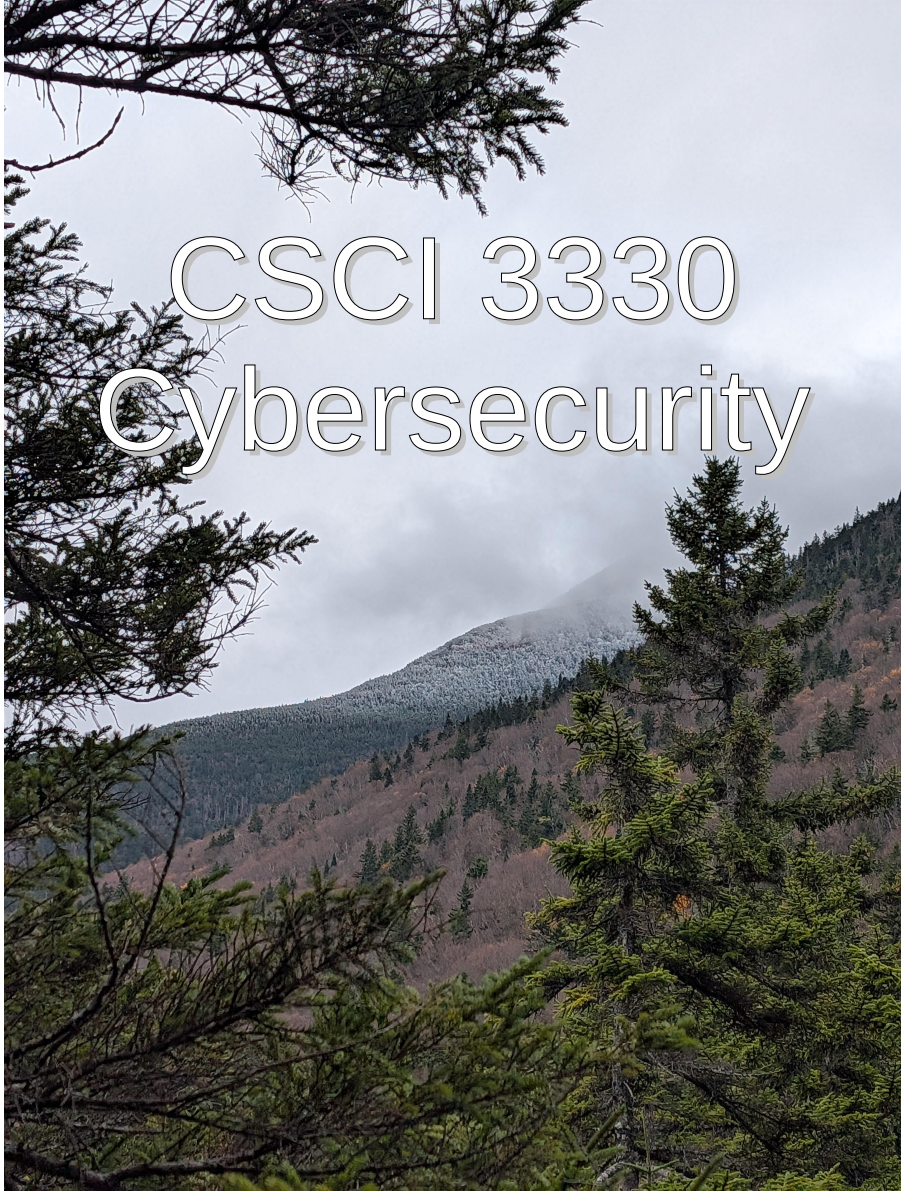

CSCI 3330
Cybersecurity

Word of the day

YET ANOTHER adj.

1. Of your own work: A humorous allusion often used in titles to acknowledge that the topic is not original, though the content is. As in ‘Yet Another AI Group’ or ‘Yet Another Simulated Annealing Algorithm’.

2. Of others' work: Describes something of which there are already far too many.

Agenda for today

- TLS
- FTP
- Memory?

TLS

- Encrypts almost all encrypted traffic on the Internet
- Powers HTTPS, DoT, SMTPS, IMAPS, FTPS, etc.

TLS

- Since the Edward Snowden disclosures, there has been a large push to encrypt all traffic on the Internet
- Almost every web site supports HTTPS these days
- Before, not the case
- Even banks would often only encrypt logins

History

- SSL 1.0 (1994)
- SSL 2.0 (1995)
- SSL 3.0 (1996)
- TLS 1.0 (1999)
- TLS 1.1 (2006)
- TLS 1.2 (2008) (still secure, still most popular…)
- TLS 1.3 (2018) (important but controversial features)

TLS ciphers (examples)

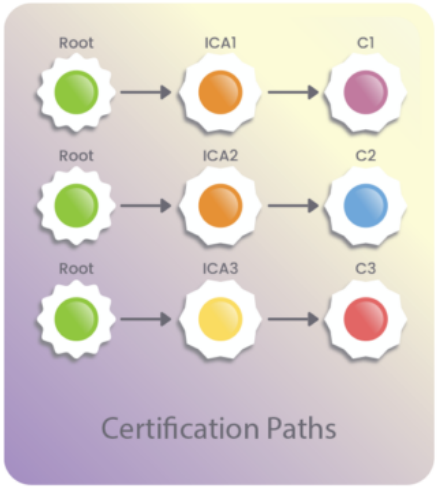
- TLS_RSA_WITH_DES_CBC_MD5
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

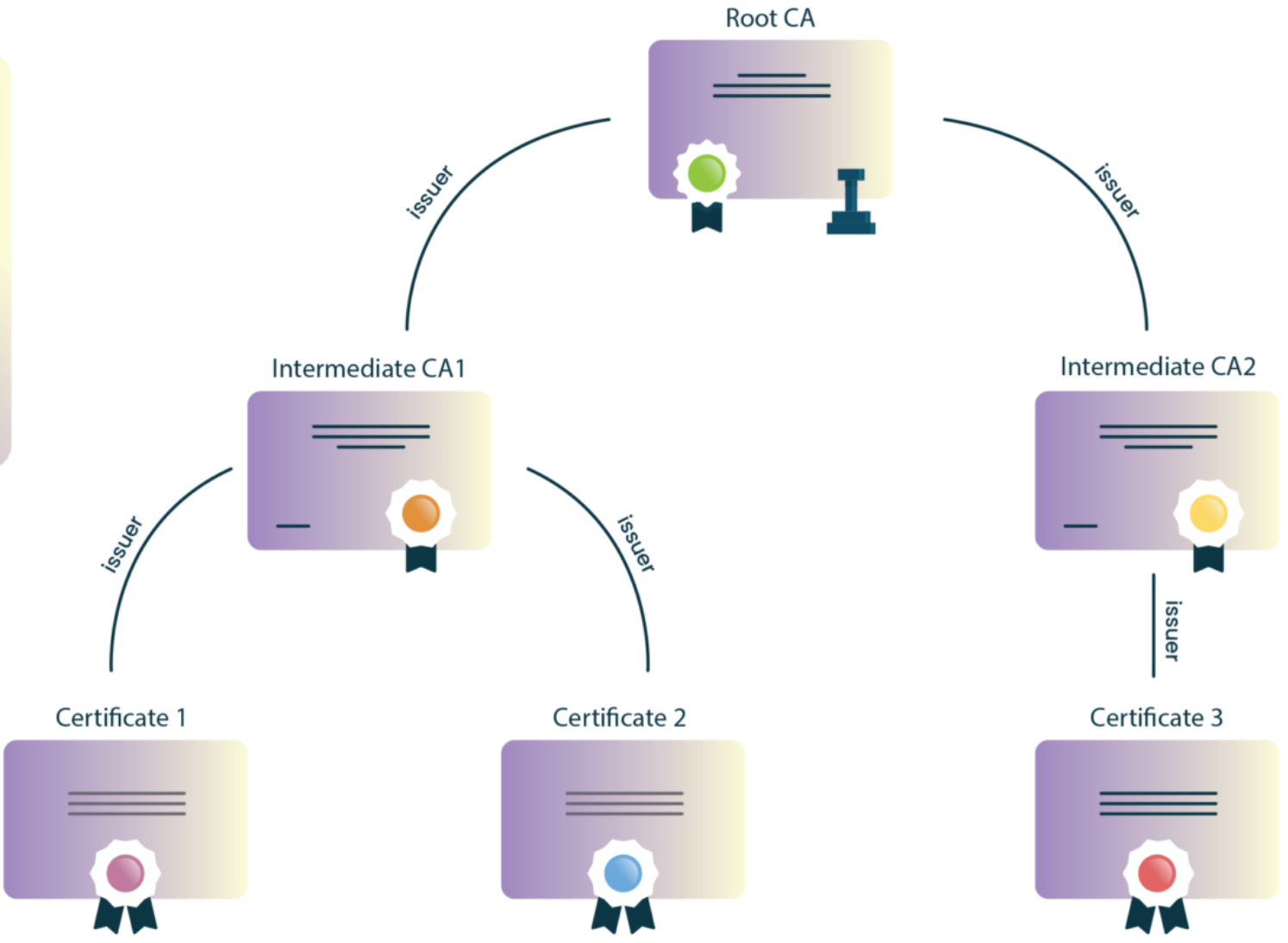
TLS handshake (assuming DH)

- Client sends “Client Hello”:
  - Supported TLS versions and cipher suites
- Server sends:
  - A “Server Hello”:
    - A TLS version and cipher suite from the list
  - An SSL certificate
  - A *signed* “Server Key Exchange” (DH credentials)
- Client *verifies* the certificate

Certificate verification

- Certificate = **a signed, public key**
  - Other metadata (domain, expiration, etc.)
- Signing a public key: a little recursive
- Signed by certificate authority
- Most browsers / operating systems agree on trusted authorities


Root
ICA1
C1
Root
ICA2
C2
Root
ICA3
C3
Certification Paths


Root CA
issuer
issuer
Intermediate CA1
Intermediate CA2
issuer
issuer
issuer
Certificate 1
Certificate 2
Certificate 3

Root CAs

- How do CAs verify *you* own a domain?
- Ask you to upload a file
- Ask you to publish a DNS entry
- LetsEncrypt: automates this process, for free

TLS handshake (assuming DH)

- Client replies with:
  - A “Client Key Exchange” encrypted with the public key in the SSL certificate
- Server and client compute a shared secret using Diffie-Hellman and “server random” and “client random”
- Client *verifies* the certificate

What is *not* encrypted?

- TLS version, cipher suite, SSL certificate, server random
- TLS extension: SNI (Server name include)
- Includes the server name so that a single IP address/port can host multiple sites
- Distinguish sites by domain name
- Unencrypted because whose key would we even encrypt it with?

TLS 1.3 + Encrypted Client Hello

- How can we encrypt hello?
- Include secret, encrypted, “inner” hello
- Encrypt with what? Server’s public key
- How does the client know the key?
- DNS SVCB/HTTPS records

Resistance to TLS 1.3 + ECH

- Many actors want SNIs to be readable
- State (governments who censor)
  - Russia and China block all TLS 1.3 + ECH traffic
- Corporate (controlling what people browse at work)

What does TLS look like?

- On the wire?
- Wireshark!

What is downside of TLS?

- Is there a downside to every web site being encrypted by TLS?
- Requiring recent TLS version to connect?

Transition

- We are done with cryptography unit
- Now we will learn how to hack an FTP server
- This will be a memory exploit
- Completely remote
- Root shell
- We will defeat all modern security measures

FTP

- File Transfer Protocol
- Network protocol timeline
  - FTP (1971)
  - TCP (1974)
  - IPv4 (1981)
  - DNS (1983)
  - HTTP (1989)

- FTP uses two connections
  - Control (commands): port 21
  - Data (directory listings, file transfers, etc.): port 20
- Our attack will be over the control connection

FTP commands

- HELP – display list of commands
- USER user – used for authenticated login
- PASS password – used for authenticated login
- PWD – print working directory
- CWD dir – change working directory
- CDUP – change working directory to parent

FTP commands

- LIST – list contents of current working directory
- MKD dir – make directory
- RMD dir – remove directory
- DELE file – delete file
- RETR file – retrieve (download) file
- STOR file – store (upload) file

FTP commands

- RNFR – rename from (part 1 in renaming a file)
- RNTO – rename to (part 2 in renaming a file)
- SITE cmd – server-specific commands
- SITE EXEC executable – execute executable
- QUIT – sign off

FTP playground

- https://brainhammer.com/ftpplayground/

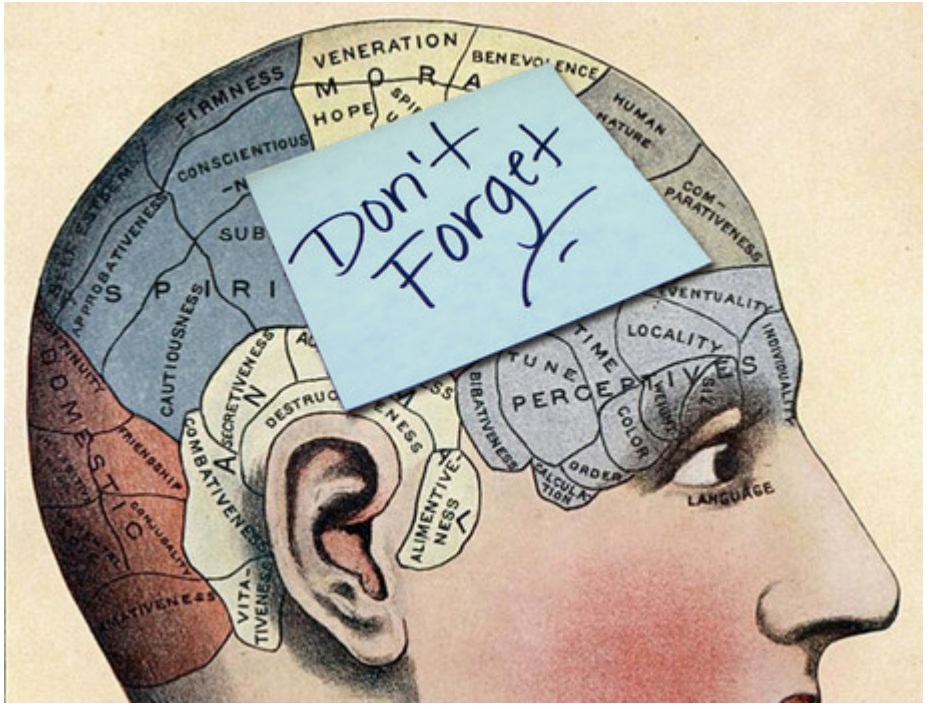
https://brainhammer.com/ftpplayground/

1. List the current directory
2. Enter the “pub” directory
3. Download the file in it
4. Go up a directory
5. Create a directory
6. Upload a file to it
7. Rename that file
8. Then download that file
9. Then delete that file
10. Then delete its directory
11. Execute whoami
12. Sign off

FTP not as common these days

- Not secure by default
  - Although can be wrapped in TLS (FTPS)
- Replaced by WWW (http)
- In what way is WWW better? In what way is it worse?

Memory

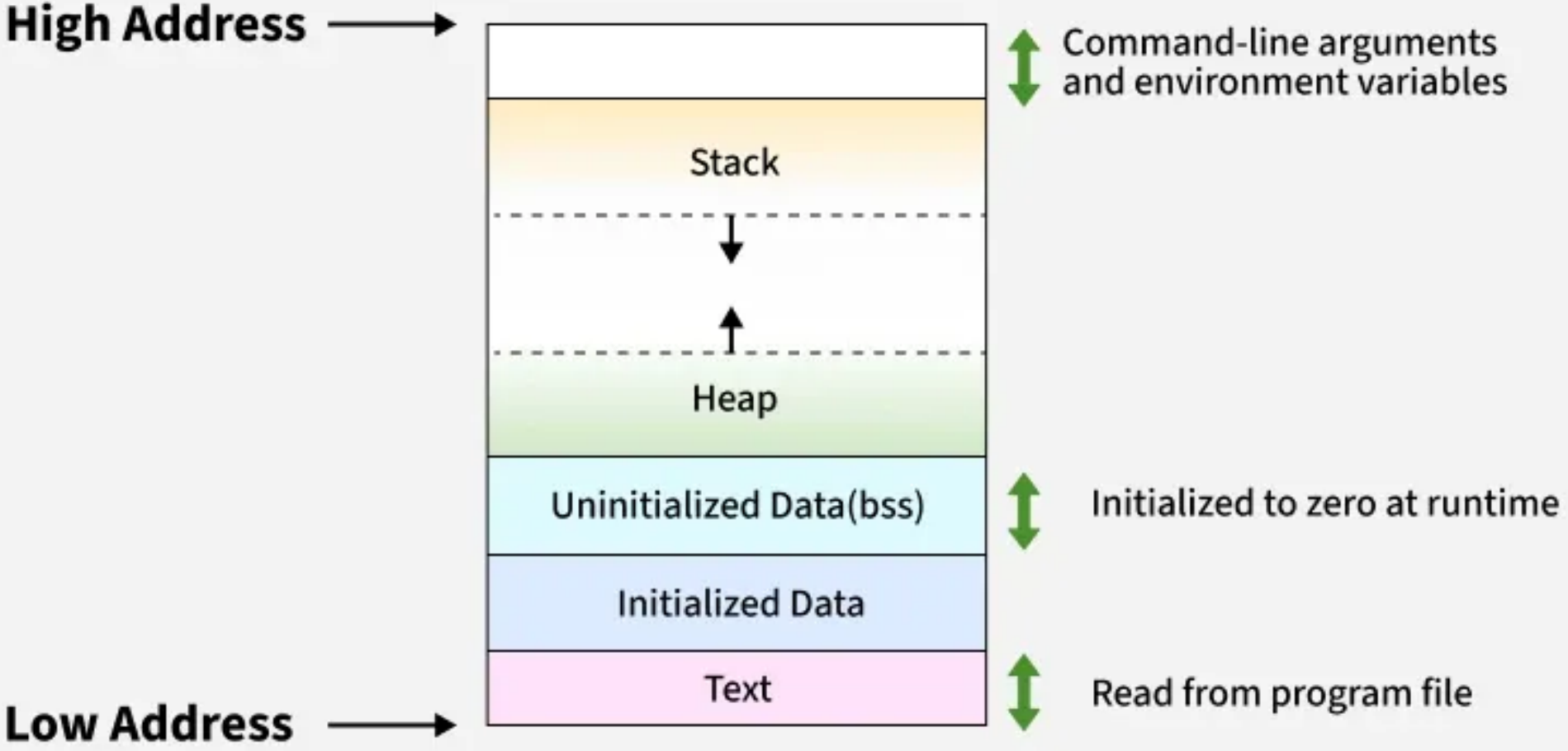

VENERATION
FIRMNESS
HOPE
HUMAN NATURE
CONSCIENTIOUS
SUB
CAUTIOUSNESS
LOCALITY
COLOR
ORDER
LANGUAGE
COMBATIVENESS
DESTRUC
DOMESTIC
FRIENDSHIP
ALIMENTIVE-NESS
Don't Forget

Physical vs. virtual memory

- Only one process mapped into memory at a time
- Unless you are dealing with kernel exploits, your exploit will be working with virtual memory

Virtual memory

- OS maintains page table
- On some architectures (e.g., x86)
  - The CPU walks the page table (faster)
  - But the OS must maintain it very precisely
- On others
  - The OS walks the page table (flexible)
  - The OS can maintain the table however it wants


High Address
Command-line arguments
and environment variables
Stack
Heap
Uninitialized Data(bss)
Initialized to zero at runtime
Initialized Data
Text
Read from program file
Low Address

Steps to compile a C program

- Preprocessing
- Compiling
- Assembling
- Linking

Libraries

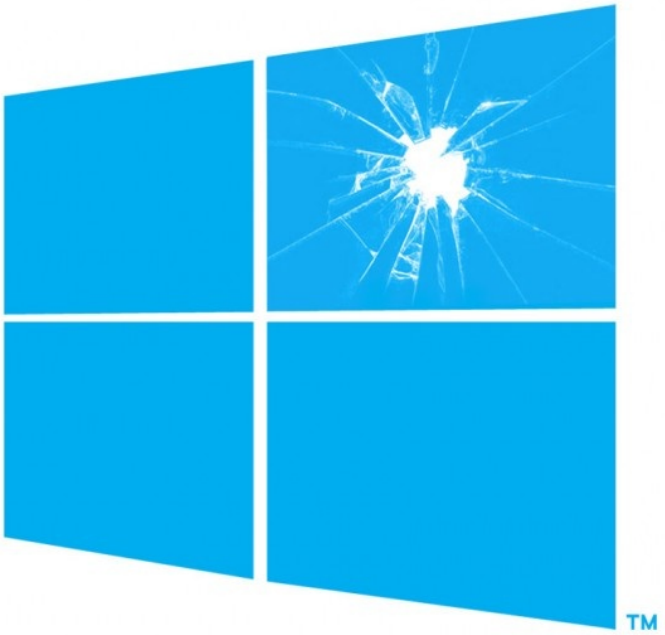
- Collection of functions (and data)
- Static linking: linked at compile time
  - Advantage: don’t need additional libraries at runtime
- Dynamic linking: linked at runtime
  - Advantage: can update libraries independently
  - Saves space and memory

Dynamic libraries

- *.so (Linux)
- *.dylib (macOS)
- *.dll (Windows)

Dynamic libraries

- *.so (Linux)
- ~~*.dylib (macOS)~~
- ~~*.dll (Windows)~~



Dynamic linking

```
$ ldd /usr/bin/whoami
linux-vdso.so.1 (0x00007ffed67fa000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x000072fc4a000000)
/lib64/ld-linux-x86-64.so.2 (0x000072fc4a2ce000)
```

Dynamic linking

- Who loads the dynamic linker?
- The kernel does
- The dynamic linker (ld.so) is loaded as the program’s “interpreter”

Global offset table (GOT)

- Entry in table for every dynamically linked function
- Each entry initially they point to code in the PLT

Procedure Linkage Table (PLT)

- The entry for each dynamically linked function contains executable code to jump into resolver
- Resolver (in ld.so) finds the address of the function
- Updates GOT to point away from the PLT to the actual function (so future calls call it directly)
- Finally, jumps to actual function

Assignment 7

- Deadline: November 18, 2025, 10:00pm
- Textbook RSA attacks
  https://tildesites.bowdoin.edu/~j.knockel/csci3330f25/assignment7.pdf