programming languages aren't designed well for security
- Aren't security aware,
- Humans make mistakes

Use Memory safe languages

- Designed to check bounds and prevent undefined memory access

Why use None-memory safe languages

- Most common reason
    - Performance
    - Malloc usually runs in amortized constant time
    - Java: memory safe: garbage collect may need to run at any arbitrary point in time, adding 10-100 ms delay
- Myth of Performance
    - Safer languages were slower,
    - Safe alternatives have comparable performance (go and rust)

Real reason: legacy

- Inertia and legacy
- Huge existing code bases are written in C: building on existing code is easier than starting from scratch
- 2014: swift new memory=safe languages

Learn to write memory-safe code

- Always add checks in yoru code just in case
- Always check pointer is not null before dereferencing it, even if you're sure the pointer is going to be valid
- Use safe libraries
    - Functions that check bounds
- Structure user input
    - Constrain how untrusted sources can interact with the system
- Reason carefully about your code
    - When writing code, define set of preconditions

1. Building secure software

- Use tools for analyzing and patching code
- Run time checks for automatic bounds-checking

- Monitor code for run-time misbehavior
    - Look for illegal calling sequences never call execve
- Contain potential damage
- run system components in sandboxes or virtual machines
- Privilege separation

2. Approaches for building secure software/systems

- Bug finding tools
- Code review
- Vulnerability scanning
- Penetration testing

Modern software often import lost of different libraries

- Libraries often updated with security patches
- Keep libraries updated with latest security patches

Vulternatibilties become exploits quickly, crucial to patch asap

3. Add mitigations that make it harder to exploit common vulnerabilities

- Exploit mitigations (code hardening): compiler and runtime defenses that make common exploits harder
    - Make exploits into program crashes, crashing safer than exploits
- Arms race

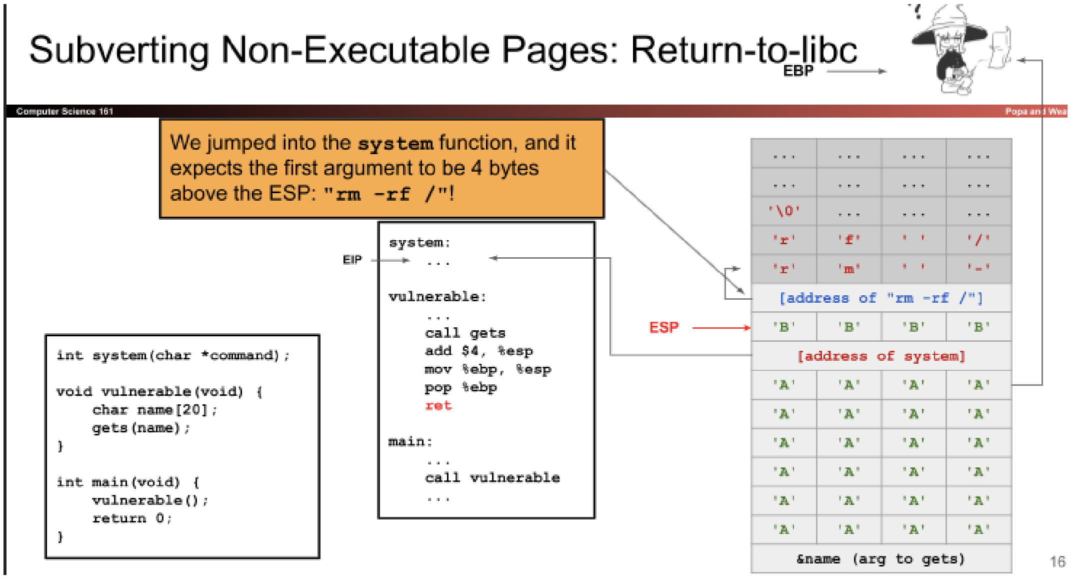
## Week 3: Lecture 7 Memory Safety Vulnerabilities Cont. and Mitigations (9/10)

Mitigation: Non-Executable Pages

- prevent begin malicious shellcode
- most programs don't need memory that is both written to and executed so make portions of memory either executable or writable but not both
- DO it by: Page table entries have a writable bit and executable bit
- W^X (Write XOR execute)
- No execute bit
- DEP (Data Execution Prevention, named used by Windows)

Subverting Non-executable Pages

- Non-executable pages doesn't prevent an attacker from leveraging existing code in memory as part of the exploit
- Functions that are in code can cause issues
    - Return to libc: overwrite the rip to jump to a function in standard C library
    - Return oriented programming: constructing custom shellcode using pieces of code already exist in memory
- Libc function
- Attacker can leverage code already in lib c



Subverting Non-Executable Pages: Return-to-libc

Computer Science 161

Popa and Wea

We jumped into the **system** function, and it expects the first argument to be 4 bytes above the ESP: **"rm -rf /"**!

```
int system(char *command);

void vulnerable(void) {
    char name[20];
    gets(name);
}

int main(void) {
    vulnerable();
    return 0;
}
```

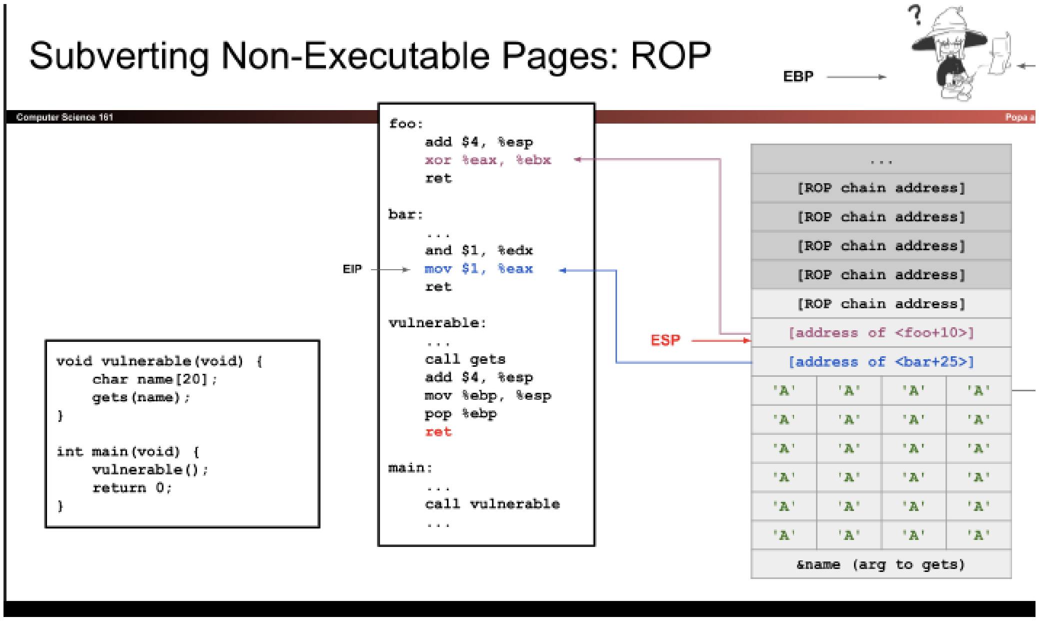
```
system:
    ...

vulnerable:
    ...
    call gets
    add $4, %esp
    mov %ebp, %esp
    pop %ebp
    ret

main:
    ...
    call vulnerable
    ...
```

| | | | |
|---|---|---|---|
| ... | ... | ... | ... |
| ... | ... | ... | ... |
| '\0' | ... | ... | ... |
| 'r' | 'f' | ' ' | '/' |
| 'r' | 'm' | ' ' | '-' |
| [address of "rm -rf /"] | | | |
| 'B' | 'B' | 'B' | 'B' |
| [address of system] | | | |
| 'A' | 'A' | 'A' | 'A' |
| 'A' | 'A' | 'A' | 'A' |
| 'A' | 'A' | 'A' | 'A' |
| 'A' | 'A' | 'A' | 'A' |
| 'A' | 'A' | 'A' | 'A' |
| 'A' | 'A' | 'A' | 'A' |
| &name (arg to gets) | | | |

16

## Subverting Non-Executable Pages: ROP

- Gadget: A small set of assembly instructions that already exist in memory
    - End in ret instruction
    - Not usually full functions
    -



Subverting Non-Executable Pages: ROP

Computer Science 161

Popa a

```
void vulnerable(void) {
    char name[20];
    gets(name);
}

int main(void) {
    vulnerable();
    return 0;
}
```

```
foo:
    add $4, %esp
    xor %eax, %ebx
    ret

bar:
    ...
    and $1, %edx
    mov $1, %eax
    ret

vulnerable:
    ...
    call gets
    add $4, %esp
    mov %ebp, %esp
    pop %ebp
    ret

main:
    ...
    call vulnerable
    ...
```

| | | | |
|---|---|---|---|
| ... | | | |
| [ROP chain address] | | | |
| [ROP chain address] | | | |
| [ROP chain address] | | | |
| [ROP chain address] | | | |
| [ROP chain address] | | | |
| [address of <foo+10>] | | | |
| [address of <bar+25>] | | | |
| 'A' | 'A' | 'A' | 'A' |
| 'A' | 'A' | 'A' | 'A' |
| 'A' | 'A' | 'A' | 'A' |
| 'A' | 'A' | 'A' | 'A' |
| 'A' | 'A' | 'A' | 'A' |
| 'A' | 'A' | 'A' | 'A' |
| &name (arg to gets) | | | |

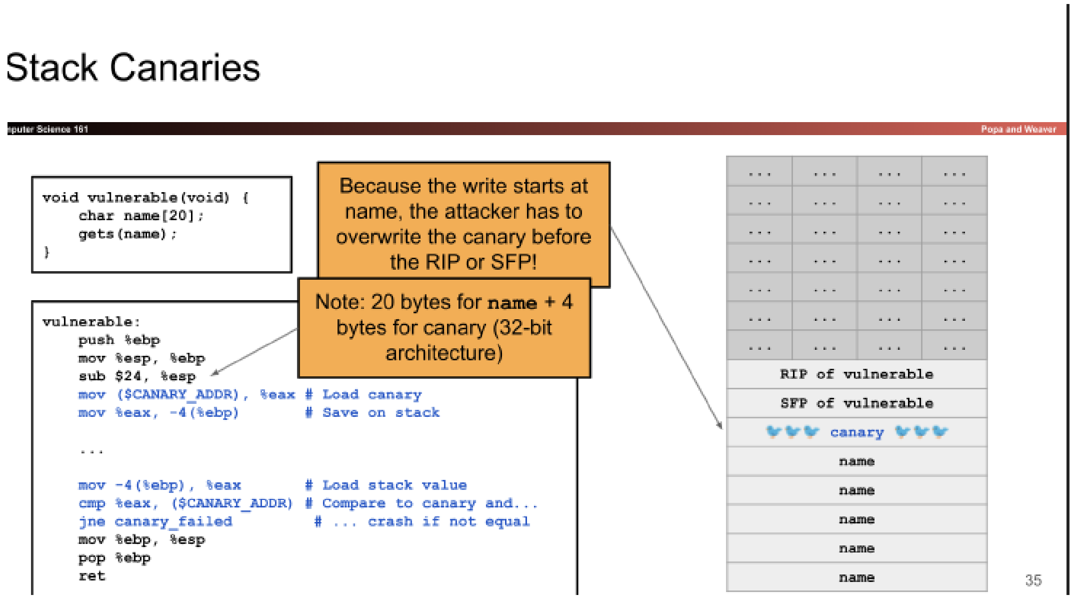
## Subverting Non-executable Pages: ROP

- ROP compilers available
- Non-executable pages not a huge issue for attackers nowadays

## Mitigation: Stack Canaries

- Prevent overwrite the rip with the address of the shell code
- Idea: add a sacrificial value on the stack and check if it has been changed
    - Generate a random secret value and save it in the canary storage

    - Place canary value on the stack below the SFP/RIP
    - In epilog, check the value on stack and compare it against the value in canary storage



## Stack Canaries

puter Science 161 Popa and Weaver

```
void vulnerable(void) {
    char name[20];
    gets(name);
}
```

```
vulnerable:
    push %ebp
    mov %esp, %ebp
    sub $24, %esp
    mov ($CANARY_ADDR), %eax # Load canary
    mov %eax, -4(%ebp)       # Save on stack

    ...

    mov -4(%ebp), %eax       # Load stack value
    cmp %eax, ($CANARY_ADDR) # Compare to canary and...
    jne canary_failed         # ... crash if not equal
    mov %ebp, %esp
    pop %ebp
    ret
```

Because the write starts at name, the attacker has to overwrite the canary before the RIP or SFP!

Note: 20 bytes for **name** + 4 bytes for canary (32-bit architecture)

| | | | |
|---|---|---|---|
| ... | ... | ... | ... |
| ... | ... | ... | ... |
| ... | ... | ... | ... |
| ... | ... | ... | ... |
| ... | ... | ... | ... |
| ... | ... | ... | ... |
| ... | ... | ... | ... |
| RIP of vulnerable | | | |
| SFP of vulnerable | | | |
| 🐦🐦🐦 canary 🐦🐦🐦 | | | |
| name | | | |
| name | | | |
| name | | | |
| name | | | |
| name | | | |

35

Cheap way to stop lots of common attacks

Subverting Stack canaries

- Leak the value of the canary: overwrite the canary with itself
- Bypass the value of the canary: use a random write, not a sequential write
- Guess the value of the canary: brute force

Leak the canary

- Any vulnerability that leaks stack memory can be used to leak the canary value
- When you learnt the value of the stack canary, place it in the exploit st canary overwritten with itself

Bypassing the canary

- can stop attacks that write to increasing, consecutive addresses on the stack
- Cannot stop write around the canary
    - Heap overflow
    - Format string vulnerability
    - c++ viable exploits

Guessing the canary

- less effective on 32 bit systems
- How running the program, if the program has a timeout

Mitigation: Pointer Authentication

- Mitigate Override the rip with address of the shell code
- Instead of placing the secret value below the pointer, store a value in the pointer itself, use unused bits in a 64 bit address to store a secret value

- Replace unused bits with a pointer authentication code (PAC)
- CHeck if pac is still valid

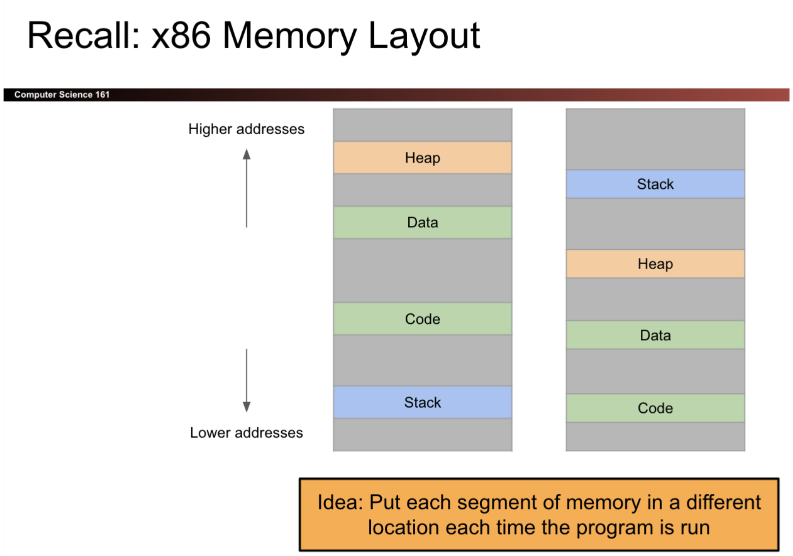
PAC Properties

- Each possible address has its own PAC
- Only someone who knows CPUs master secret can generate a PAC for an address
-

## Week 4: Lecture 8 Cryptography (9/13)

Address Space Layout Randomization

- Mitigate write malicious shellcode at known memory address
- Put each segment of memory in a different location each time the program is run
- Can shuffle all 4 parts of memory



Recall: x86 Memory Layout

Computer Science 161

Idea: Put each segment of memory in a different location each time the program is run

Subverting ASLR

- Leak a pointer to an address
- Brute force and search through address space

Combining Mitigations

- Use multiple mitigations together
- Combine ALSR and non-executable pages
- Attacker would need to find 2 vulnerabilities
- Many are insignificant against performance

Summary: memory safety mitigations

- Mitigation: pointer authentication
- Mitigation: address space layout randomization
- Combining authentications

Cryptography

- Provide rigorous guarantees on the security of data and computation in the presence of an attacker
- Eve: eavesdropping
- Mallory: manipulator

THree goals of cryptography

1. Confidentiality: An adversary cannot read our messages
2. Integrity: an adversary cannot change our messages without being detected
3. Authenticity: prove that this message came from the person who claims to have written it

Building block of any cryptographic scheme

- The key
- Use the key in our algorithms to secure messages
    - Symmetric key model:
        - Both know the value of the same secret key
    - Asymmetric key model:
        - everyone has two keys, a secret key and a public key

Kerckhoff's principle

- Principle is closely related to shannon's Maxim: don't use security through obscurity

Confidentiality

- Confidentiality: An adversary cannot read our messages
- Plaintext: the original message
- Ciphertext: encrypted message

## Week 4: Lecture 9 Symmetric-Key Cryptography (9/15)

Cryptography by computers

- Claude shannon
- Started with paper and pencil algorithms
- Moved to machines,

Confidentiality

- Ciphertext should not give the attacker any additional information
- Two time pads
    - Can learn if have multiple keys

Impracticality of one time pads

1. Key Generation
    a. Keys must be randomly generated for every message
2. Key

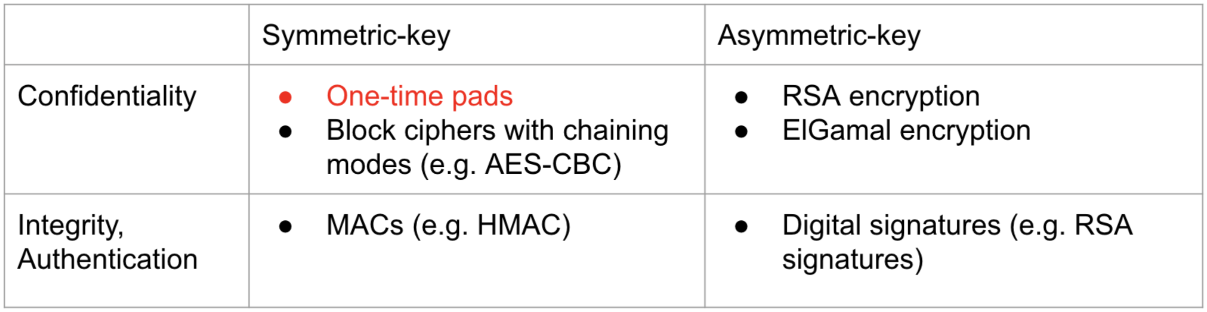
## Week 4: Lecture 10 Symmetric-Key Cryptography (9/17)

One time pad

- Can only use the key once

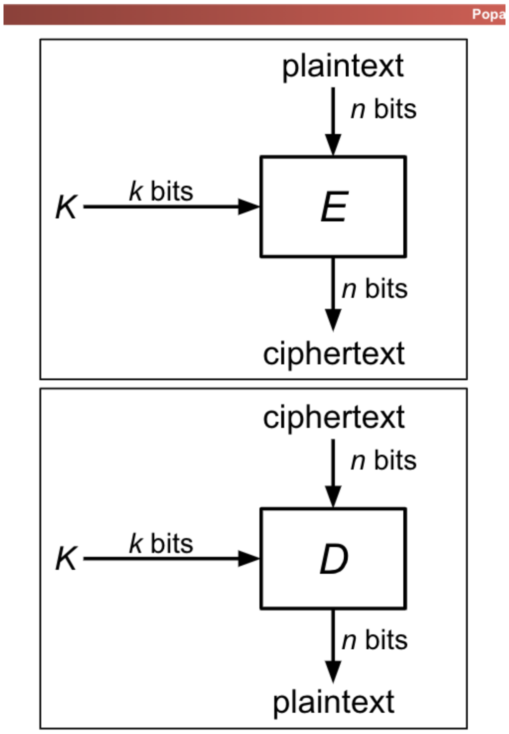
Block Ciphers

- Build integrity, useful



| | Symmetric-key | Asymmetric-key |
|---|---|---|
| Confidentiality | ● One-time pads<br>● Block ciphers with chaining modes (e.g. AES-CBC) | ● RSA encryption<br>● ElGamal encryption |
| Integrity, Authentication | ● MACs (e.g. HMAC) | ● Digital signatures (e.g. RSA signatures) |

-



Block Cipher: an encryption/decryption algorithm that encrypts a fixed-sized block of bits

- $E_k(M) \rightarrow C$: Encryption
  - Inputs k bit key K and n bit plaintext M
  - Output: An n bit ciphertext
- $E_k(M)$ must be a permutation (bijective function) on n bit strings

Secure block cipher behaves like a randomly chosen permutation from set of all permutation on n bit strings

- Brute force attacks on modernism block ciphers are not possible assuming the key is random and secret

Efficiency

- Encryption and decryption should be computable in microseconds
- Typically use XOR and bit shifting

Advanced Encryption Standard) AES

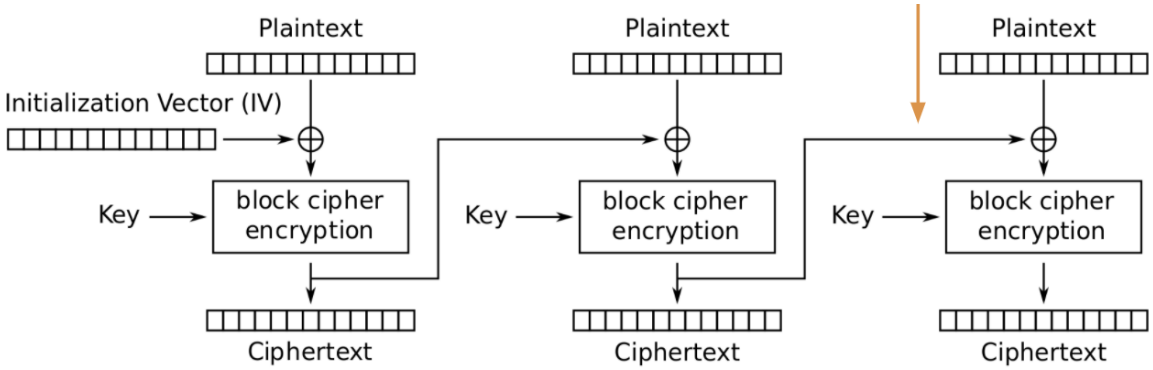
- Twofish was designed by David Wagner
- Rijindael was always second fastest
- Key size 128 block size 128
- Shift rows

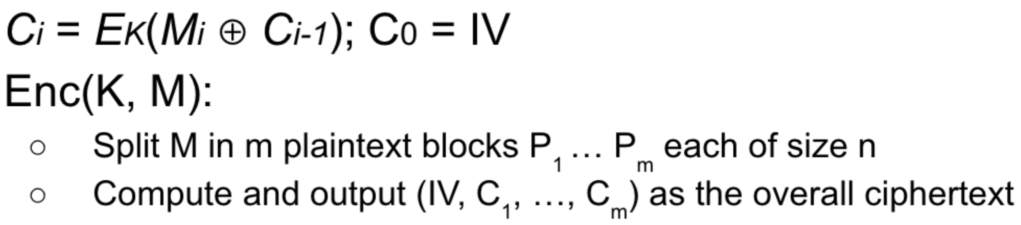
Block Ciphers: Summary

- Correctness: when key is fixed Ek(M) should be bijective
- Security: without the key, E(m) computationally indistinguishable from a random permutation
- Efficiency: algorithms
- Can only encrypt messages of a fixed size
- deterministic

CBC Mode

- Cipher block chaining mode



-


$C_i = E_K(M_i \oplus C_{i-1})$; $C_0 = IV$

Enc(K, M):

- Split M in m plaintext blocks $P_1 \ldots P_m$ each of size n
- Compute and output ($IV, C_1, \ldots, C_m$) as the overall ciphertext

CBC Mode: Efficiency & Parallelism

- E is sequential
- Description is parallel

## Week 5: Lecture 11 Cryptographic Hashes (9/20)

CBC Mode is IND-CPA

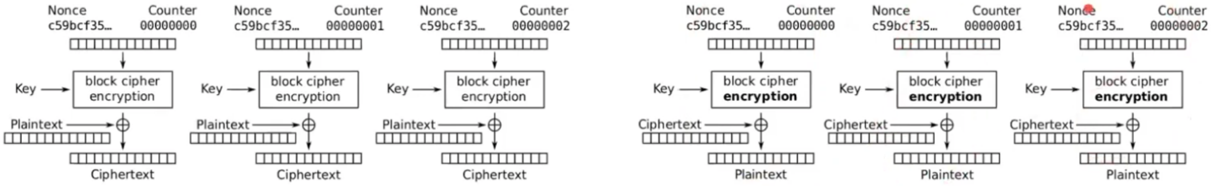
- Need to chain block ciphers

Why IND-CPA

- If there exists an attacker that wins in the IND-CPA Game there exists an attacker that breaks the block cipher security

CTR Mode

- One time pads are useful if key is only used once
- Efficiency
    - Encryption can be parallelized
    - Decryption can be parallelized
- Padding
    - Canning no necessary, can just cut off parts of XOR that are longer
    -



Counter (CTR) mode encryption

Counter (CTR) mode decryption

        -
        - AES-CTR is IND-CPA secure

IVs and Nonces

- Initialization vector (IV): random, but public, one use value to introduce randomness into algorithm
- Never reuse IVs

Comparing modes of Operation

- High performance, which mode is better
    - CTR
- If you're paranoid about security which mode is better
    - CBC is better
    - Used improperly CTR fails catastrophically

Block Cipher Modes of Operation:

- ECB MOde: deterministic, not IND-CPA secure
- CBC mode
    - IND-CPA secure
    - Encryption is not parallelizable
    - Decryption is parallelizable
    - Must pad plaintext to a multiple of the block size
    - IV reuse leads to leaking the existence of identical blocks at the start of the message
- CTR mode
    - IND-CPA secure, no IV reuse
    - Encryption and description are parallelizable
    - Plaintext do not need to be padded
    - Nonce reuse leads to losing all security

Hash Function

- Hash function provides a fixed-length "fingerprint" over a sequence of bits
- Document comparison
    - 1 GB comparison
- Given output y it is infeasible to find any input x such that H(x) = y
-

## Week 5: Lecture 12 Message Authentication Codes (MACs) (9/22)

Hash Function: Collision Resistance

- Collision: Two different inputs with the same output

Hash Function

- MD5
    - 128 bits
    - Security: Broken
- SHA-1
    - 160 bits
    - Broken 2017
- SHA2
    - 256, 384, 512 bits
    - Not currently broken vulnerable to length extension attack
- SHA3
    - 256, 384, 512

Length extension attack

- Given H(x) and length of x and not x an attack can create H(x||m) for any m of the attacker's choosing

SHA 256 bit version vulnerable

SHA 3 is not vulnerable

Do hashes provide integrity

MACs: Definition

- Two parts
    - keygen() -> K Generate a key K
    - MAC(K, M) -> T: Generate a tag T for the message M using key K
        - Secret key and an arbitrary length message
        - Output: a fixed length tag on the message
    - Properties
        - Correctness: determinism
        - Efficiency: Computing a mac should be efficiency
        - Security : EU-CPA
- Defining integrity
    1. EU-CPA
        a. Mallory may send messages to Alice and receive their tags
        b. Eventually Mallory creates a message-tag mapir (M', T') M' cannot be a message that mallory requested earlier
        c. If T' is a valid tag for M' then Mallory wins, otherwise she's loses
        d. Scheme is EU-CPA secure if for all polynomial time adversaries, the probability of winning is 0 or negligible
    2. $K, M = H((K \oplus opad))$

- Do MACs provide integrity
    - Yes An attacker cannot tamper with the message without being detected
- Do MACs provide authenticity
    - Depends on threat model
    - If a message has a valid MAC can be sure it came from someone with secret key

Authenticated encryption

Key Reuse

- Using the same key in two different algorithms

## Week 5: Lecture 13 Pseudorandom Number Generators (PRNGs), Steam Ciphers and Diffie-Hellman Key Exchange (9/24)

MAC

- input : a secret key and a message
- Output: a tag on the message

Authenticated Encryption

- Authenticated encryption: scheme that guarantees confidentiality and integrity (and authenticity)
- Always encrypt then MAC
- Side channel attack

AEAD Encryption

- Second Method for authenticated encryption
    - Use a scheme that is designed to provide confidentiality, integrity and authenticity
- Authenticated encryption with additional data (AEAD)
    - Algorithm that provides both confidentiality and integrity over plaintext and integrity over additional data

Authenticated Encryption Summary

1. Combine schemes that provide confidentiality with schemes that provide integrity and authenticity
    a. Encrypt and then MAC: $Enc(K_1, M) \,||\, MAC(K_2, Enc(K_1, M))$
2. Use AEAD Encryption modes designed to provide confidentiality, integrity, and authenticity

Pseudorandom Number GEnerators (PRNGS)

Entropy

- A measure of uncertainty
- High entropy = unpredictable outcomes = desirable in cryptography

Pseudorandom Number Generators (PRNGs)

- Algorithm that uses a little bit of true randomness to generate a lot of random -looking output

PRNG

- PRNG.Seed(randomness): Initializes the internal state using the entropy
- PRNG.Generate(n) generate n pseudorandom bits
    - input : some truly random bits
    - output : n pseudorandom bits
- Properties
    - Correctness: deterministic
    - Efficiency: efficient to generate pseudorandom bits
    - Security: indistinguishability from random

PRNG: security

- Secure PRNG cannot be truly random
- Computationally indistinguishable from random to an attacker

Universally Unique Identifiers (UUIDs)

- Set of objects want to have a unique name,
- If enough randomness generating a random number will give you a unique name

Steam Ciphers

- Another way to construct symmetric key encryption schemes
- enc(K, M)
- Choose a random value IV
- PRG. SEED (K|IV)
- C = Prg. generate (|M|) XOR M
- Output (IV, C)

Diffie-Hellman

- Asymmetric -key
- Option 1:
    - Alice lock box, send to bob
    - Bob also lock box, send to alice
    - Alice unlock her lock, send to bob
    - Bob unlock his lock, unlock
- Option 2
    - Bob and alice his lock
    - Alice locks down msg with bobs lock

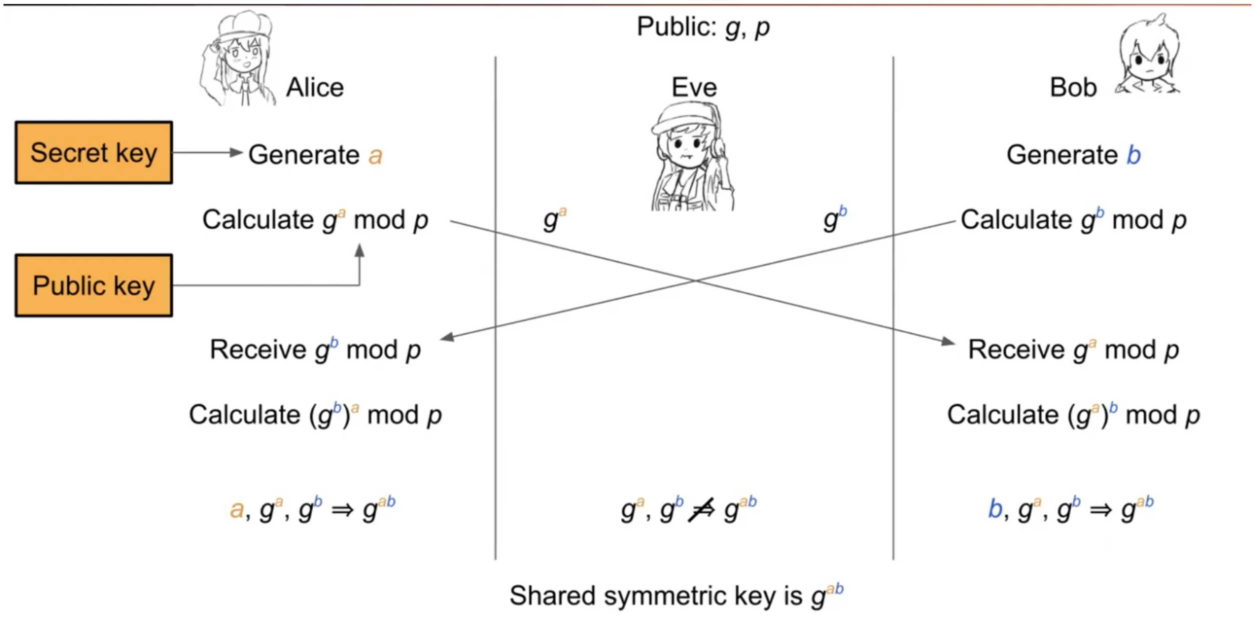
## <u>Week 6: Lecture 14 Diffie-Hellman Key Exchange  (9/27)</u>

Public Key Exchange

Discrete Log Problem and Diffie-Hellman Problem

- Discrete Logarithm Problem (discrete log problem) : Give g, p, $g^a \bmod p$ it is computationally hard to find $a$

- Diffie-Hellman problem: Given $g,\ p,\ g^{a}\ mod\ p$and $g^{b}$mod $p$, computationally hard to find $g^{ab}$mod $p$

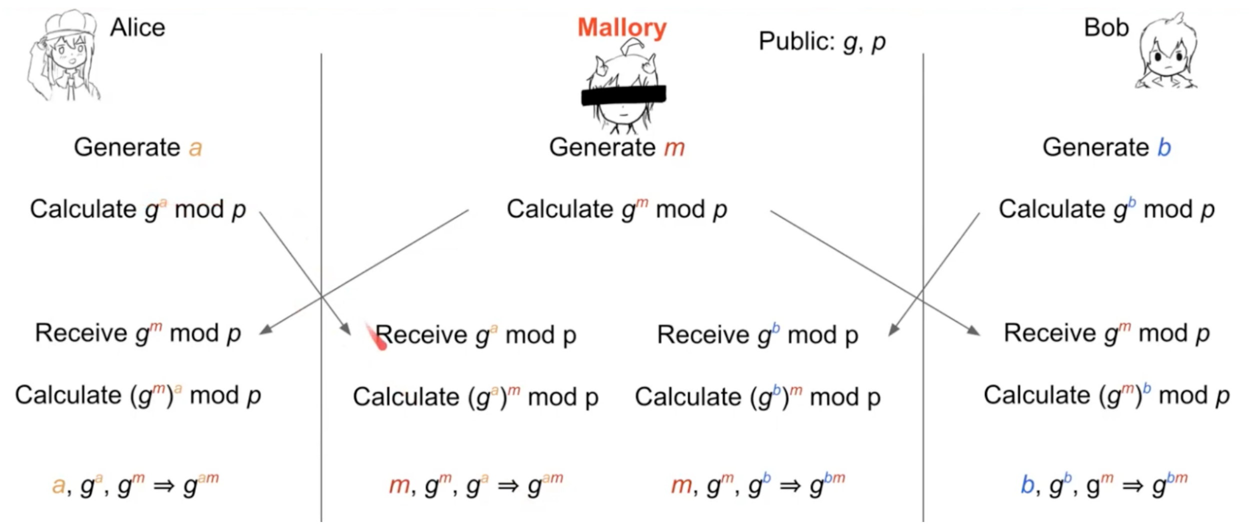


Ephemerality of Diffie-Hellman

- Can be used ephemerally
- Ephemeral: Short-term and temporary, not permanent
- Alice and Bob discard a, b, and $K\ = g^{ab}mod\ p$
- K is a session key, because it’s only used for a an ephemeral session
- Benefit of DHE: Forward Secrecy
    - Eve records everything sent over the insecure channel
    - Alice and Bob use DHE to agree on a a key K
    - After they’re done, discard a, b , and K ‘

What Can Mallory doe

- Mallory can alter messages


Alice
Mallory
Public: $g$, $p$
Bob
Generate $a$
Generate $m$
Generate $b$
Calculate $g^a$ mod $p$
Calculate $g^m$ mod $p$
Calculate $g^b$ mod $p$
Receive $g^m$ mod $p$
Receive $g^a$ mod p
Receive $g^b$ mod p
Receive $g^m$ mod $p$
Calculate $(g^m)^a$ mod $p$
Calculate $(g^a)^m$ mod p
Calculate $(g^b)^m$ mod p
Calculate $(g^m)^b$ mod $p$
$a, g^a, g^m \Rightarrow g^{am}$
$m, g^m, g^a \Rightarrow g^{am}$
$m, g^m, g^b \Rightarrow g^{bm}$
$b, g^b, g^m \Rightarrow g^{bm}$

-

Diffie-Hellman: Issues

- Not secure against a MITM adversary
- DHE is an active protocol: Alice and Bob need to be online at the same time
- Does not provide authentication

Public Key Cryptography

Public key schemes, each person has two keys

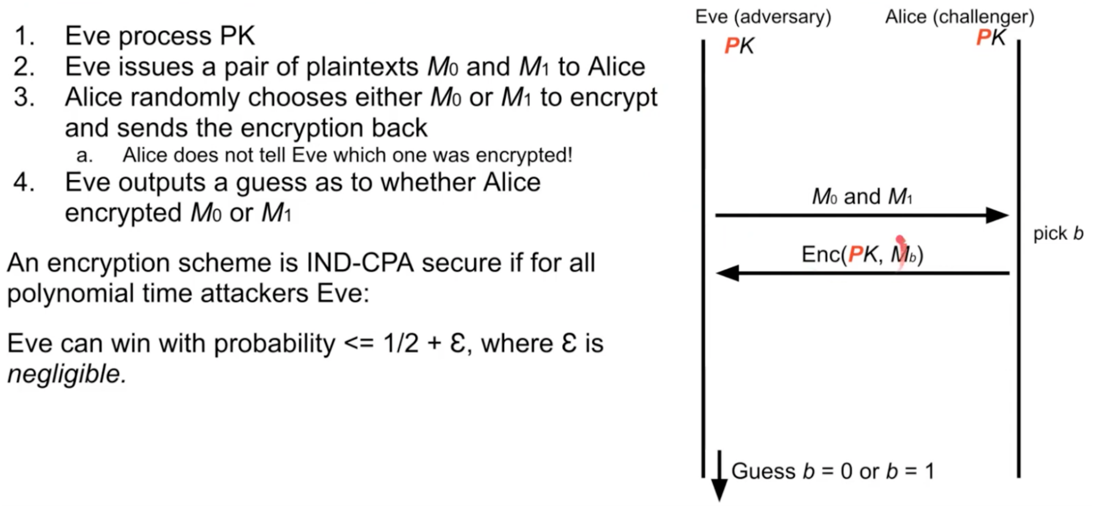
- Public key: known to everybody
- Private key: only known by that person
- Keys come in pairs: every public key corresponds to one private key
- Messages are bit strings
- Benefit: no longer need to assume that Alice and Bob already share secret

Public-Key Encryption: Definition

- KeyGen() -> PK, SK, generate a public/private keypair where PK is public key, and SK is private secret key
- Enc(PK, M) -> C: Encrypt a plaintext M using public key PK to produce ciphertext C
- Dec(SK, C) -> M: Decrypt a cipher text C using secret key SK
- Properties
    - Correctness: decrypting a ciphertext should result in the message that was originally
    - Efficiency: should be fast
    - Security: similar to IND-CPA, but ALice just gives Eve the public key

Semantic Security

- Adapted for Public/private key

- 

1. Eve process PK
2. Eve issues a pair of plaintexts $M_0$ and $M_1$ to Alice
3. Alice randomly chooses either $M_0$ or $M_1$ to encrypt and sends the encryption back
   a. Alice does not tell Eve which one was encrypted!
4. Eve outputs a guess as to whether Alice encrypted $M_0$ or $M_1$

An encryption scheme is IND-CPA secure if for all polynomial time attackers Eve:

Eve can win with probability <= 1/2 + Ɛ, where Ɛ is *negligible.*

Eve (adversary) Alice (challenger)

PK PK

$M_0$ and $M_1$

pick *b*

Enc(PK, $M_b$)

Guess *b* = 0 or *b* = 1

ElGamal Encryption

- Asymmetric-key
- Key management
- Password management

ElGamal Encryption: Protocol

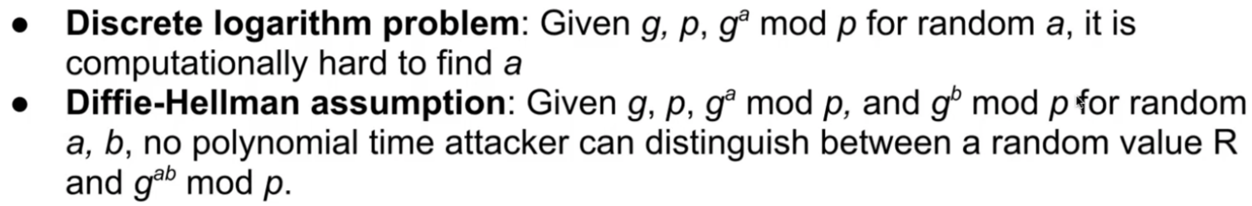
- KeyGen()
    - Generate private key b and public key $B = g^b mod\, p$
- Enc($g^b$, $M$):
    - Alice generates a random ra dn computes $g^r mod\, p$
    - Alice computes $M * (g^b)^r mod\, p$
        - Alice derives the shared secret and multiple her message by the secret
    - Alice sends $C_1 = g^r mod\, p,\ C_2 = M * (g^b)^r mod\, p$
- Dec($b, g^r, M * g^{gr}$)
    - Bob computes M $M * g^{br} * (g^r)^{-b} mod\, p$
    - Bob derives the inverse shared secret and multiples the ciphertext by the inverse shared secret

## Week 6: Lecture 15 Public-Key Encryption and Digital Signatures (9/29)

ElGamal Encryption

- Assume everyone knows a large prime p (2048 bits long) and a generator g

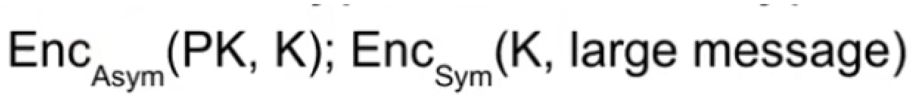
Cryptographic Assumptions



- **Discrete logarithm problem**: Given $g$, $p$, $g^a$ mod $p$ for random $a$, it is computationally hard to find $a$
- **Diffie-Hellman assumption**: Given $g$, $p$, $g^a$ mod $p$, and $g^b$ mod $p$ for random $a$, $b$, no polynomial time attacker can distinguish between a random value R and $g^{ab}$ mod $p$.

ElGamal Encryption: Issues
- Semantically secure
    - For 1 < M < p-1

ElGamal Malleability
- Adversary can very easily tamper with the message
- Encryption does not provide integrity
- Can be a feature: homomorphic encryption (operation on the ciphertext results in an operation on the underlying plaintext)

Hybrid Encryption
- Issues with public-key encryption
    - Can only encrypt small messages bc modulo operator
- Use Diffei to figure out and share a symmetric key
- Hybrid Encryption:
    - Encrypt data under a randomly generated key K using symmetric encryption, and encrypt K using asymmetric encryption
    - 

      $\text{Enc}_{\text{Asym}}(\text{PK}, \text{K}); \text{Enc}_{\text{Sym}}(\text{K}, \text{large message})$

<u>Digital Signatures</u>

Digital Signatures
- 3 parts
    - KeyGen() -> PK, SK: Generate a public/private keypair, where PK is the verify (public) key and SK is the signing (secret) key
    - Sign(SK, M) -> sig: sign the message M using the singing key SK to produce the signature sig
    - Verify(PK, M, sig) -> {0, 1}: Verify the signature sig on message M using verify key PK and output 1 if valid and 0 if invalid
- Properties
    - Correctness: verification be successful for a signature generated over any message
    - Efficiency: should be fast
    - Security: EU-CPA

RSA Signatures
- KeyGen():
    - Randomly pick two large primes, p and q

        - Computer N = pq
        - Choose e
            - Requirement: e is relatively print o (p - 1) (q - 1)
            - Requirement: 2 < e < (p -1) (q - 1)
        - Computer $d = e^{-1} mod\ (p - 1)\ (q - 1)$
        - Public key: N and e
        - Private key: d
- Sign(d, M)
    - Compute $H(M)^{d}\ mod\ N$
- Verify(e, N, M, sig)
    - Verify that $H(M) \equiv sig^{e}\ mod\ N$

RSA Digital Signatures: Security

- Factoring hardness assumption: Given N large, it is hard to find primes p q = N
- Discrete logarithm hardness assumption: Given N large, hash, and $hash^{d}$mod N, hard to find d

Summary

- Public-key cryptography: two keys; one undoes the other
- Public key encryption: one key encrypts, the other decrypts
    - Security properties similar to symmetric encryption
    - ElGamal: Based on Diffie-Hellman
        - Public key is $g^{b}$and C1 is $g^{r}$
        - Not IND-CPA secure on its own
- Hybrid encryption: encrypt a symmetric key and use symmetric key to encrypt the message
- Digital signatures: integrity and authenticity for asymmetric schemes
    - RSA: Same as RSA encryption but encrypt the has with the private key

## Week 6: Lecture 16 Public-Key Encryption and Digital Signatures (10/1)

Trust-on-first-use

- first time you communicate, trust the public key that is used and warn the user if it changes in the future
    - Attacks aren't frequency so assume that you aren't being attacked the first time communicate

Certificates

- Signed endorsement of someone's public key
- Alice wants Bob's public key, ALice trusts Charlie

Certificate Authorities

- trust authorities

- Get chain of certificates
- Signed by certificate authority that my broker would trust

Revocation

- Announcing Revoked Certificates
    - Lists can get large
- Set expiration date
- Trusted Directory

Storing Passwords

- A secret string a user types in to prove their identity
- How does the service check that your password is correct
- Hash the password,
- Brute force attack if two different users decide to use password123
- Known common passwords, hash entire dictionary of commons passwords
- Rainbow tables
    - Algorithm for computing hashes that makes brute force attacks easier store hash mapped to password, attack can easily look up every hashed password in the database

Salted Hashes

- Add a unique, random salt for each user
- Salt: random, public value designed to make brute-force attacks harder
- For each user, store, username, salt, H(password || salt)
- Look up salt in passwords and compute
- Brute force attacks are now much harder

Slow Hashes

- Use slower hashes
- Cryptographic hashes designed to be fast

Offline attack

- attacker performs all the computation themselves
- Steal the password file
- Salted slow hashes

Online attacks:

- attacker interacts with the service
- Use timeouts

## Week 7: Lecture 17 Bitcoin/Intro to Web (10/8)

Finance is all zero sum at best

- Every dollar made in cryptocurrency came from someone else
    - No dividends or interest flowing in unlike stock a or bond market
- Similar to ponzi scheme

What is the web?

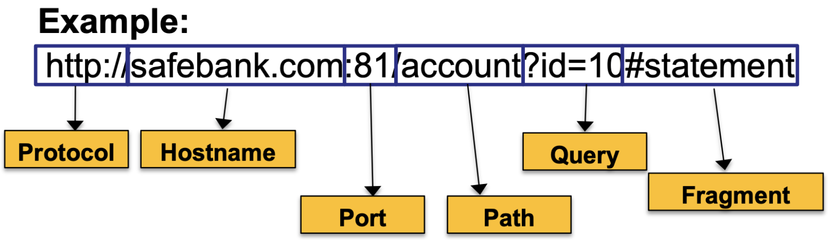
- Platform for deploying applications and sharing information portably and securely

HTTP (hypertext transfer protocol)

- Common data communication protocol on the web

URLs

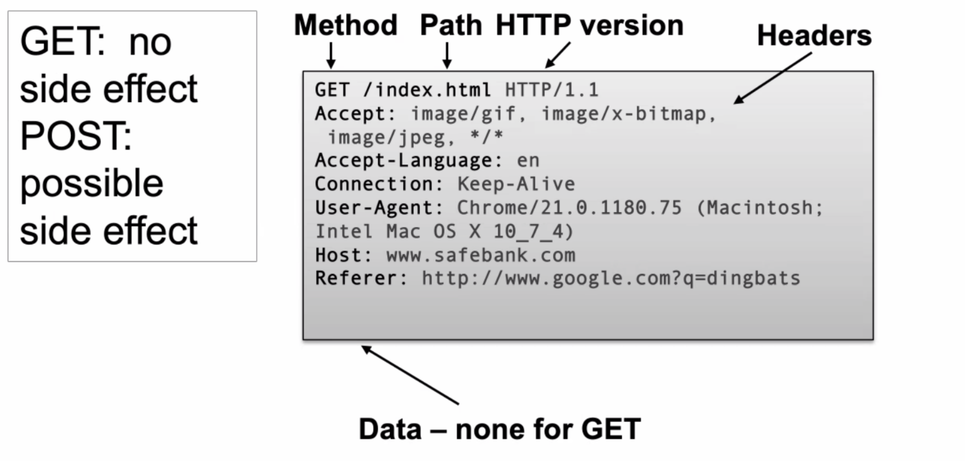
- Global identifiers of network-retrievable resources



**Example:**

http://safebank.com:81/account?id=10#statement

HTTP Request

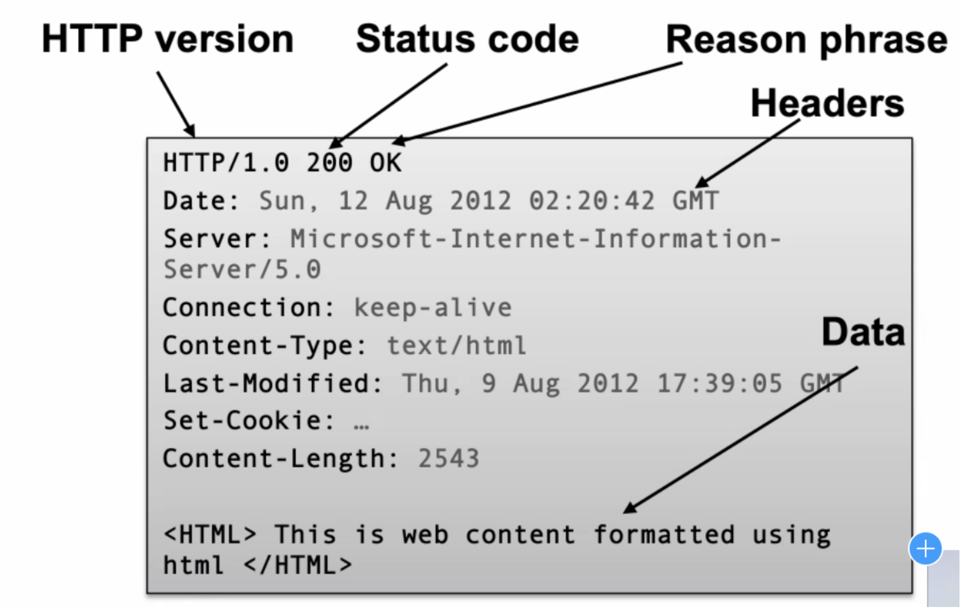


GET: no side effect
POST: possible side effect

```
GET /index.html HTTP/1.1
Accept: image/gif, image/x-bitmap,
 image/jpeg, */*
Accept-Language: en
Connection: Keep-Alive
User-Agent: Chrome/21.0.1180.75 (Macintosh;
Intel Mac OS X 10_7_4)
Host: www.safebank.com
Referer: http://www.google.com?q=dingbats
```

-

HTTP Response



```
HTTP/1.0 200 OK
Date: Sun, 12 Aug 2012 02:20:42 GMT
Server: Microsoft-Internet-Information-
Server/5.0
Connection: keep-alive
Content-Type: text/html
Last-Modified: Thu, 9 Aug 2012 17:39:05 GMT
Set-Cookie: …
Content-Length: 2543

<HTML> This is web content formatted using
html </HTML>
```

HTML

- Language to create structured documents, embed images, objections

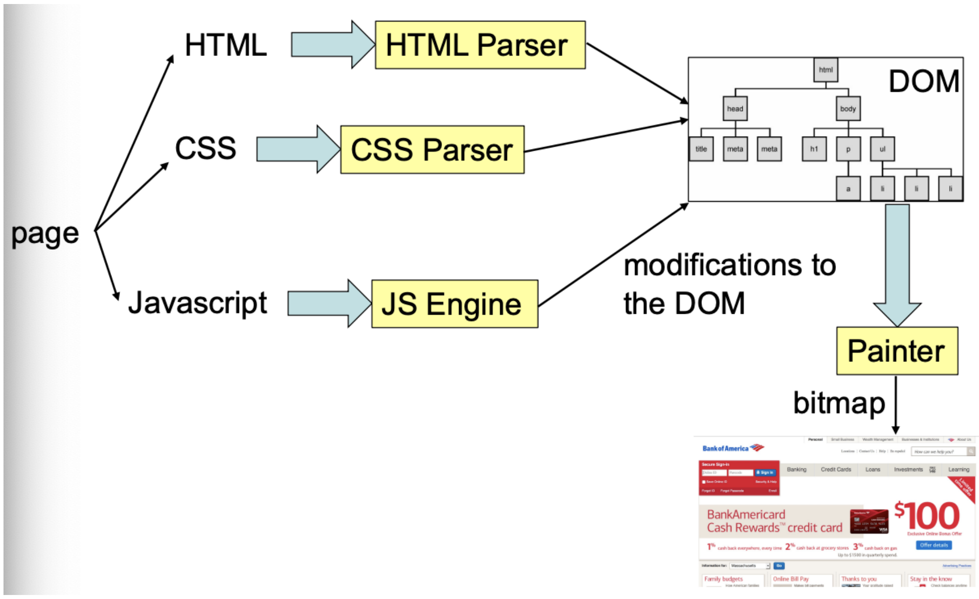
CSS (Cascading Style Sheets)

- Style sheet to describe presentation of document

JavaScript

- Programming language used to manipulate web pages, high-level, untyped and interpreted language with support for objects

Page rendering

- 

DOM

- Cross-platform model for representing and interacting with objects in HTML

JavaScript

- Can read cookies, change cookies

Frames

- Enable embedding a page within a page
- Modularity, delegation: can only draw on its own rectangle

Web

- Bolt on security, security was added as an afterthought
- Added embedded images, javascript, dynamic html, ajax, audio video
- Web security is a challenge

Desirable security goals

- Integrity
    - Should not be able to tamper with integrity of my computer or info from other web sites
- Confidentiality
    - Malicious web sites should not be able to learn confidential info from my computer to other websites

- Privacy
    - Malicious web sites should not be able to spy on me or my activities online
- Availability: attacker cannot make site unavailable

Security on the web

1.

## Week 8: Lecture 18 Same-origin Policy, SQL injection (10/11)

Security on the web

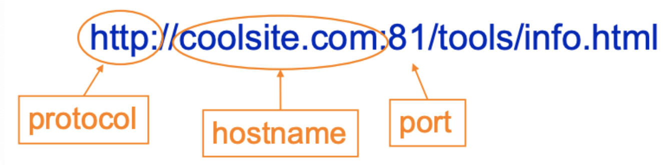
1. Don’t want a malicious site to be able to access files/programs on my computer
    a. Defense: javascript is sandboxed
2. Don’t want a malicious site to spy or tamper with my information interactions with other websites
    a. Defense: the same-origin policy
3. Data stored on a web server to be protected from unauthorized access
    a. Defense: server-side security

Same-origin policy

- Multiple pages from same site are not isolated but security barrier when site from different pages

Origin

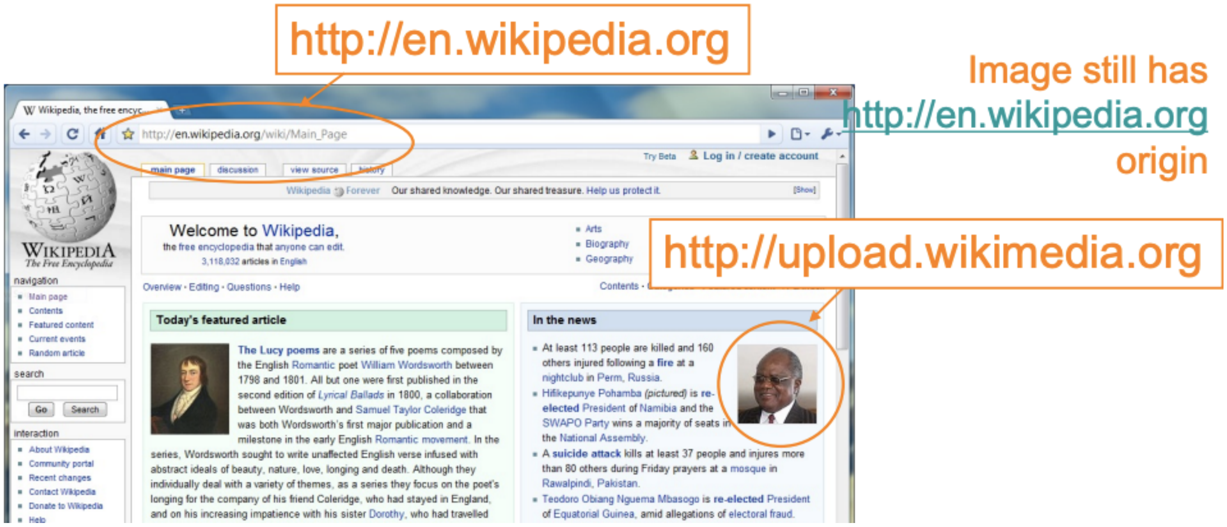
- Porocol, hostname, port



- String matching between these

Same-origin policy

- One origin should not be able to access the resources of another origin

- 

http://en.wikipedia.org

Image still has http://en.wikipedia.org origin

http://upload.wikimedia.org

Wikipedia Forever Our shared knowledge. Our shared treasure. Help us protect it.

Welcome to Wikipedia,
the free encyclopedia that anyone can edit.
3,118,032 articles in English

Today's featured article

The Lucy poems are a series of five poems composed by the English Romantic poet William Wordsworth between 1798 and 1801. All but one were first published in the second edition of *Lyrical Ballads* in 1800, a collaboration between Wordsworth and Samuel Taylor Coleridge that was both Wordsworth's first major publication and a milestone in the early English Romantic movement. In the series, Wordsworth sought to write unaffected English verse infused with abstract ideals of beauty, nature, love, longing and death. Although they individually deal with a variety of themes, as a series they focus on the poet's longing for the company of his friend Coleridge, who had stayed in England, and on his increasing impatience with his sister Dorothy, who had travelled

In the news

- At least 113 people are killed and 160 others injured following a **fire** at a nightclub in Perm, Russia.
- Hifikepunye Pohamba *(pictured)* is **re-elected** President of Namibia and the SWAPO Party wins a majority of seats in the National Assembly.
- A **suicide attack** kills at least 37 people and injures more than 80 others during Friday prayers at a mosque in Rawalpindi, Pakistan.
- Teodoro Obiang Nguema Mbasogo is **re-elected** President of Equatorial Guinea, amid allegations of electoral fraud.

Origins of other components

- Iframe: origin of the URL from which the iframe is served, not the loading website

Exercise

- Wikipedia.org vs www.wikipedia.org because string matching not same origin

Cross-origin communication

- Allowed through narrow API: postMessage
- Receiving origin decides if to accept the message based on origin (enforced by browser)

What happens if a web server is compromised

- Steal sensitive data
- Change server data
- Gateway to enabling attacks

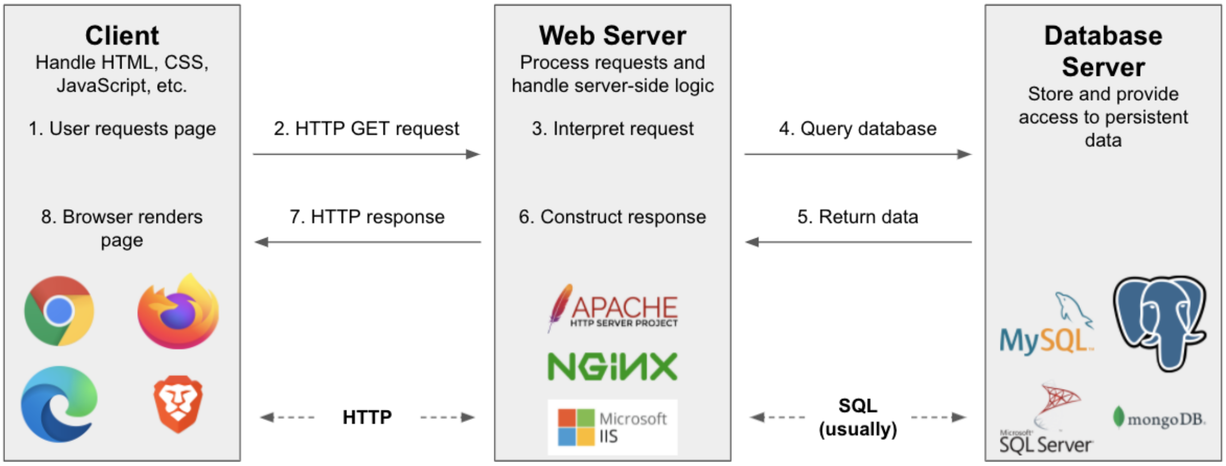
A set of common attacks

- SQL Injection
    - browser sends malicious input to sever
    - Bad input checking can lead to malicious SQL query
- XSS - Cross-site scripting
- CSRF - Cross-site request forgery

Structure of Web Services

- Websites need to store data
    - User accounts

- HTTP is stateless: only handles the HTTP requests and needs to have some way of storing and retrieving personal data



SQL

- Structured Query Language
- Language used to interact with and manage data stores in a database

SQL syntax

- SELECT to get data
- WHERE to filter out certain rows
- INSERT INTO to get more rows
- UPDATE to change values of existing rows in table
- DROP to delete table
- (--) used for single-line comments
- Semicolons separate different statements

## Week 8: Lecture 19 Cookies and Session Management (10/13)

SQL injection defenses

- Escape the parser

1. SQL Parser
    a. Seels ' it considers a string starting or ending
    b. \' converts character as part of a string convert to
    c. Disallow special characters or escape special characters

Defense against command injection in general

1. Input sanitization
    a. Hard to implement
2. Use safe APIs
    a. Executing shell to execute command is too powerful

Untrusted input is used as parsed SQL

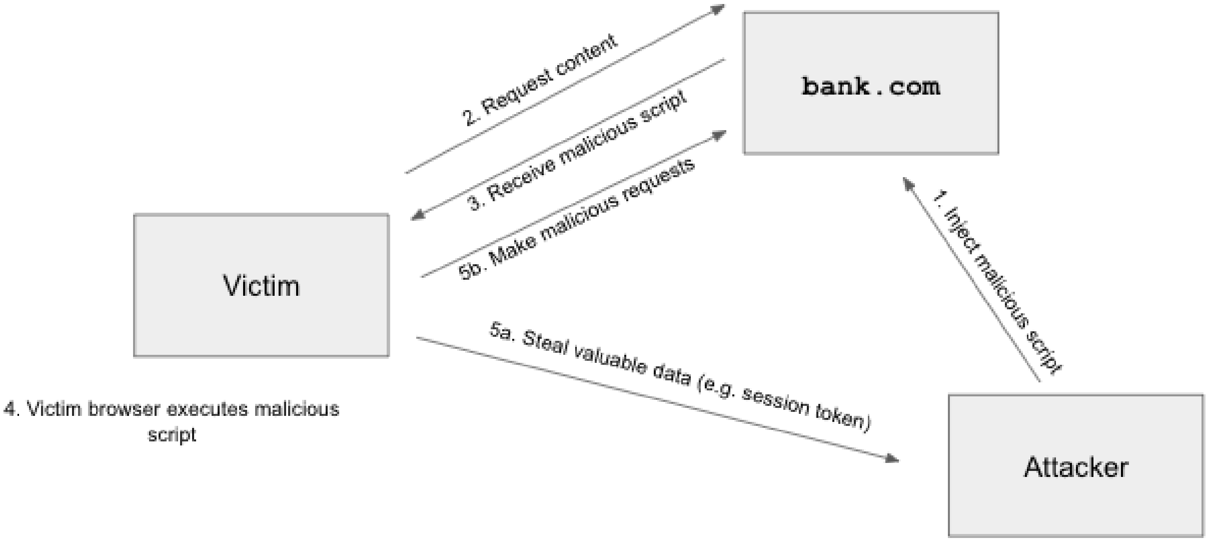
Command injection : untrusted input is used as any parsed language

Cross-site Scripting (XSS)

- Attacker can add malicious code to a legit website
- Injecting javascript into websites that are viewed by other users
- Stored XSS, reflected XSS

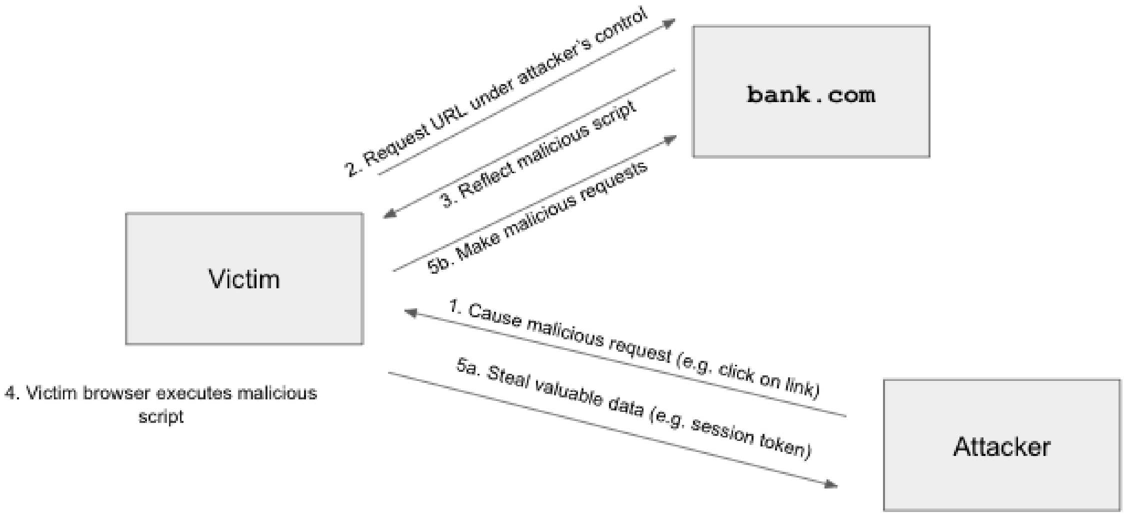
Stored XSS

- Attacker's javascript is stored on the legitimate server and sent to browser
- Requires victim to load the page with injected javascript


bank.com
Victim
Attacker
1. Inject malicious script
2. Request content
3. Receive malicious script
4. Victim browser executes malicious script
5a. Steal valuable data (e.g. session token)
5b. Make malicious requests

Reflected CSS:

- attackers causes the victim to input javascript into a request and the content is reflected in the response from the server
- Requires victim to make a request with injected javascript
- Visit shady website


bank.com
Victim
Attacker
2. Request URL under attacker's control
3. Reflect malicious script
5b. Make malicious requests
1. Cause malicious request (e.g. click on link)
5a. Steal valuable data (e.g. session token)
4. Victim browser executes malicious script

- Trick victim into visiting website

XSS Defenses

1. HTML sanitization
    a. No open tag or close tag
    b. Ampersand and end with a semicolon
2. Content Security Policy (CSP)
    a. Instruct browser to only use resources loaded from specific pages
    b. Disallow inline scripts, only allow scripts from specific domains
    c. Relies on breower to enforce security

## Week 8: Lecture 20 Cookies (10/15)

Cookies

- Cooties are a way to add state to HTTP stateless
- View cookies document.cookie

Cookie Scope

- Expires is expiration date
    - Delete cookie by setting expires to date in past
- http only
    - Coodie cannot be accessed by javascript but only sent by browser

Coodie policy

- Cooide policy has two parts

1. What scopes a URL host name web server is allowed to set on a cookie
2. When the browser send a cookie a url

Cookie scope

- Checks if the web server may set the cookie and if not it will not accept the coodie
- Domain: any domain-suffix of URL homstmena except TLD

- Host = “login.site.com”
- Allowed domains
    - Login.site.com
    - .site.com
- Disallowed domains
    - User.site.com
    - Othersite.com
    - .com
- Login.site.com can set cookies for all of .site.com but not for another site or TLD
- Path: can be set to anything

Only subdomains

When browser sends cookie

- Goal server only sees cookies in its scope
- Browser sends all cookies in URL scope
    - Cookie-domain is domain-suffix of URL-domain and
    - Cookie-path is prefix of URL path
- *.food.exmaple.com
    - Cookie secure only for https

Cookie policy versus same-origin policy

- Consider javascript on a page loaded from URL U
- If a cookie is in scope for a URL U, it can be accessed by Javascript loaded on the page unless cookie as http only slide

Indirectly bypassing same-origin policy using cookie policy

- Cookie policy and same origin policy are different

## Week 9: Lecture 21 CSUF (10/18)

Specter Attack

- Exploits a hardware side-channel attack,
- Use a side channel, force speculative channel

Cookie

- A piece of data used to maintain state across multiple requests
- Cookie policy
    - Server with domain X can set cookie with domain attribute Y if the domain attribute ids domain suffix of the server’s domain and the domain attribute Y is not a top level domain
    - Browser attaches a cookie on a request if the domain attribute is a domain suffix of the servers domain and the path attribute is a prefix of the server’s path

Session Authentication

- Session: a sequence of requests and responses associated with the same authenticated user

- Way the browser can automatically send some info in a request

Session TOken

- Session token: as secret value used to associate request with an authenticated user

Session Tokens with Cookies

- Session token can be implemented with cookies
- Cookies can be used to save any state across requests
- When you log out the browser and server deletes the server

Session TOkens: Security

- Need to generate randomly securely
- Browsers should not send session tokens to the wrong websites

Session Token Cookie Attributes

- What attributes should the server set for the session token
- Domain and path
- Secure
- http Only
- Expires
- HMAC() with a secret key
- Ensure integrity on all other cookies

Cross-site request Forgery (CSRF)

- Tricks the victim into make an unintended request
- Exploits cookie based authentication to perform an action as the victim

Steps of a CSRF Attack

1. User authenticates to the server
2. Attacker tricks the victim into making a lcia request to the server
    a. Tracker the viim into clicking a link
        i. Link can directly make a get request
    b. Put some html on a website the victim will visit
    c. Put javascript on a website
3. Server accepts the malicious request form the victim

CSRF defenses

- CSRF tokens (server based)
- Referer validation (server validation)

CSRF Tokens : Usage

- HTML forms
- CSRF tokens are a defense against this attack
- Everytime the user request a from from the legitimate website, the server attaches a CSRF token as a hidden form field
- When user submits the form the form contains the CSRF token
- Attacker's javascript won't bea bel to create a valid form

Referer Header

- Victim usually makes the malicious request form a different website
- Referrer header: a header in an http request indicates which web page the request was made from

Referer Header

- Checking the referer header
  Allow same-site requests:
- Disallow different-site requests
- Ferrara header may leak private information

Same Site Cookie Attribute

- Idea; implement a flag on a cookie that makes it unexploitable
- Samesite flag: a flag on a cookie that specifies it should be sent only when the domain of the cookie exactly matches the domain of the origin

If you are doing production website

- Enable same site cookies content security policy

CSRF defenses: Summary

- CSRF token: a secret value provided by the server to the user. The user must attach the same value in the request for the server to accept the request
- Referer EHeader: Allow same-site request but disallow cross-site requests
- Same-site cookie attribute: the cookie is sent only when the domain of the cookie exactly machines the domain of the origin

## **Week 9: Lecture 22 UI attacks/CAPTCHAs (10/20)**

Clickjacking Download

- Invisible iframes: frame the legitimate site invisibly, over visible enticing content

2. Frame legitimate site visibly, under invisible
3. Frame the legitimate site visibly, under malicious content partially overlapping the site

Clicking jacking: temporal attack

- Javascript can detect the position of the cursor and put something else in front of it

Clickjacking: cursor jacking

- Can make a fake cursor and make the original cursor less visible

Clickjacking: defenses

- Enforce visual integrity
- Ensure clear visual separate between important dialogs and content

Phishing

- Copied paypal source
- Tricking the victim into sending the attacker personal information
- Main vulnerability: person can’t distinguish

Phishing: homograph Attacks

- Check if the URL is correct
- Creating malicious URLs that look similar to legitimate URLs

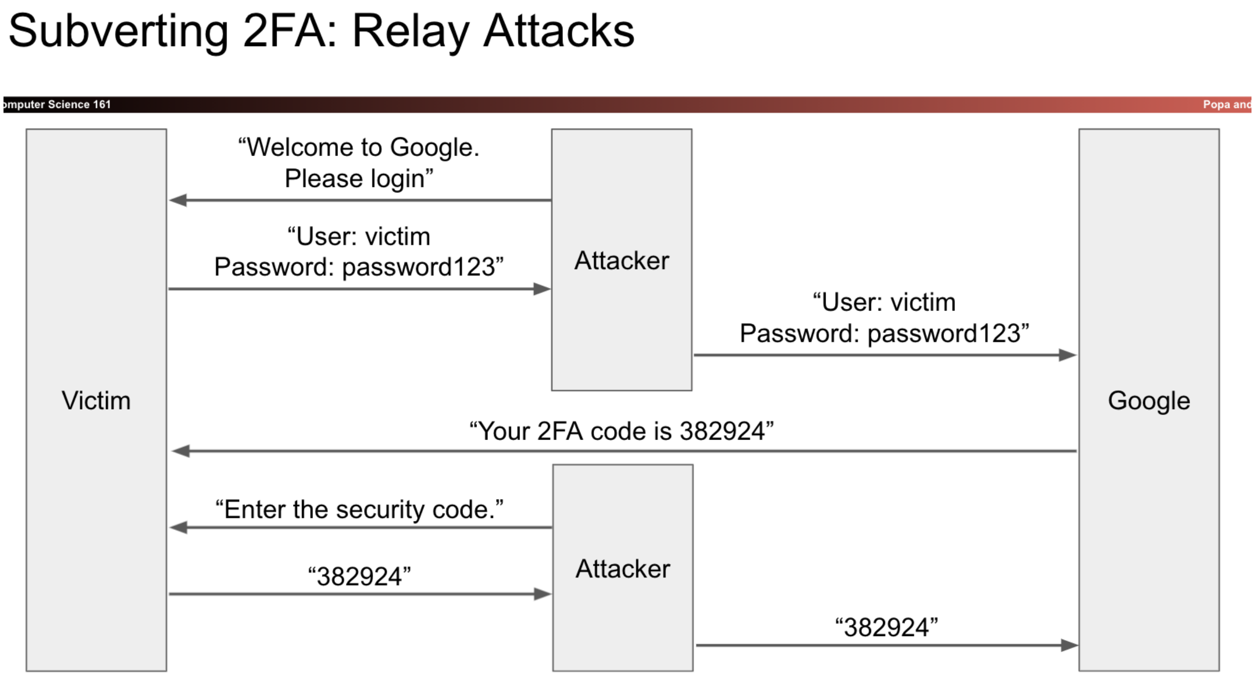
        - Homograph: two words that look the same but have different meanings

Phishing: Browser in browser Attacks

- Don’t blame the Users
- Attacks are uncommon: users don't always suspect malicious action
- Detecting phishing is hard, even if you’re on the lookout for attacks

Relay attack

- Counter 2 factor
- Immediately after phishing webpage login, log into the actual website which will prompt for 2 factor authentication and input into
- Steal both attacks in same relay attack



Subverting 2FA: Relay Attacks

mputer Science 161
Popa and
Victim
Attacker
Google
Attacker
“Welcome to Google. Please login”
“User: victim Password: password123”
“User: victim Password: password123”
“Your 2FA code is 382924”
“Enter the security code.”
“382924”
“382924”

2FA Example: Authentication Tokens

- Device that generates secure second-factor codes
- Something the user owns

## Week 9: Lecture 23 Networking (10/22)

Phishing

- Defense: Two-Factor Authentication
- User must prove their identity in two different ways

What is the Internet

- network: a set of connected cam Hines that can communicate with each other
- Internet: global network of computers
    - Web sends data between browsers and servers using the internet

Protocol

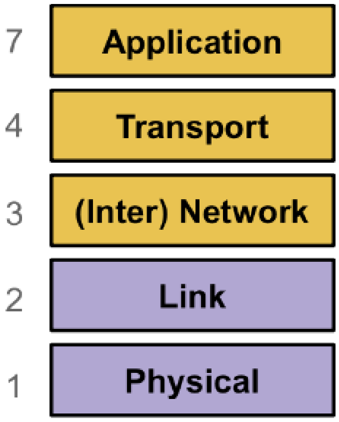
- agreement on how to communicate that specifies syntax and semantics
    - Syntax: how a communication is specified and structured
    - Semantics: what a communication means (actions taken when sending/receiving messages)

Layering

- Each layer has a protocol, communicates with each other

OSI Model

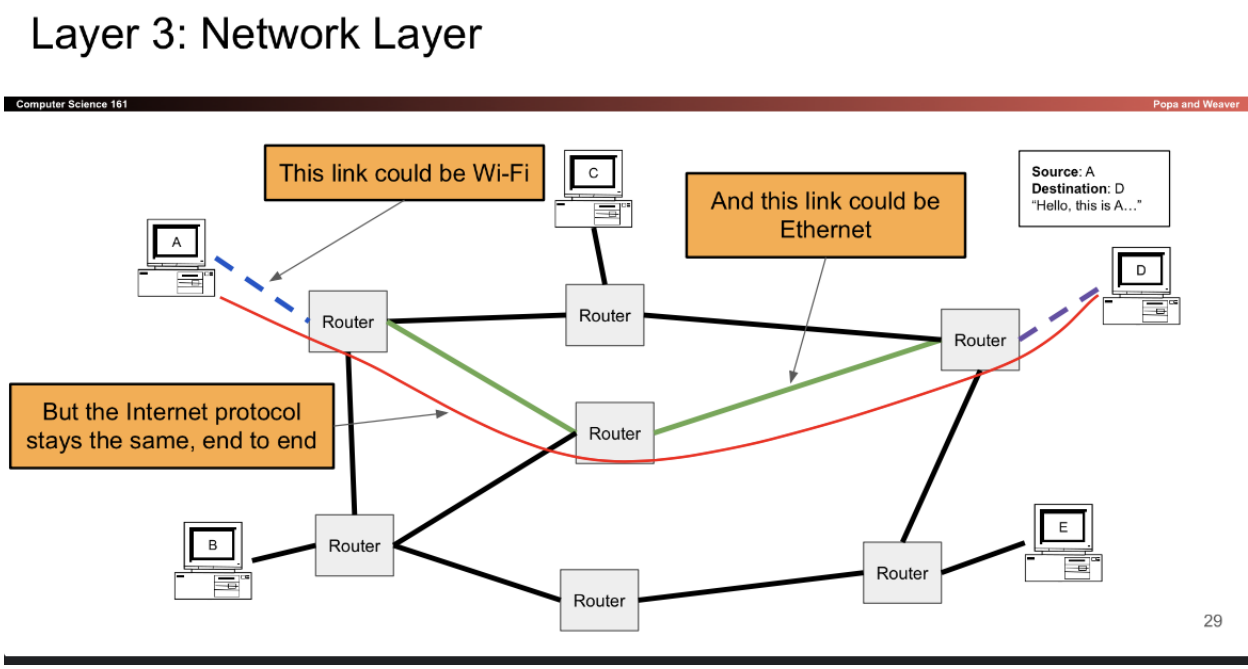
- open systems interconnection model, a layered model of internet communication



-

1. Physical
    a. Sends bits from one device to another
    b. Wi-Fi radio, Ethernet voltages
2. Link
    a. Sending frames directly from one device to another
    b. encode messages into groups of bits called frames
    c. Allows direct addressing between any two devices
    d. Ethernet frames
3. Inter - network
    a. Sending packets from any device to any other device
    b. Encodes messages into groups of bits called “packets”
    c. Adds routers to get packet to next closest, routed across different devices to reach the destination
    d. Internet Protocol (IP)

e.



## Layer 3: Network Layer

Computer Science 161 Popa and Weaver

This link could be Wi-Fi

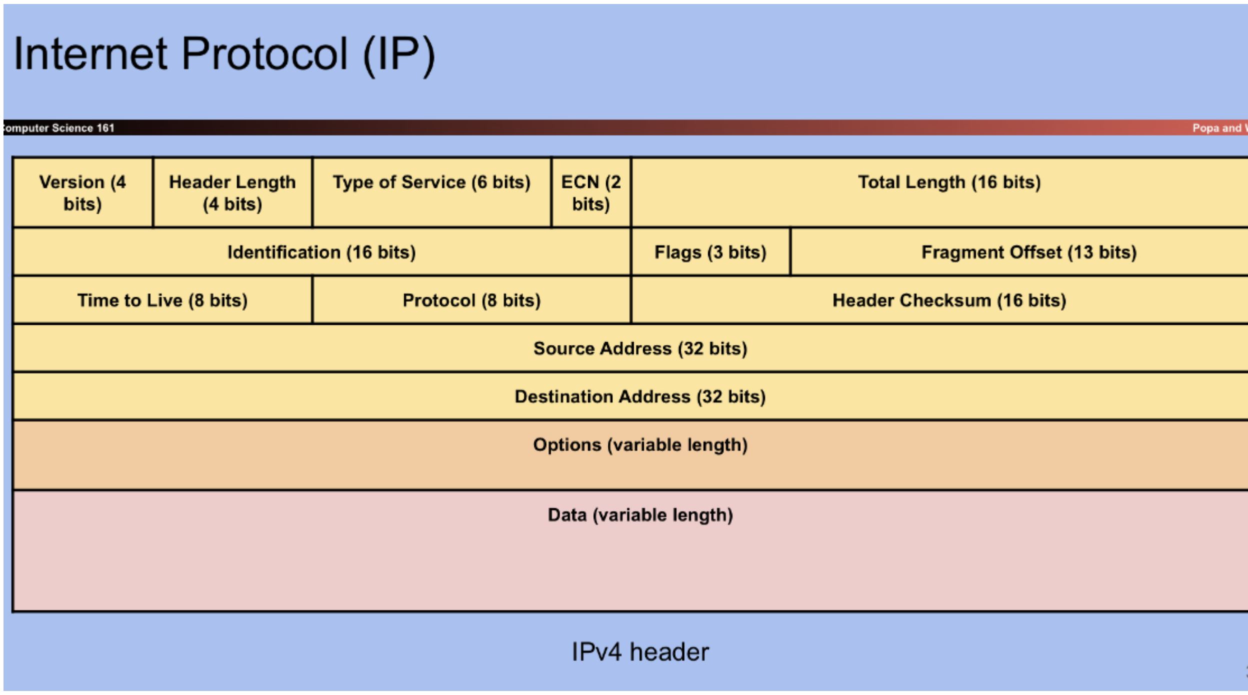
And this link could be Ethernet

But the Internet protocol stays the same, end to end

**Source**: A
**Destination**: D
"Hello, this is A…"

29

f.



## Internet Protocol (IP)

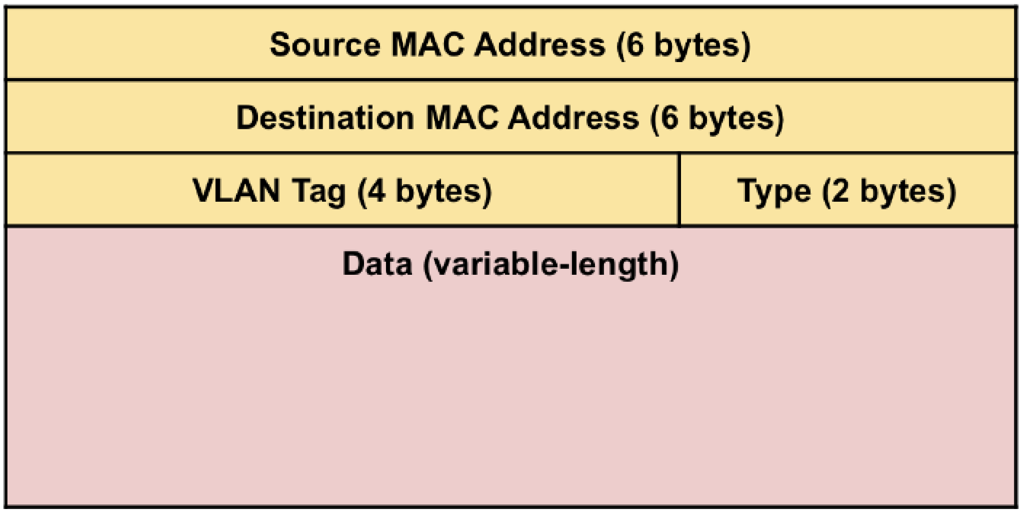
omputer Science 161 Popa and \

<table>
<tr><td><b>Version (4 bits)</b></td><td><b>Header Length (4 bits)</b></td><td><b>Type of Service (6 bits)</b></td><td><b>ECN (2 bits)</b></td><td colspan="2"><b>Total Length (16 bits)</b></td></tr>
<tr><td colspan="4"><b>Identification (16 bits)</b></td><td><b>Flags (3 bits)</b></td><td><b>Fragment Offset (13 bits)</b></td></tr>
<tr><td colspan="2"><b>Time to Live (8 bits)</b></td><td colspan="2"><b>Protocol (8 bits)</b></td><td colspan="2"><b>Header Checksum (16 bits)</b></td></tr>
<tr><td colspan="6"><b>Source Address (32 bits)</b></td></tr>
<tr><td colspan="6"><b>Destination Address (32 bits)</b></td></tr>
<tr><td colspan="6"><b>Options (variable length)</b></td></tr>
<tr><td colspan="6"><b>Data (variable length)</b></td></tr>
</table>

IPv4 header

g. Each router forwards a given packet to the next hop, may be fragmented into smaller packets
h. Must have source, destination, data
i. Source address, destination address, should be unique
j. Internet protocol: Universal layer 3 protocol that all devices use to transmit data over the internet
k. IP Address: an address that identifies a device on the internet

    l. Reliability: ensure that packets are received correctly or if random errors occur not at all
    m. IP is unreliable and only provides a best o effort delivery service, may be lost, corrupted, delivered out of order
4. Transport
    a. Transportation of variable length data from any point to any other points
    b. Reliability: reliably in order
    c. Ports: provide multiple addressee per real IP address
    d. TCP: Provides reliability and ports
    e. UDP: provides ports but no reliability
5. -
6. -
7. Application
    a.



| | |
|---|---|
| Source MAC Address (6 bytes) | |
| Destination MAC Address (6 bytes) | |
| VLAN Tag (4 bytes) | Type (2 bytes) |
| Data (variable-length) | |

Ethernet header

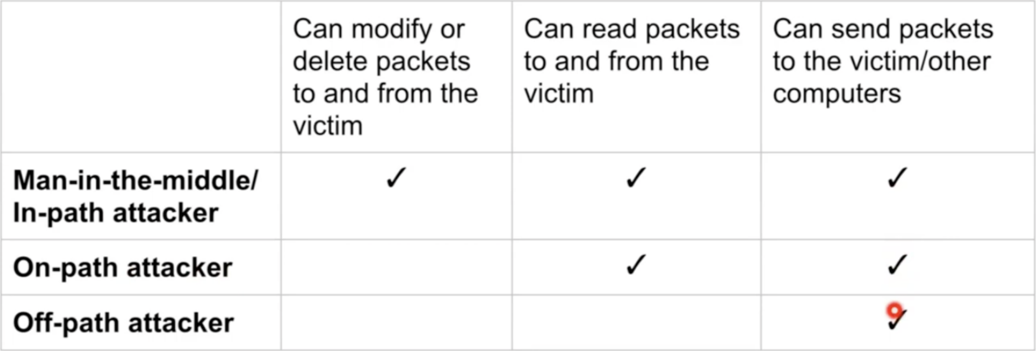
Ethernet and MAC Addresses

- a common layer2 protocol that most endpoint devices use
- MAC address: A 6 byte address that identifies a piece of network equipment

## Week 10: Lecture 25 Low Level Network Attacks: ARP, DHCP (10/25)

Network Attackers

- Threat Model



| | Can modify or delete packets to and from the victim | Can read packets to and from the victim | Can send packets to the victim/other computers |
|---|---|---|---|
| **Man-in-the-middle/ In-path attacker** | ✓ | ✓ | ✓ |
| **On-path attacker** | | ✓ | ✓ |
| **Off-path attacker** | | | ✓ |

Spoofing

- Lying about the identity of the send
- All types of attackers can spoof packets, amy be hardr if attack can't read or modify packets
- Layer 1/2/3 does not include protections against spoofing

On-Path attackers

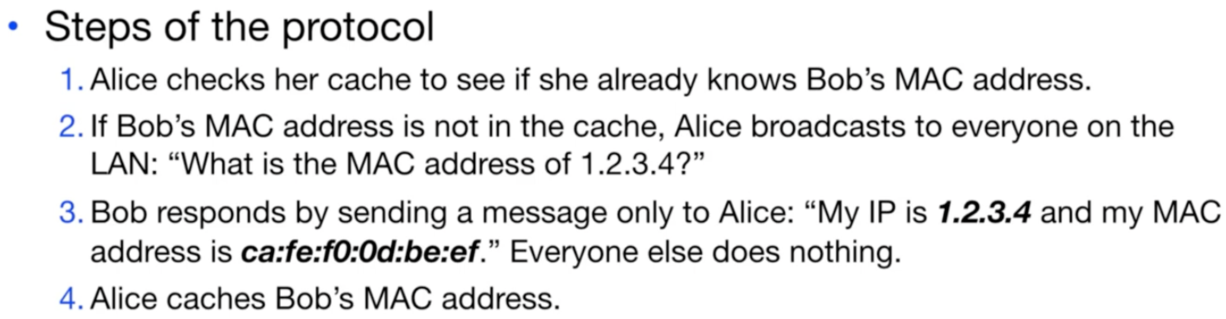
- Read data from the data being transferred

The Law and Sniffing Packets

- Can sniff packets on your own network
- Highly illegal and immoral to sniff traffic

Address Resolution Protocol (ARP)

- Layer 2 and Layer 3
    - Local area network (LAN): A set of machines connected in a local network
    - Set of machines connected together with local network of routers
- Translates Layer 3 IP addresses to layer 2 MAC addresses
- Steps of the protocol



- Steps of the protocol
    1. Alice checks her cache to see if she already knows Bob's MAC address.
    2. If Bob's MAC address is not in the cache, Alice broadcasts to everyone on the LAN: "What is the MAC address of 1.2.3.4?"
    3. Bob responds by sending a message only to Alice: "My IP is ***1.2.3.4*** and my MAC address is ***ca:fe:f0:0d:be:ef***." Everyone else does nothing.
    4. Alice caches Bob's MAC address.

- Alice knows when Bob is outside of the LAN and knows subnet
- Alice knows the IP Address of the "Gateway router"

ARP Spoofing

- Alice has no way of verifying the ARP response, MITM attacker
- Mallory can forward her own MAC address

ARP Defenses

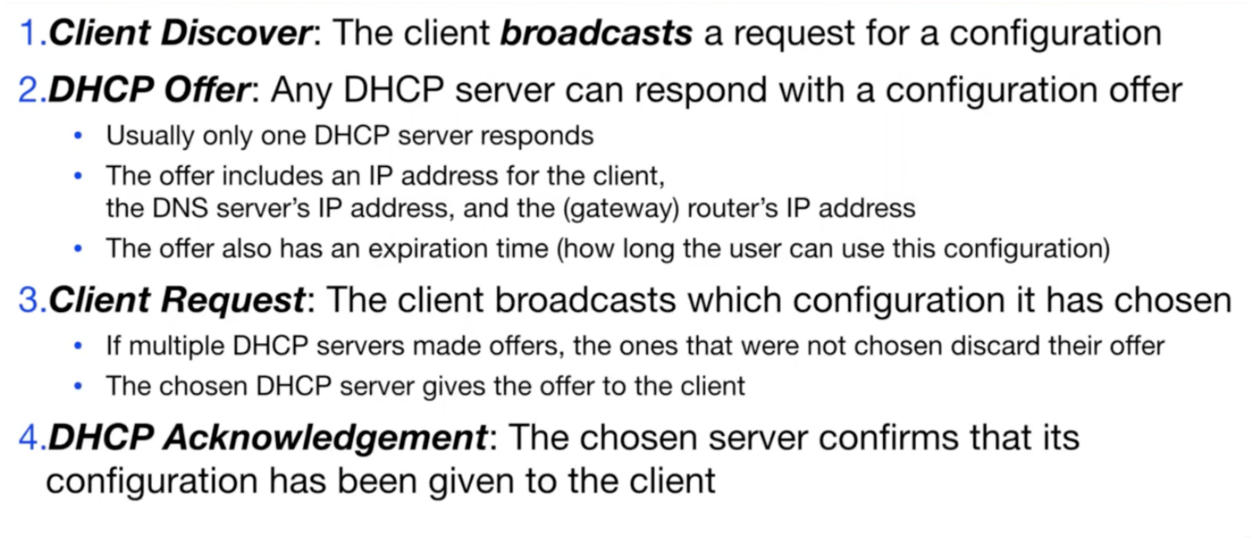
- When Alice wanted to send message to Bob, sends message to a switch on the LAN switch maintains a cache of MAC to port, physical connection mappings
- Switch sends message directly to Bob

Dynamic Host Configuration Protocol (DHCP)

- Initial network Configuration
- To connect to a network, user needs
    - An IP address, IP address of the DNS server, IP address of the router
- First time user doesn't know who to ask for this information

DHCP

- Gives handshake for user configuration when joining



1. ***Client Discover***: The client ***broadcasts*** a request for a configuration
2. ***DHCP Offer***: Any DHCP server can respond with a configuration offer
    - Usually only one DHCP server responds
    - The offer includes an IP address for the client, the DNS server's IP address, and the (gateway) router's IP address
    - The offer also has an expiration time (how long the user can use this configuration)
3. ***Client Request***: The client broadcasts which configuration it has chosen
    - If multiple DHCP servers made offers, the ones that were not chosen discard their offer
    - The chosen DHCP server gives the offer to the client
4. ***DHCP Acknowledgement***: The chosen server confirms that its configuration has been given to the client

-

DHCP Attacks

- Alice cannot verify the DHCP response
    - Spoofing
- Alice usually expects only one DHCP server to respond, will accept the first response
- DHCP attacks require mallory to be in the same LAN as Alice
- DHCP attacks let Mallery be MITM attacker

DHCP Defenses

- Enterprise class switches can offer protection
    - Similar to the ARP-spoofing protection
- Rely on defenses provided in high layers

Summary

- Classes of attackers
    - Off path: can't see, modify or drop but can spoof packets
    - On path: can ses and spoof packets, but can't modify or drop packets

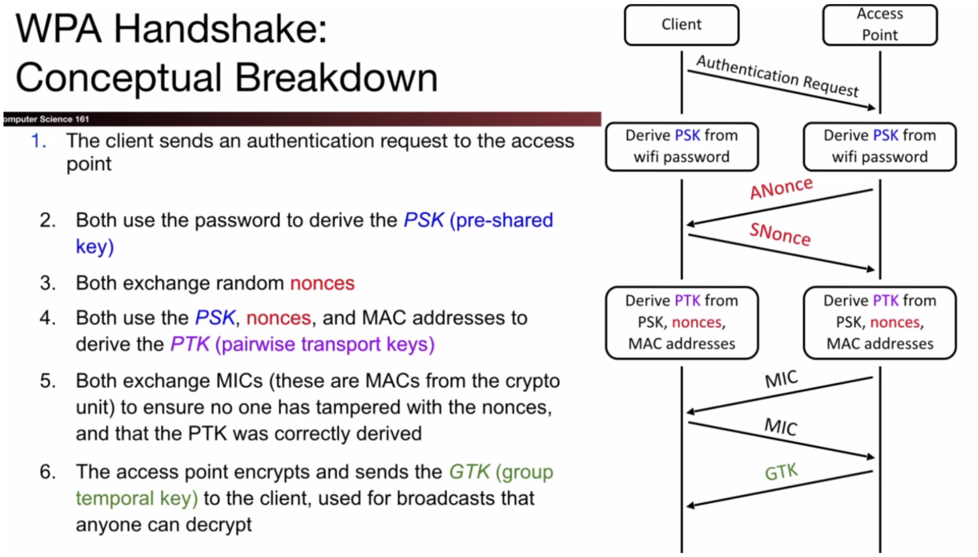
## Week 10: Lecture 26 Wireless Networks (10/27)

Wi-Fi

- Layer ½ protocol that wirelessly connects machines in a LAN
    - Alt is Ethernet, uses wires to connect machines in a LAN
    - Wireless Ethernet
- Parts of the Wifi Network
    - Access Point: a machine that will help users connect

WPA2

- Wifi protected Access 2 (WPA2) : protocol for securing Wifi network communications with cryptography
- Design goals
    - Everyone with wifi password can join the wetwork



WPA Handshake: Conceptual Breakdown

omputer Science 161

1. The client sends an authentication request to the access point
2. Both use the password to derive the *PSK* (pre-shared key)
3. Both exchange random nonces
4. Both use the *PSK*, nonces, and MAC addresses to derive the *PTK* (pairwise transport keys)
5. Both exchange MICs (these are MACs from the crypto unit) to ensure no one has tampered with the nonces, and that the PTK was correctly derived
6. The access point encrypts and sends the *GTK* (group temporal key) to the client, used for broadcasts that anyone can decrypt

Client

Access Point

Authentication Request

Derive PSK from wifi password

Derive PSK from wifi password

ANonce

SNonce

Derive PTK from PSK, nonces, MAC addresses

Derive PTK from PSK, nonces, MAC addresses

MIC

MIC

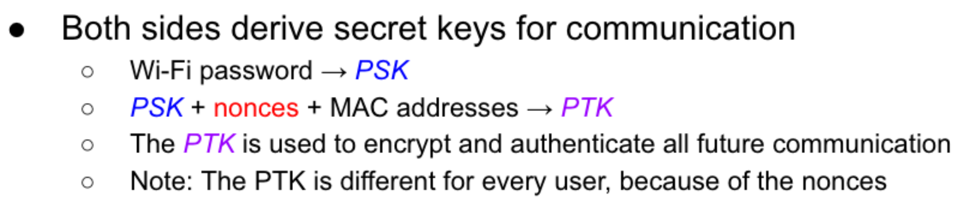
GTK

-

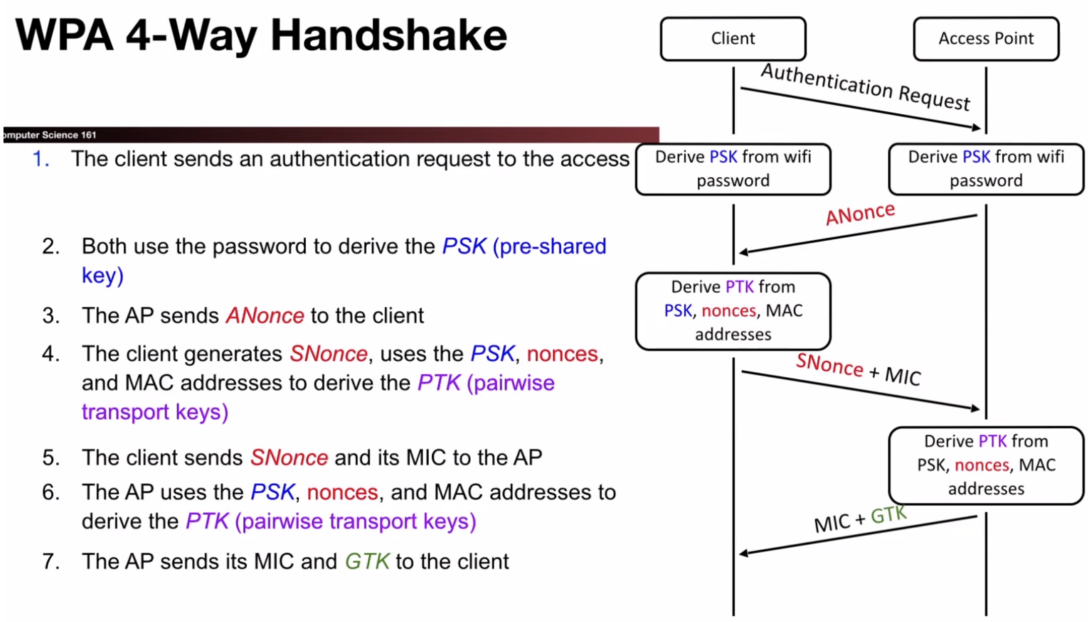
MIC/MAC

- In cryptography
    - MAC: Message Authentication Code
- In networking:
    - MAC == Media Access controller: identifier for devices on network
    - MIC: Message Integrity Code: cryptographic MAC in networking

WPA



- Both sides derive secret keys for communication
    - Wi-Fi password → *PSK*
    - *PSK* + nonces + MAC addresses → *PTK*
    - The *PTK* is used to encrypt and authenticate all future communication
    - Note: The PTK is different for every user, because of the nonces



# WPA 4-Way Handshake

omputer Science 161

1. The client sends an authentication request to the access
2. Both use the password to derive the *PSK* (pre-shared key)
3. The AP sends *ANonce* to the client
4. The client generates *SNonce*, uses the *PSK*, nonces, and MAC addresses to derive the *PTK* (pairwise transport keys)
5. The client sends *SNonce* and its MIC to the AP
6. The AP uses the *PSK*, nonces, and MAC addresses to derive the *PTK* (pairwise transport keys)
7. The AP sends its MIC and *GTK* to the client

Client

Access Point

Authentication Request

Derive PSK from wifi password

Derive PSK from wifi password

ANonce

Derive PTK from PSK, nonces, MAC addresses

SNonce + MIC

Derive PTK from PSK, nonces, MAC addresses

MIC + GTK

WPA-PSK Attacks

- Rogue AP: pretend to be an AP and offer your own ANonce to the client
- Offline brute force attacks: People tend to choose good passwords,and you have enough information to know if you guessed the password correctly
  - Check that the mic from the guess mathes the MIC that was sent
  - No forward secrecy: an eavesdropper who records the values of ANonce and SNonce can derive the key if they alter learn the password or PSK

Disassociation Attack

- A wifi disassociation attack involves spoofing a wifi frame
- A frame saying “hey, disconnect and you can try to reconnect”, denial of service attack
- Only requires knowing the MAC of the client and AP to send
- Send the frame to the victim/client

WPA-Enterprise

- Every client starts with the same PSK to derive the PTK
- Using a randomly generated key by an authentication server
- Accept a digital certificate sending one time key to use instead of PSK to both the client and the AP

Reflections on KRACK

- Getting cryptography right in the real world is hard
- Some cryptographic primitives are far more vulnerable to misuse
- Counter mode and counter mode variants lead a minor screwup and everything fails
- Signatures algorithms that if misused lose all security

## Week 10: Lecture 27 BGP, TCP, UDP (10/29)

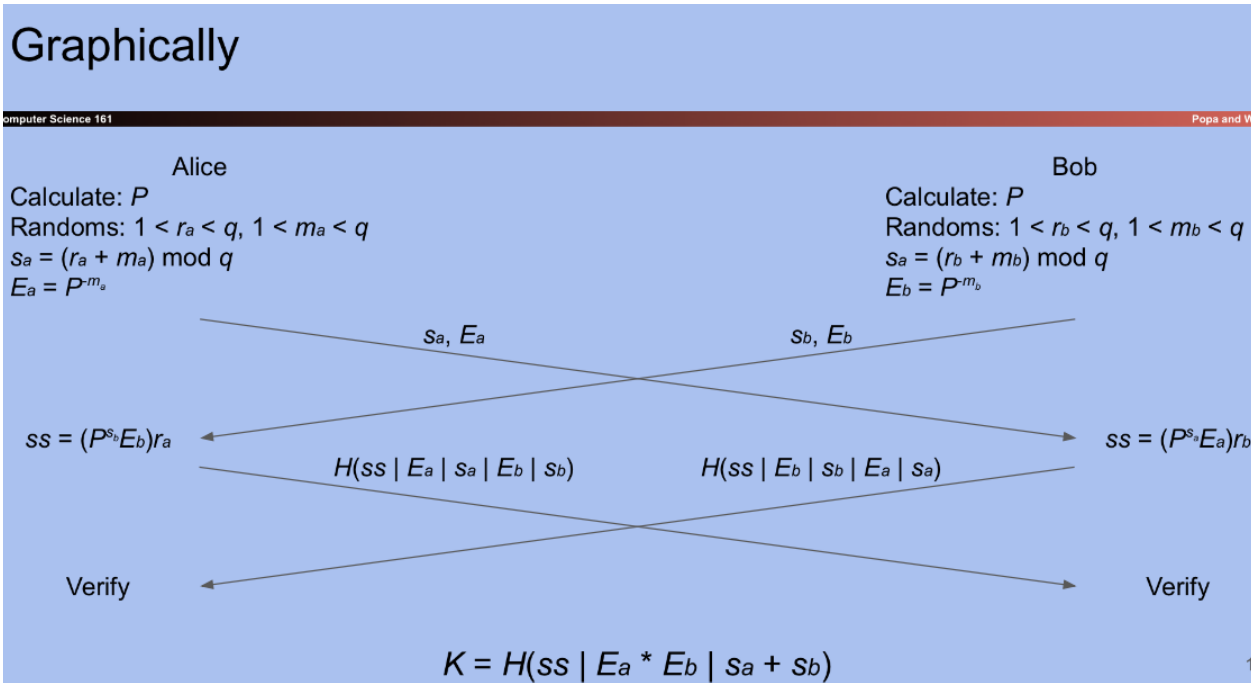
Simultaneous AUthentication of Equals

- ALice and BOb want to create a shared secret
- Alice and bob both know a password, can only generate a shared secret if both of them know the password

Dragonfly

- protocol for simultaneous authentication of equals
    - Based on diffie Hellman
- Main idea; use the password and Alice and bob's identities to generate g
- Picking element at random and seeing if its valid

Prove that everybody knows the same P, and generate a key

- Alice creates
- 

Graphically

omputer Science 161 Popa and W

Alice

Calculate: $P$

Randoms: $1 < r_a < q$, $1 < m_a < q$

$s_a = (r_a + m_a) \bmod q$

$E_a = P^{-m_a}$

Bob

Calculate: $P$

Randoms: $1 < r_b < q$, $1 < m_b < q$

$s_a = (r_b + m_b) \bmod q$

$E_b = P^{-m_b}$

$s_a, E_a$

$s_b, E_b$

$ss = (P^{s_b}E_b)r_a$

$ss = (P^{s_a}E_a)r_b$

$H(ss \mid E_a \mid s_a \mid E_b \mid s_b)$

$H(ss \mid E_b \mid s_b \mid E_a \mid s_a)$

Verify

Verify

$K = H(ss \mid E_a * E_b \mid s_a + s_b)$

Dragonfly and WPA3

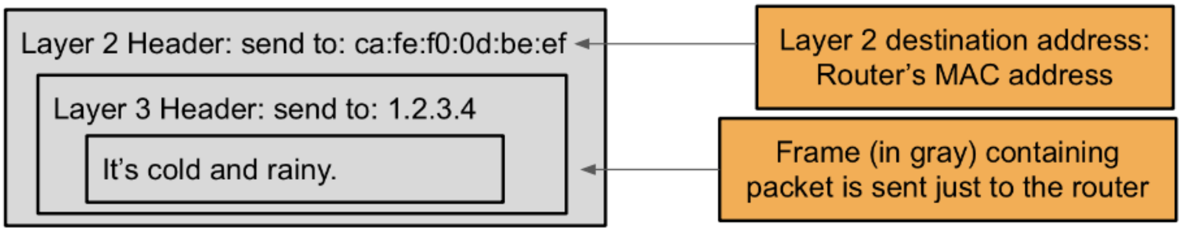
- WPA3 adds dragonfly key exchange before the standard WPA handshake
    - Password from WPA2 is replaced with he shared secret from Dragonfly
- Performance cost: extra latency per handshake
- Security benefits

Wireless Local Networks: Summary

- WPA: A protocol to encrypt Wi-Fi connections at layer 1
- Messages between the client and the AP are encrypted with keys
- WPA-PSK, WPA-Enterprise, WPA-3

Border Gateway Protocol (BGP)

- Internet Protocol
    - Universal layer3 protocol that all devices use to transmit data over the internet

- IP Address: an address that identifies a device on the network
- Layer 3 routes packets across multiple nodes on different LANs
- IP routes by subnets which are grouped of addresses with a common prefix
- 
Layer 2 Header: send to: ca:fe:f0:0d:be:ef
Layer 3 Header: send to: 1.2.3.4
It's cold and rainy.
Layer 2 destination address: Router's MAC address
Frame (in gray) containing packet is sent just to the router

Routing Packets

- TO send a package to computer within the local network, use ARP to get destination MAC address,

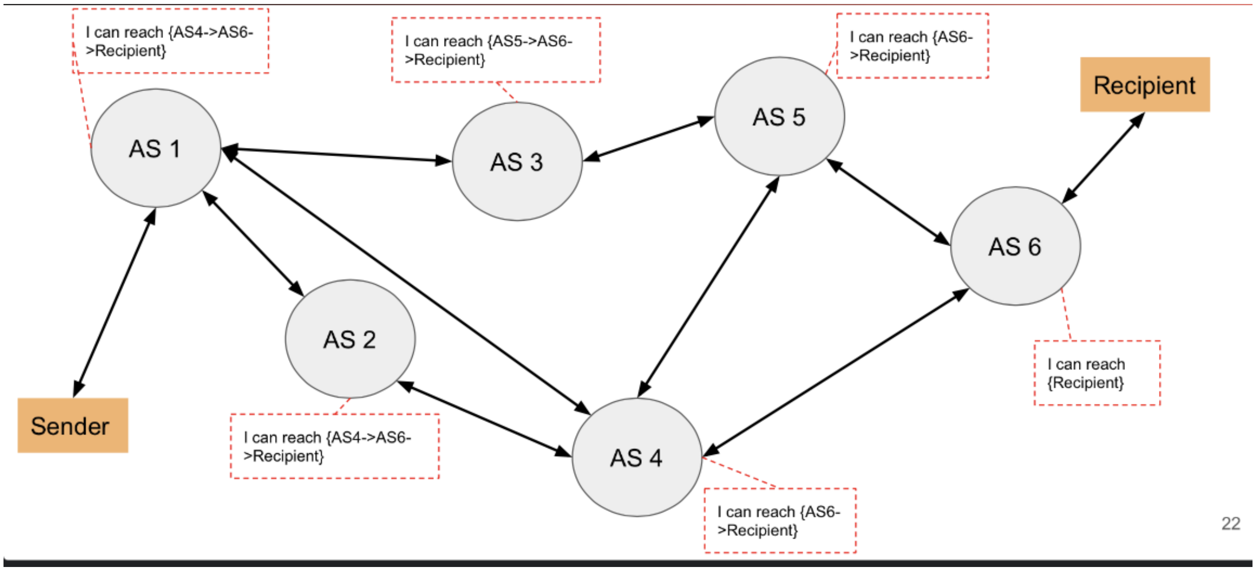
Autonomous Systems

THe internet is a network of networks, comprised of many autonomous systems (AS)

Handoles its internal routing

Forward packet to other connected ASes

Protocol for communicating between different autonomous systems is Border Gateway Protocol (BGP)

- choose shortest path
- 
I can reach {AS4->AS6->Recipient}
I can reach {AS5->AS6->Recipient}
I can reach {AS6->Recipient}
Recipient
AS 1
AS 3
AS 5
AS 6
AS 2
AS 4
Sender
I can reach {Recipient}
I can reach {AS4->AS6->Recipient}
I can reach {AS6->Recipient}
22
- AS implicitly trusts the surrounding ASes and accepts advertised routes

IP and BGP attacks

- IP spoofing: malicious clients can send IP packets from source IP values set to a spoofed value
- BGP hijacking: a malicious autonomous system can lie and claims itself to be responsible for a network which it isn't

Transmission Control Protocol (TCP)

- Reliability: Reliability ensure that packets are received correctly and
- IP is unreliable and is a best effort delivery service

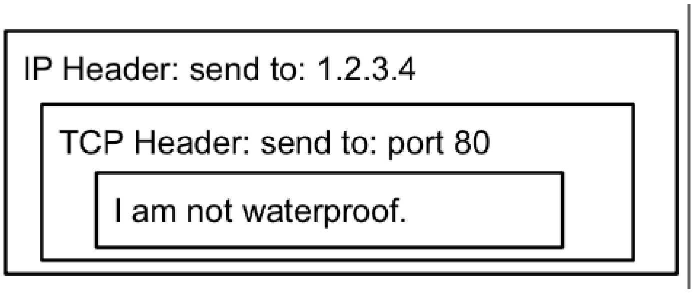
Scratchpad: design

- Packets have a limited size, manually break messages into packets, split and reassembly messages

Transmission Control Protocol (TCP)

- provides a byte stream abstraction
- Bytes in one end of stream at the source and come out at the other end of the destination

Ports

- help us distinguish between different apps on the same computer or server
- 

Establishing Sequence Number

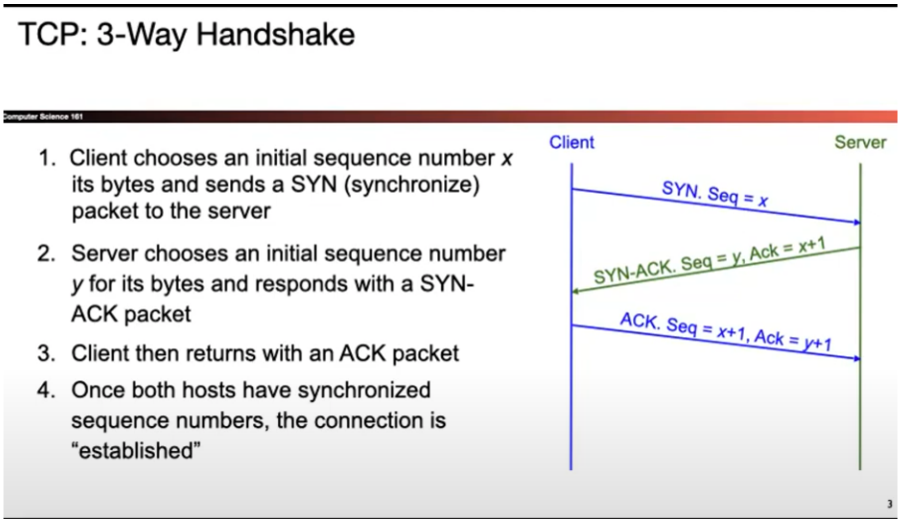
- before starting tcp connection, the client and server must agree on two initial sequence numbers
    - Different and random for every connection
- CLient chooses an initial sequence number x its bytes and sends a SYN packet to the server

TCP: Sending and Receiving Data

- TCP handlers on each side track which TCP segments have been received for each connection
- Identified by these 5 values
    - Source IP, destination IP, source port, destination port, protocol

## Week 11: Lecture 28 TCP, and TLS (11/2)

TCP Handshake



TCP: 3-Way Handshake

Computer Science 161

1. Client chooses an initial sequence number *x* its bytes and sends a SYN (synchronize) packet to the server
2. Server chooses an initial sequence number *y* for its bytes and responds with a SYN-ACK packet
3. Client then returns with an ACK packet
4. Once both hosts have synchronized sequence numbers, the connection is "established"

Will keep sending until ACK is received

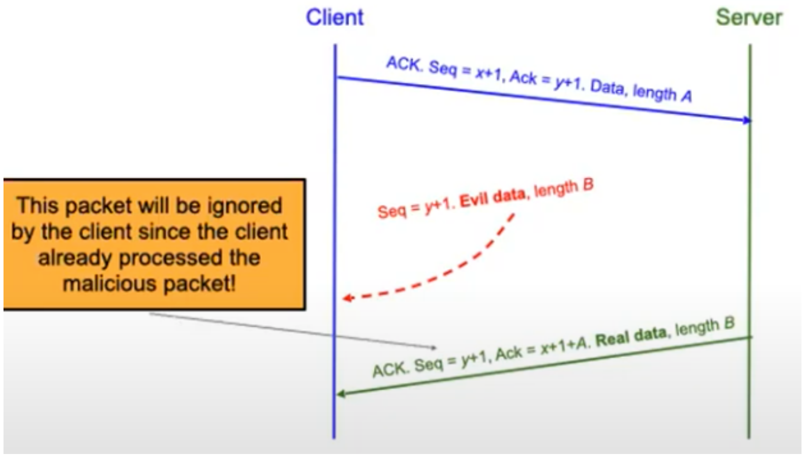
- Uses drops to show congestion
- FIN to end a connection, when packet which should be acknowledged
- RST to abort a connection, are not acknowledged

TCP Flags

- ACK:
    - Indicator that user is acknowledging the receipt of something
    - Pretty much always set except the very first packet
- SYN
    - Indicator of the beginning of the connection

TCP Attacks

- TCP hijacking: Tampering with an existing session to modify or inject data into a connection
    - Data injection: spoofing packets to inject malicious data into a connection
        - On path attackers is a race condition between them
    - RST injection: Spoofing a RST packet to forcibly terminate a connection
- TCP Data injection

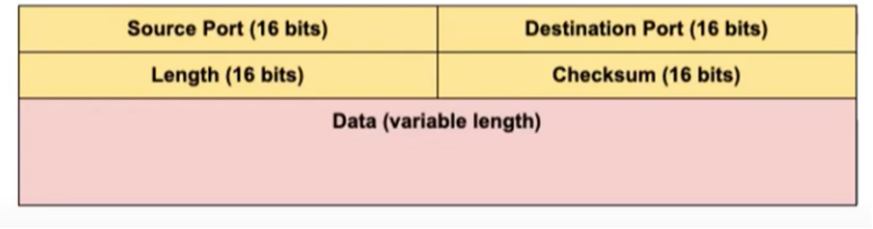

Client
Server
ACK, Seq = x+1, Ack = y+1. Data, length A
Seq = y+1. Evil data, length B
This packet will be ignored by the client since the client already processed the malicious packet!
ACK, Seq = y+1, Ack = x+1+A. Real data, length B

TCP Attacks

- TCP Spoofing: Spoofing a TCP Connection to appear to come from another source IP address
    - IP Packets can be spoofed if the AS doesn't block source addresses

User Datagram Protocol (UDP)

- datagram abstraction
    - Message sent in a single layer 3 packet, break data into datagrams, sent and received as a single unit
- Much faster than TCP, no 3-way handshake



| | |
|---|---|
| Source Port (16 bits) | Destination Port (16 bits) |
| Length (16 bits) | Checksum (16 bits) |
| Data (variable length) | |

Summary

- BGP: routing packets
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

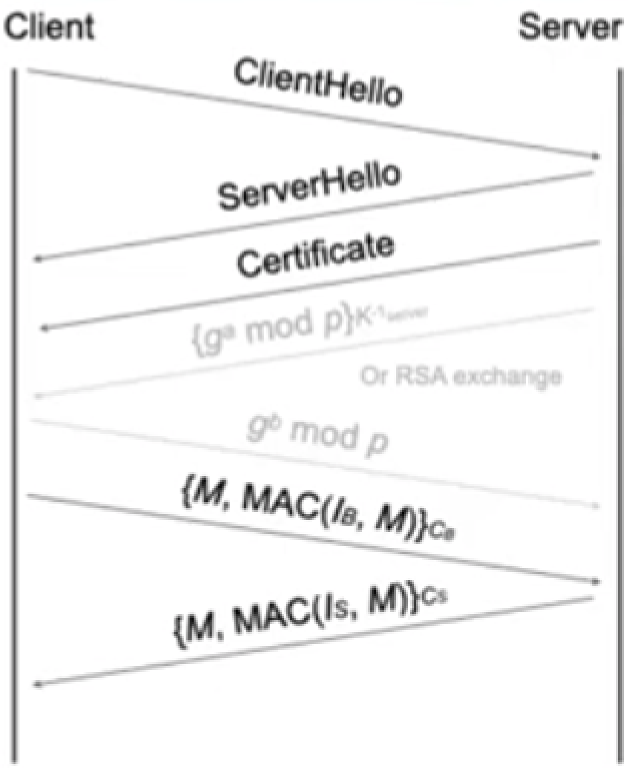
TLS

- Transport layer security (4.5)
    - Protocol for creating a secure communication channel over the internet
- TLS built on top of TCP

Goals of TLS

- Confidentiality: ensure that attackers cannot read your traffic
- Integrity: Ensure that attackers cannot tamper with your traffic
    - Prevent replay attackers
- Authenticity: make sure you’re talking to legitimate server

TLS Handshake


Client
Server
ClientHello
ServerHello
Certificate
$\{g^a \bmod p\}_{K^{-1}_{server}}$
Or RSA exchange
$g^b \bmod p$
$\{M, MAC(I_B, M)\}_{C_B}$
$\{M, MAC(I_S, M)\}_{C_S}$

1. ClientHello, ServerHello:
    a. random number supported cryptographic algorithms
2. Certificate:
    a. Server’s identify and public key signed by trusted certificate authority
3. Premaster Secret
    a. Server prove, client encrypts PS with server’s public key and sends it to the server
    b. (RSA) Server decrypts the pre master secret
    c. (RSA) Client and server now share
    d. (DHE) generate secrete a computes g^a mod p
    e. (DHE) premaster secret: g^ab mod p
4. Derive Symmetric Keys
    a. Server and client each derive symmetric keys
5. Exchange MACs
    a. Server and client exchange MACs on all the messages of the handshake so far
    b. Messages can now be sent securely

Confidentiality and Integrity since

Forward Secrecy

- If attacker records a connection now and comprises secret values later, cannot compromise the recorded connection
- RSA TLS: No forward secrecy is guaranteed

- RSA TLS: No forward secrecy is guaranteed
- DHE TLS: Guaranteed forward secrecy
- AEAD mode encryption

## Week 11: Lecture 29 TLS (11/4)

TLS in Practice Security

- DHE TLS: Forward secrecy
- RSA TLS: No forward Secrecy
- End-to-end security: secure even if all intermediate parties are malicious
- Not anonymous: attackers can determine who

TLS: Efficiency

- Public-key cryptography: minor costs
    - Client server must perform Diffie-Hellman key exchange or RSA enc/dec
- Symmetric-key cryptography: effectively free
    - Modern hardware has dedicated support for symmetric-key cryptography
    - Performance impact is negligible
- Latency: extra waiting time before the first message
    - Must perform the entire TLS handshake before sending the first message

TLS provides End-to-end security

- Secure communication between the two endpoints, with no need to trust intermediaries

Using TLS defends against most lower-level network attacks

TLS Does not provide anonymity

- Hiding the client's and server's identities from attackers
- - attacker can figure out who is communicating with TLS

TLS Does not provide availability

- Aviability: keeping the connection open in the face of attackers
- Attackers can stop a TLS connection
    - On path attackers do RST injection to about the TCP connection
- Result: A TLS Connection can still be censored
    - Censor can block TLS connections

TLS for applications

- TLS provides services to higher layers
- HTTPS: HTTP Protocol run over TLS
- Other secure application layer protocols besides HTTPS exist
- TLS does not defend against application-layer vulnerabilities

TLS in Browsers

- Lock icon means
    - Communication encrypted, legitimate server
- Add a “non secure” icon to connections that don't use TLS

- Delegate the process of trust
- No longer guarantee low image server

Issues: revocation

- What if an attacker steals the server's private key?
    - TLS certificates have an expiration date,
    - Certification renovation lists
- No longer valid if anchors not trusted

Solving Trust Issues

- Certificate Pinning: browser restricts which CAs are allowed to issue a certification for each website
- Certificate transparency: Public logs provided by CAs
    - Specific are out of scope
    - High-level idea: use hash chains to keep a record of all issued certificates
    - Server can tell browser to only accept certificates from CAs implemented transparency
    - Make sure CA never screws up

Certificate authority Example: Let’s Encrypt

- TLS requires every website to obtain and maintain certificates
    - Cost overhead: certification cost money
- Let’s Encrypt: Worlds largest certificate authority
    - ACME Protocol to validate sites

## Week 11: Lecture 30 DNS (11/5)

Domain Names

- Computers are addressed by IP address on the Internet
- Want human-readable domain names

DNS: Definition

- DNS (Domain Name System): An internet protocol for translating human name to IP address

DNS Name Servers

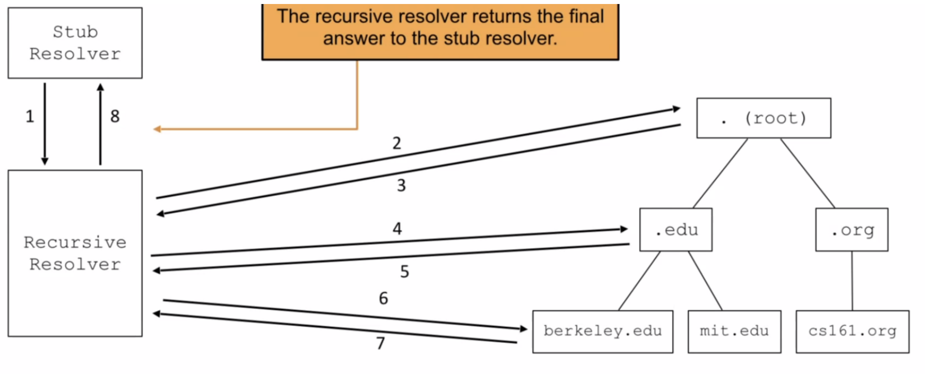
- Name server: A server on the internet responsible for answering DNS requests

DNS Name Server Hierarchy

- If one name server doesn’t know the answer to your query, the name server can direct you to another name server
- Arrange the name servers in a tree hierarchy

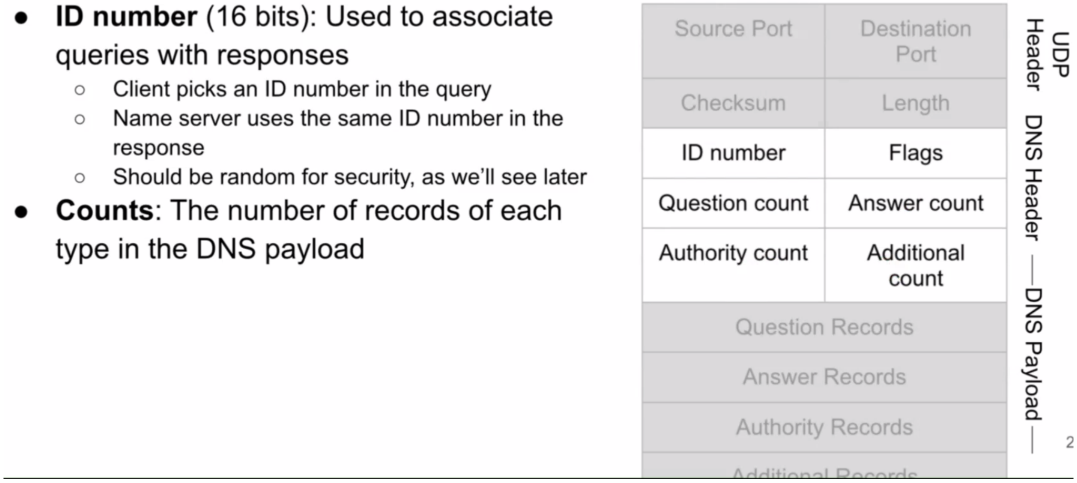
Stub Resolver

- Do a fast DNS lookup
- Cache previous lookups



DNS Message Format

- DNS uses UDP
- Want to talk as fast as possible
- Designed to be lightweight and fast latency



- **ID number** (16 bits): Used to associate queries with responses
  - Client picks an ID number in the query
  - Name server uses the same ID number in the response
  - Should be random for security, as we'll see later
- **Counts**: The number of records of each type in the DNS payload

| | | |
|---|---|---|
| Source Port | Destination Port | UDP Header |
| Checksum | Length | |
| ID number | Flags | DNS Header |
| Question count | Answer count | |
| Authority count | Additional count | |
| Question Records | | DNS Payload |
| Answer Records | | |
| Authority Records | | |
| Additional Records | | |

DNS payload has a variable number of resource records (RRs)

Malicious Cache

- Doesn't work

## Week 12: Lecture 31 DNSSEC (11/8)

DNS Security

- Send a malicious cache records

Securing DNS Lookups

- Want integrity on the response
- Do not confidentiality

DNS over TLS: Issues

- Performance: DNS needs to be lightweight and fast
- Object Security and Channel Security

Channel Security

- Securing the communication channel between two end hosts

Object Security
- Securing a piece of data

DNSSEC (DNS Security Extensions)
- Extension of the DNS protocol that ensures integrity on the results
- Backwards-compatible

Idea: 1: Sign Records
- Digital signatures provide integrity
- Digital signatures

Idea 2; Public Key Infrastructure (PKI)
- Name server are arranged in a hierarchy

Idea 3: Constrained Path of Trust
- Web PKI, DNS has constrained path of trust
- . is truncated for everything
- .edu is only trusted for names ending in edu

Idea 4: DNSSEC useless for name records
- DNSSEC for secure distribution of other keys
    - DKIM

Resource Record Sets (RRSETs)
- Group of DNS records with same name and type form resource record set
- Useful for simplifying records

## Week 12: Lecture 32 DNSSEC (11/10)

Nonexistent Domains
- DNSSEC structure works great for domains which exist
- Option 1: Don't authenticate nonexistent domain responses
    - NXDOMAIN responses can be spoof
- Option 2: keep the private key in the name server itself so it signs NXDOMAIN responses but name server have access to the private keys

NSEC: Authenticated Denial of Existence
- Prove nonexistence of a record type
- Prove nonexistence of a domain

Issues with NSEC
- Domain enumeration: easy for an attacker to find every single subdomain of a domain
- It lists all the domains that is in between if cannot find

Issues with NSEC3
- Real way to prevent enumeration is online signature generation wit the private key
    - Take the hash of the name, no exists between H(name) - 1 and H(name + 1)

Offline Signature Generation
- Offline signatures: Application that computes signatures is separate from the app that serves the signatures

- Efficiency: signed ahead of time and signature is stored and served on request
- Security: attacker must compromise the signature generation system

The Time

- Signatures generated in advance
- Keys can be fetched in parallel with the data
- Signature verification should be less than network latency

Implementation error

name-> key records not name-> IP

- Use DNSSEC as a PKI for name->key records
- DKIM: provides cryptographic proof that sending email server says it is from sender

## Week 12: Lecture 33 Denial of Service and Firewalls (11/12)

Availability and Denial of service (DoS)

- Availability: Making a service on the network available for legitimate users
- Denial of service (DoS): An attack that disrupts availability of a service, making it unavailable for legitimate users

DoS Attacks: Strategies

- Exploiting program flaws
    - Software vulnerabilities can cause a service to go offline
- Resource exhaustion
    - Everything on the network has limited resources
    - Attacker consumes all the limited resources so legitimate users cant use them
- Medications for exploitation may result in DoS
- Bottlenecks
    - Identify the bottleneck and overwhelm the bottle neck

Application-level DoS:

- Target the high-level application running on the host
- Network-level DoS: target network
- Exploit features of the application itself

Resource consumption

- Force the server to consume all its resources
    - Exhaust filesystem space (write), Exchausts RAM (malloc), exhausts processing threads (fork), Exhaust disk (read/write)

Algorithmic Complexity Attacks

- Attacks chooses inputs that cause worst-time runtime to occur
- Attacker must understand the algorithm

Application-level DoS: defenses

- Identification: Distinguish requests from different uses, identify and authenticate users
- Isolation: ensure one user’s actions do not affect another users’ experience
- Quotas: Ensure that users can only access a certain proportion resources

Application-level DoS: defenses

- Proof-of-work: Force users to spend some resources to issue a request
    - Idea: make a DoS attack more expensive for the attacker
- Overprovisioning: allocate a huge amount of resources
    - Content delivery network: a service that allocates a huge amount of resources for you

Network-Level DoS

- Approaches target network protocols to affect the victim's Internet Access
    - Send a huge amount of packets to the victim and overwhelm bandwidth
    - Overwhelm the victim's packet processing capacity

Distributed Denial-of-Service (DDoS)

- Use multiple systems to overwhelm the target system
- Botnet: collection of compromised computers controlled by one attacker
    - Attacker can tell all the computers on the botnet to flood a given target

Amplified Denial-of-Service

- Use an amplifier to overwhelm the target more effectively
    - Idea: some services send a large response when sent a small request

Network-level DoS: Defenses

- Packet filter: discard any packets that are part of the DoS attack
    - Packet filter needs to be before the bottleneck
- Overprovisioning: purchase enough networking bandwidth and equipment to make it harder for attackers to overwhelm the network
- Content Delivery Networks
    - Redirect requests to the "closest" CDN server
    - Caches all content possible

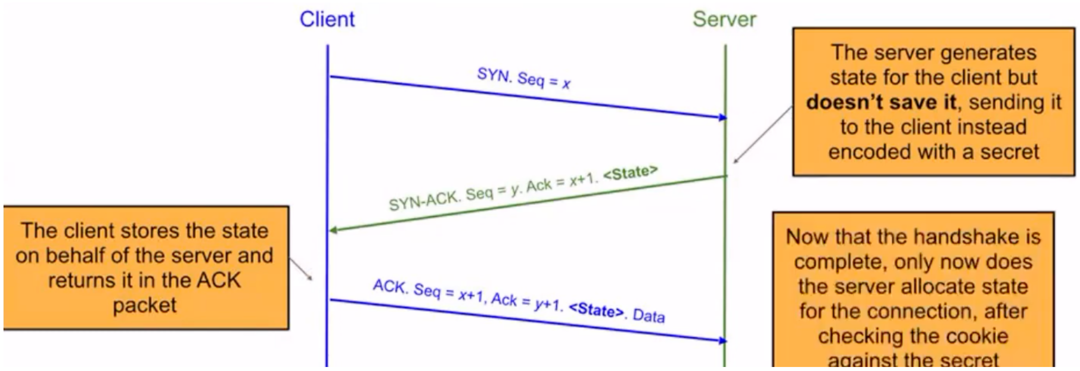
## **Week 13: Lecture 34 Denial of Service, Firewalls, and Intrusion Detection (11/15)**

SYN Flooding and Syn Cookies

- Huge number of SYN, server allocates a lot of space and runs out of memory

SYN Flooding: defenses

- Overprovisioning: ensure the server has a lot of memory
- Filtering: ensure that only legitimate connections will create state
- SYN cookies: don't store state


Client
Server
SYN. Seq = x
SYN-ACK. Seq = y. Ack = x+1. <State>
ACK. Seq = x+1, Ack = y+1. <State>. Data

The server generates state for the client but **doesn't save it**, sending it to the client instead encoded with a secret

The client stores the state on behalf of the server and returns it in the ACK packet

Now that the handshake is complete, only now does the server allocate state for the connection, after checking the cookie against the secret

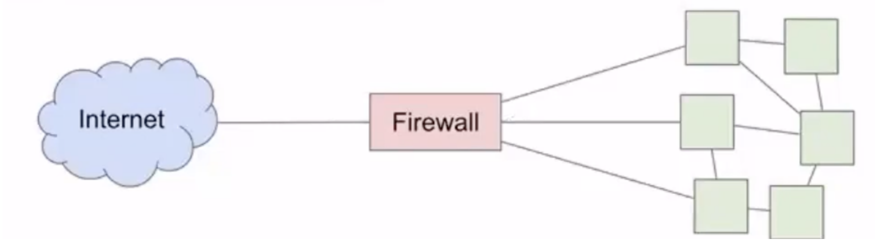
Practical SYN Cookies

- Encode state with a secret and send it to the client who will return it when needed, HMAC(k, {IPs, Ports, client ISN}) -> Server's ISN

Motivation: Scalable Defenses

- More network services = more risk

Firewalls and Security Policies



- Add a single point of access in and out of the network with a monitor
    - Ensure complete mediation
- Network access is controlled by a policy

Policy of a standard home network

- Outbound policy: allow outbound traffic
- Inbound policy: only some traffic is able to enter the network
    - Allow inbound from outbound, deny all inbound

Default Security

- Default-allow policy: allow all traffic, but deny those on a specified denylist
    - More flexible, vulnerabilities
- Default-deny policy: deny all traffic but allow those on a specified allowlist
    - More conservative, flaws less painful
- Default deny generally best

Stateless Packet Filters

- Firewalls are packet filters, inspect network packets and chooses what to do with them
- TCP: implemented with a hack
    - Allow inbound traffic with the ACK flag set
    - Deny inbound traffic without an ACK flag set

Stateful Packet Filters

- Keep state in the implementation of the packet filter
    - Filter keeps track of inbound/outbound connections, rules define what is allowed
- Stateful packets filters can also track the state of well-known applications
    - Decoding and tracking HTTP requests/responses
    - Tracking files sent in an FTP connection

Subverting Packet Filters

- IP packets have a time to live (TTL)
    - Number of hops

Other Types of Firewalls

- Proxy firewall: instead of forwarding packets, form two TCP Connections, one with source one with destination
- Application proxy firewall: certain protocols allow for proxying at the application layer

Alternative for allowing firewall traffic

- Virtual Private Network (VPN) set of protocols that allows direct access to an internal network via an external connection

Firewall Pros and Cons

- Pros
    - Centralized management of security policies
- Cons
    - Reduced network connectivity
    - Vulnerability to insiders

Firewalls

- Defend many devices by defending the network by a packet filter, either forward or drop packets, proxy creates connection to both sides

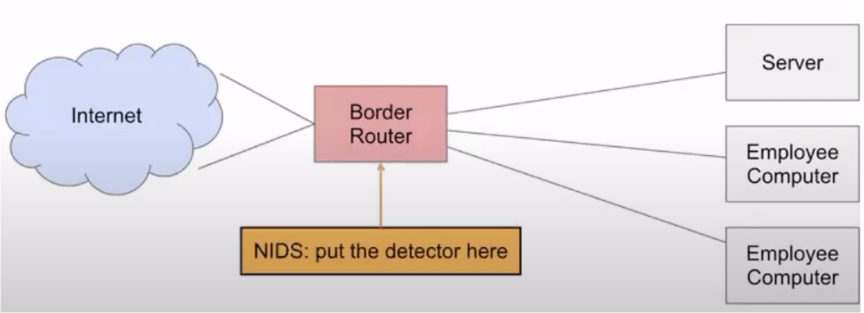
Onward: Intrusion Detection

- Path Traversal Attacks
    - Unix file paths
        - File path points to a file or a directory

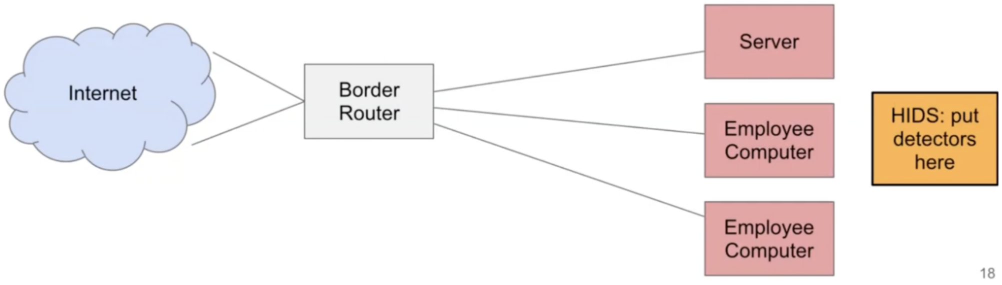
Types of Detectors

1. Network Intrusion Detection System (NIDS)
    a. Installed on Border Routers on the network between the network and rest of the internet
    b. Table of all active connections and maintains state for each connection
    c. Pro: Cheap, easy to scale, add computing power to the NIDS, simple to manage, smaller computing base
    d. Drawbacks: inconsistent and ambiguous transition,
    e. Evasion attacks: exploit inconsistency and ambiguity to provide malicious inputs that are not detected by the NIDS

f. Encrypted traffic: give NIDS access to private keys, can decrypt messages to inspect them for attacks



## Week 13: Lecture 35 Denial of Service, Firewalls, and Intrusion Detection (11/17)

2. Host-based Instruction Detection System (HIDS)
    a. A detected installed on each end system
    b. Benefits: fewer problem with inconsistencies or ambiguities, works for encrypted messages, protect against non network threats, performance scales better than NIDS
    c. Drawbacks: expensive: install fo reach host, evasion attacks still possible
    d. 

3. Logging
    a. Analyze log files generated by end systems
    b. Benefits: cheap, built in logging systems, fewer problems with inconsistencies
    c. Drawbacks:
        i. no real-time detection, detected after the attack has happened,
        ii. some evasion attacks still possible,
        iii. attack can change the logs to erase

Detection Errors

- Two main types of detector errors
    - False positives: detector alerts when there is no attack
    - False negative: Detector fails to alert when there is an attack
    - False positive rate (FPR): detector alerts, given no attack
    - False negative rate (FNR) prob detector does not alert, given there is attack

Detection tradeoffs

- Achieving an effective balance between false positives and false negatives
    - Quality of the detector depends on the system you're using it on

Signature-based Detection

- Flag any activity that matches the structure of a known attack
- Blacklisting: keep a list of patterns that are not allowed
- Path detection exploits
- Benefits:
    - Conceptually simple,
    - Good at detecting known attacks
    - Create vulnerability signatures
- Drawbacks
    - Can't catch attacks without known signatures

Specification-based Detection

- Specify allowed behavior and flag behavior that isn't allowed
- Whitelist approach
- Benefits:
    - Detect new attacks we never seen
    - Low positive rate
- Drawbacks
    - Takes lot of time and effort

Anomaly-based detection

- Attack look unusual
- Develop a model of what normal activity looks like and alert for deviation from normal
- Benefits
    - Detect new attacks
- Drawbacks
    - Can fail to detect known attacks
    - What if model is trained on bad data

Behavioral Detection

- Look for evidence of compromise
- Look for exploitation
- Benefits
    - Detect attacks we haven't seen before
- Drawbacks
    - Legit processes could perform the behavior as well
    - Only finds after access

Vulnerability Scanning

- Launch attacks on your own system first and add defense against any attacks that worked

- Use a tool that probes own system with wide range of attacks and fix any successful attacks
- One common form is red teaming
- Benefits
    - Accuracy: if scanning tool is good find real vulnerabilities
    - Proactive: Prevents attacks before they happen\
    - Intelligence: if intrusion detection system alerts on an attack you know already fixed
- Drawbacks
    - Can take a lot of work
    - Not helpful for systems you can't modify
    - Dangerous for disruptive attacks

Honeypots

- A sacrificial system with no real purpose
    - No legitimate systems ever access the data
- Benefits
    - Can detect attacks we haven't seen before
    - Can analyze the attacker's action
    - Distract the attacker
- Drawbacks
    - Need to trick attacker into accessing the honeypot

Forensics

- Analyzing what happened after a successful attack
- Digital forensics and incident response (DFIR)
    - Need detailed logs os system activity
    - Tools for analyzing

Blocking: intrusion Prevention system

- Intrusion Prevention system (IPS) blocks attacks
- Attacks on intrusion detection system (IDS)
    - System with limited resources vulnerable to DoS attacks

Detection summary

- Types
    - NIDS
    - HIDS
    - Logging
- Intrusion detection
    - Signature based
    - Specification based
    - Anomaly based
    - Behavioral

## Week 13: Lecture 36 Malware (11/19)

Malware:

- Malicious software: attacker code running on victim computers
    - Code that deletes files, sends spam

Self-replicating code:

- Code snippet that outputs a copy of itself
- Can be used to propage malware

Viruses and Worms

- Malware that automatically self-propagate
- Virus: code that requires user action to propagate
- Worm: code that does not require user action to progate

Propagation strategies

- Infect existing code that will eventually be executed by the user
- Modify the existing code to include the code

Arms Race

- Attackers look for evasion strategies
- Attackers have slight advantage in being able to see detection strategies

Polymorphic Code

- Each time the virus propagates, it inserts an encrypted copy of the code
- Produce different looking output on repeated encryptions

Encryption schemes produce different-looking output on repeated encryptions

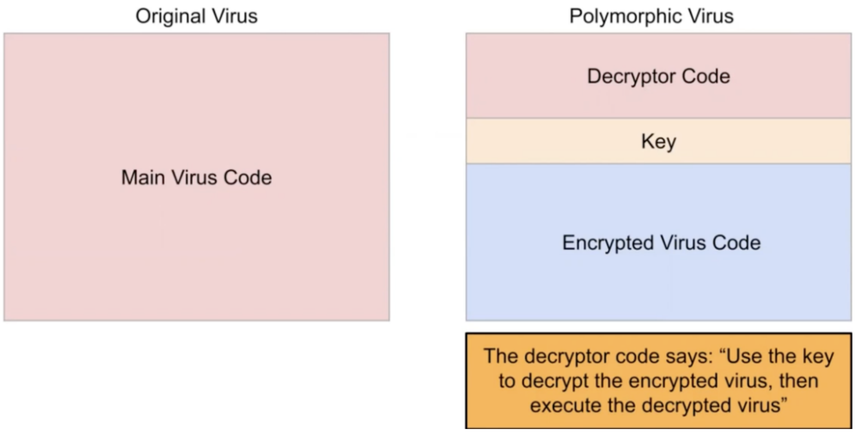
Encryption is being used for obfuscation

Polymorphic code: defenses

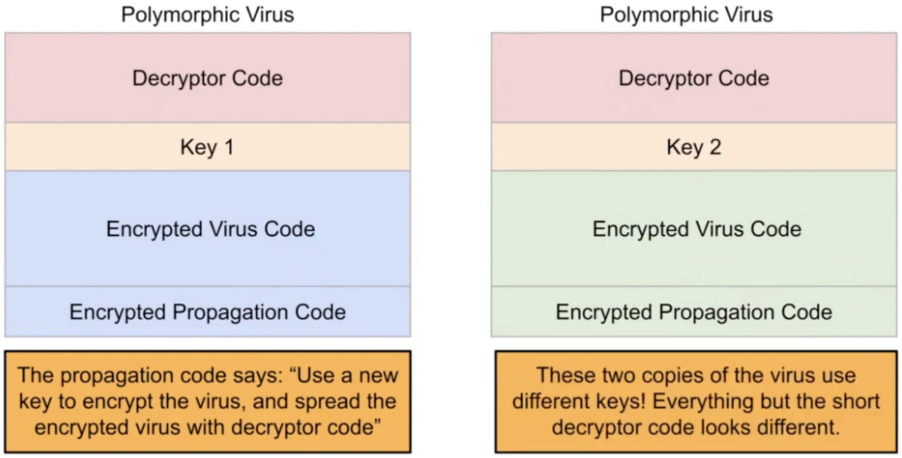
- Add a signature for detecting the decryptor code
- Safely check if the code performs description

Metamorphic Code

- Each time the virus propagates it generates a semantically different version of the code
- Include a code rewriter with the virus to change the code randomly each time



The decryptor code says: "Use the key to decrypt the encrypted virus, then execute the decrypted virus"



The propagation code says: "Use a new key to encrypt the virus, and spread the encrypted virus with decryptor code"

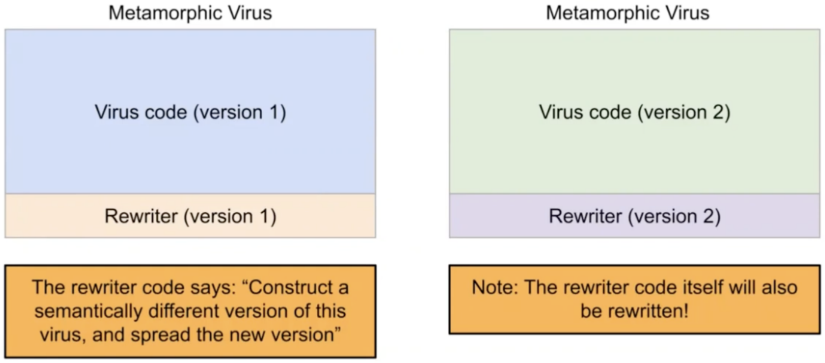
These two copies of the virus use different keys! Everything but the short decryptor code looks different.

Metamorphic Code: Defenses

- Behavioral detection
- - analyze behavior instead of syntax
- Subverting behavioral detection

Defense: Flag unfamiliar code

- Antivirus software instead looks for new, unfamiliar code
- Flagging unfamiliar code is a powerful defense
- Detector for malicious behavior



The rewriter code says: “Construct a semantically different version of this virus, and spread the new version”

Note: The rewriter code itself will also be rewritten!

Worms

- Code that does not require user action to propagate
- Generate random 32 bit IP address and try connecting to it
- Worms spread faster because they parallelize spreading and can be modeled as an infection epidemic
- Spread of the worm depends on
  - Size of the population
  - Proportion of population vulnerable to
- Number of infected grows logistically

## Week 14: Lecture 37 Anonymity and Tor (11/22)

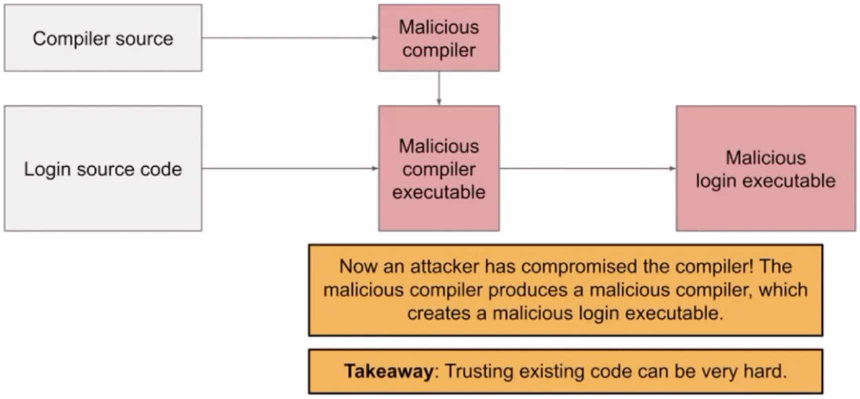
Infection Cleanup

- Might need to rebuild the system, but need to rebuild the system from outside

Reflections on Trusting Trust

Rootkit: malcode in the operating system that hides its presence

- Hide that the malcode is stored on disk
- Malcode currently running in a process



Now an attacker has compromised the compiler! The malicious compiler produces a malicious compiler, which creates a malicious login executable.

**Takeaway**: Trusting existing code can be very hard.

Anonymity

- Concealing your identity
- Anonymous is not confidentiality, confidentiality is content

Anonymity on the Internet

- Hard difficult to achieve on your own
- Easier for the attacker: hack into someone else’s computer and send comm
- Ask someone else to send the message

Proxies

- Alice wants to send a message to Bob
    - Bob shouldn't know the message is from alice
    - Eavesdropper cannot deduce that alice is talking to bob
- Proxy: a third party that relays our internet

Real Use for VPNs

- Evading censorship
    - Local adversary can't see your traffic
- Access control
    - System allow internal access or access from known networks
- Separating idiots from their money
    - Commercial VPN services

Tor

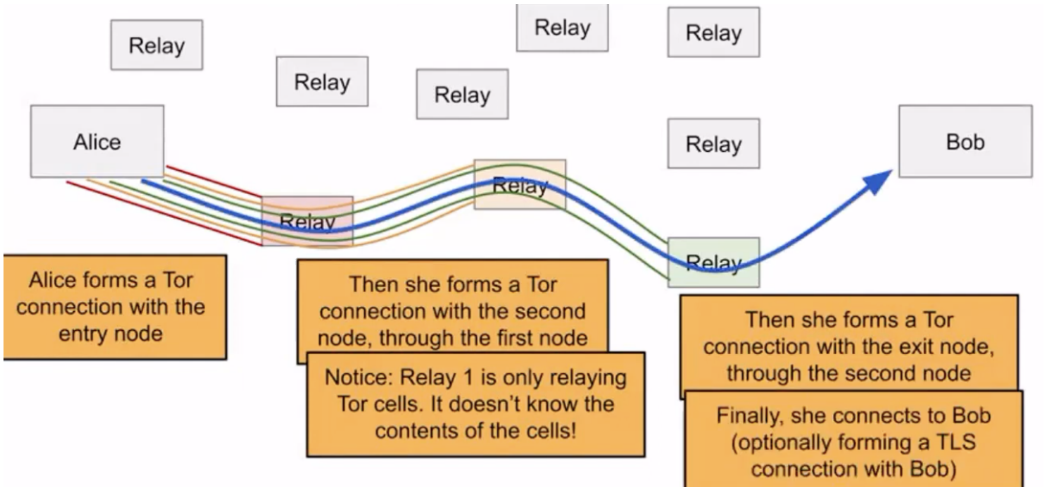
- Idea: send the packet through multiple proxies instead of just one proxy
- Tor: network that uses multiple relays to enable anonymous communications
    - The Onion Router
- Components of Tor
    - Tor network: network of many Tor relays for forwarding packets
    - Directory server: lists all tor relays and public keys

Tor Threat Model

- Security: Client anonymity and censorship resistance
- Performance: low latency
- Tor preserves anonymity against local adversaries
    - On path attacker sees Alice

Tor Circuits

- Communicate anonymously with server, tor client forms a circuit consisting of 3 relays
    1. Query the directory server for a list of relays
    2. Choose 3 relays to form a tor circuit
    3. Connect ot first relay
    4. Connect to the second reality through the first relay
    5. Third relay through second relay
    6. Connect ot the web server


Relay
Relay
Relay
Relay
Relay
Alice
Relay
Bob
Relay
Relay
Relay
Alice forms a Tor connection with the entry node
Then she forms a Tor connection with the second node, through the first node
Notice: Relay 1 is only relaying Tor cells. It doesn't know the contents of the cells!
Then she forms a Tor connection with the exit node, through the second node
Finally, she connects to Bob (optionally forming a TLS connection with Bob)

Tor Exit nodes

- Notice: exit node can see the message and the recipient
- Exit node is man-in-the-middle attacker

Tor Weakness: Timing Attacks

- Network attacker who has a full view of the network can learn that Alice and Bob are talking
- Global adversaries are outside of Tor's threat model and are not defended against

Tor Weakness: Collusion

- Multiple nodes working together and sharing information

Tor Weakness: Distinguishable traffic

- Tor does not hide the fact ahta you are using Tor

## Week 15: Lecture 38 Abuse (11/29)

Facebook messenger only has encrypted chat option on phone app

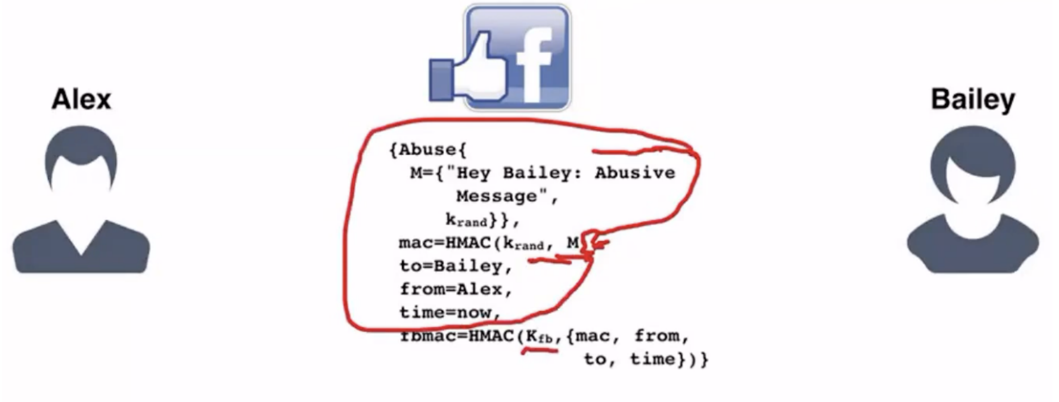
- Alice wants to send a message to bob
- Queries for bob's public key from facebook's server
- Encrypts message and send it to facebook
- Facebook forward to bob where they both use encrypted and authenticated channels to facebook

Facebook's Problem

- Bailey forward the unencrypted message to Facebook
- Bailey does forward the encrypted message to facebook but only facebook should be able to verify
- Trust but verify

Some notes

- Facebook know their HMAC key
- Bailey upon receipt checks that ALex's HMAC is correct



```
{Abuse{
  M={"Hey Bailey: Abusive
       Message",
      krand}},
 mac=HMAC(krand, M
 to=Bailey,
 from=Alex,
 time=now,
 fbmac=HMAC(Kfb,{mac, from,
                  to, time})}
```

## Week 15: Lecture 39 Boeing 737 MAX, Nuclear Weapons (11/29)

The Boeing 737

- Probably the most successful commercial airliner
- Wing mounts to the low part of the plane
- 300-400-500
    - Bigger better , more efficient
    - Engines mounted to the front of the wing
- 600-700-800-900
    - Bigger wings, flat buttoned engines
    - New cockpit for the pilots
- Airbus
    - A320 slightly bigger than a 737
    - Wings are much higher up, clearance to bottom of engine and airfield
    - A320 NEO new engine option
        - Bigger engines
    - Larger engines that burn hotter are much more fuel efficient
- 737 MAX
    - 2011 boeing responded to the A320 bc AA ordered Airbus
    - Fatal Decision 1:
        - Must be no significant pilot retraining, should be able to retrain
    - Fatal Decision 2:larger engine
        - Went to a 61” to 69” engine
        - Forced to move the engines further forward and upward which changes the dynamic balance of the aircraft otherwise reengineering the entire wing setup
    - Fatal Decision 3: software fix
        - If plane goes too nose up wants to stall
        - Modify the computer to have the plane adjust itself so it plays like MCAS (Maneuvering Characteristics Augmentation System)
    - Fatal Decision 4: engineering the software fix
        - Boeing pilot is supposed to be the boss
        - Two flight computers, each only listen to its own set of sensors Computer was an advisor
        - MCAS program suck with the 737 design
        - If computer saw its pitch sensor too high, it would act
        - If you fight the computer on 737 NG, computer gives up
        - MCAS, tries again and again
    - Fatal Decision 5: regulatory Capture
        - FAA certified planes
        - Aircraft mostly self certified
        - MCAS determined to create hazardous condition if it erroneously activated at the wrong time but kept single-sensor design

    - Boeing blamed the pilots

Nukes

How a Nuclear Weapon works

- Hollow sphere of fissile material

Problem: when to use nukes

- Launch nukes when you shouldn't
- Fail to launch the nukes when you should

Launch on warning and the US C&C structure

- President has three items
    - “Biscuit” of authentication does kept on his person
    - “Football” containing menu of options for ordering a nuclear strike
    - Encrypted secure phone
- Calls over football, picks option
- Called NORAD phone
- < 5 ICBMs leave their silos

Limiting Use

- Who might use a nuke without authorization
    - Allies where we station our nukes
    - No one without authorization can cause a nuke to go off
- Mandated solution
    - Permissive Access Link (PAL)
- Nuclear Safety Features

Bomb safety systems

- Trusted base insite tamper detecting membrane
- Hard to disable modern nukes

Some more on Tor

- Exit nodes have not always been honest
- ISP does not allow

Censorship Evasion

- Very bad at evading censorship

Dark Market

- Censorship-resistant payment (Bitcoin)(
- eBay-style rating system with mandatory feedback
- Escrow service to handle disputes
- Accessible only as a tor hidden service

Dark Market: history

- Follow template of the original “silk road”
    -

## Week 15: Lecture 40 Security (12/3)

Iphone

- Secure enclave is trusted base for the phone
- Roll of SEP is f SEP are things too important to allow the OS to handle

Apple Pay

- Authenticates to user with face reader
- Handles emulation of the credit card
- “Tokenized” NFC wireless protocol
- Tokenized public key protocol for payments through the app
- Faster more secure privacy sensitive

Cryptographic key exchange by close proximity and the camera

Credit Card Fraud

- Good protections against fraud
- Cost of credit card fraud is cost of recovery from fraud

Existing project: C/C++

- Turn on all compiler OS mitigations
- Add rate limits: auto restart increasing delay
- Look at the continuous integration testing flow
- Add more machines to test infrastructure