Technical Security Concepts - ivan notes 2022

1. Compliation & Interpretation

- compilation
  - a special program to convert your english-like code into 1s and 0s for the computer to understand (binary), special program = compiler
  - common for many languages - C, C++, java, C#, and all compiled languages
  - this is a two step process to actual code execution - we must first compile, then execute
  - *if there are any problems with the compiling process, we cannot execute any code
- interpretation
  - a program which concerts intrsuctions into executable form as it is running is called an interpreter
  - many modern langs use interpretation in which decision about how to store things in memory or perform ops are decided during execution
  - very common for many langs - js, Ruby, Bash, Python, and all interpreted langs
  - an interpreted lang will run until you hit a bug. This can make code that doesn't execute frequently difficult to debug
- *note
  - often times the env in which the code is executed can determine whether it is compiled, interpreted, or a little of both
- Python interpreter
  - -many interpreters for common langs come packaged with the OS
    -when we write a script on our machine we can use the python command paired with the file to execute to run this file
  - instead of always writing python3 to execute a particular file, we can specify the interpreter that should be used as the first line within our file

    #!/user/bin/env python3
    #!/usr/bin/bash

    chmod +x my_file.py

- Running an executable
  - by running an exec we can take arguments, this will make our script dynamis and be able to take in variable info
  - we will be able to take in values (arguments) from the CLI, space separated, which could represent values or even paths to directories of files
- Summary
  - all code must be 1s and 0s for the computer to understand it, but there are options such as compilation and interpretation for how it gets there
  - we can define which interpreter for our kernel to use in the 1st line of our script
  - we can use our file as an executable and even take in/expect other arguments from the CLI

2. Networking & Web 1 (intro)

- Clients vs Servers
  - clients make requests
    - our browser gives us this capability through our url bar, curl is also an http client
  - servers generate responses
    - servers interpret the request that is made and generate an appropriate response, often time these responses include all of the content necessary to render our web browser
- rendering a webpage
  - we need 3 major pieces
    - HTML - the actual content of the webpage
    - CSS - the layout/styling of a webpage
    - js - the interactivity/logic of the webpage
- The internet
  - basic communication over the internet relies on a client making a request and a server making a response
  - linux http commands
    - curl - make an http request
      wget - fetch and download resources
- DNS

- associate IPs to server names
  DNS resolver > root server > TLD server > example.com > resolver > server
- linux DNS commands
  - host - query DNS for records belonging to host
    dig - query DNS for records and info regarding DNS servers
    nslookup - query DNS for IP address belonging to host.domain.com
- Query Types
  - Forward
    Reverse
    Mail
    Text

- Summary
  - many protocols rely on the comms between 2 devices often referred to as client and server
  - initial comms generally happens between the browser and servers for DNS resolving

3. Data Transmission & Capture

- OSI Model
  - PDNTSPA
    - L5 - L7: Data
      L1 - L4:
      Segments
      Packets
      Frames
      Bits
    - L1 -> L7 = deencapsulation
      L7 -> L1 = encapsulation
- limitations
  - how data is transported from one layer to another is a combo of encoding and physical limitations
  - often data can be(must be) broken into smaller packets called fragments
  - this is one driver behind the port system
- TCP vs UDP
  - TCP

- -reliable
  -connection oriented
  -segment retransmission
  -segment sequencing
  -lots of overhead including 3-way handhsake

- UDP
  - -unreliable
    -not connection oriented
    -packets lost and not retransmitted
    -no packet numbering
    -very little overhead
  - -can take advantage of the fact that most protocols operate over TCP/IP by using nc
    -can attempt to make a connection with a port using nc and if we get a handhsake, we know that port is open
    -can also use nc to create a listener

- Security implications
  - lack of Confidentiality - maybe we can sniff packets?
    lack of Integrity - maybe we can spoof packets?
    Affect Availability - was can DOS
  - packet sniffing - as long as that data is an encoding format we can understand, we can piece it together fairly easily
    -can use tools like Wireshark to listen on our network and observe traffic

- Wireshark
  - Follows the stream of packets
    right-click on the packet and choose which stream to follow (TCP or HTTP)
  - Open PCAPs
    open from within Wireshark
    on cmd line
    wireshark <path-to-PCAP>
  - -Filter for Traffic
    display vs capture filters
    -Export objects
    File - > Export Objects -> HTTP

4. Encoding

- overview

- -to convert data from one format to another
  -not inherently a security concept
  -most important part of encoding is that our scheme is standardized
  -at the bottom line, all data must be represented by 1s and 0s

- ASCII
- UTF 8
  - -backwards compatible with ASCII
    -expandable past 8 bits
- Hexadecimal (Hex)
  - 4 bit / hex
- Base 64
  - designed to carry data stored in binary formats across channels that only reliably support text conent
  - can be used on the web to embed images or files or other binary assets inside html and css files
- URL encording
  - -made up of reserved and unreserved chars
    -reserved chars = control characters
  - automatically done by browser when we make a request, but often other HTTP clients will not automatically do this
  - any reserved or chars outside the unreserved set must first be encoding with a % sign and then their hex code
- Python encoding/decoding
  - binascii library
  - binhex and ascii are wrapper libraries, there are also additional libs such as base 64
- Summary

- -Base64 base64 Character set: A-Z 0-9
  -Ascii Character set: 0-9
  -Hex xxd Character set: 0-9 A-F \0x00 \0xFF
  -Binary python
  -URL Encoding
  -Difference between encoding, encryption, & hashing
  -Encoding: transferring data
  -Encryption: secrecy
  -Symmetric - one key need to safeguard the key
  -Asymmetric - public and private key used in HTTPS

5. Cryptography

- encryption
  - -hiding msg using the power of math
    -used for comm between parties where secrecy is needed
    -the hidden msg, cipher, should be able to be shown to anyone without revealing the actual encryption
    -reversible process
  - plaintext <> scheme <> ciphertext
  - symmetric
    - both parties have copy of the same key, used to encrypt and decrypt
  - asymmetric
    - one key for encrypting and one for decrypting
    - Key gen reqs:
      -there must be both a public key and private key
      -public key can be given to anyone and allows anyone to lock things up for you
      -private key is your own
    - RSA
      use modular math
      - -product of two large numbers (and some math) will be our encryption key
        -the two large numbers themselves will be our decryption key
    - confidentiality: if someone encrypts a msg for me using my pub key, i can decrypt the msg
      integrity: if i encrypt the msg with my priv key, people can decrypt with my pub key (effectively a signature)
    - Diffie hellman - often used to exchange future symmetric keys
    - SSL/TLS using RSA

- client msg's server to initiate SSL/TLS comm > server sends back an encrypted pub key/cert > client checks the cert, creates and sends an encrypted key back to server (if cert is not ok, comms fail) > server decrypts key and delivers encrypted content with key to client > client decrypts the content completing the SSL/TLS handshake

- Cipher Types
  - Block
    - processing of plain text is done as fixed length block one by one, primarily used for symmetric
  - Stream
    - processing or encoding is done bit by bit, primarily used for asymmetric

- hashing
  - -hiding a msg using the power or math
    -used for storage and/or verification of data
    -finding a particular hash should tell you nothing about the original data
    -irreversible process
  - Good hash key function:
    - -should be slow
      this will hurt attackers more than you
    - -should have low collision frequency
      fewest possible duplicates
  - - Know the popular hashing functions
      MD5 , Sha1, SHA256, etc.
      `md5sum`
      `shasum`
- Encoding: Transferring data
  Encryption: Confidentiality
  Hashing: Integrity

6. Web 2: Verbs

- HTTP review
  - every request gets exactly one (total) response (sometimes a response is broken up into chunks)
- HTTP Req/Res structure
  - VERB / URI
    GET / books HTTP/1.1
    [headers]

- HTTP Verbs
  - GET - read
    POST - create
    PUT - update
    DELETE - delete
    - GET
      -to read content
      -only req we can control in browser without a special form, extensions, or coding knowledge
      -NO body for a GET req
      -can only provide content req info through two means: the path and the query params
    - POST
      -request to create content
      -common type of request on signup pages, forum posts, comment posts, status posts, any time you are creating content
      -rely on the path of a URI to designate to the server what kind of content is being created
      -most important info about a post request is included in the body of the request

      body
      bookID=1234@author=corey&content=this%20is%20a%20story...
    - PUT
      -request to modify/update conent
      -on login pages, eit posts/comment, edit profile info
      -rely on the path of a URI to designate to the server what type of content and the specific content being modified
      -most important info in the body
    - DELETE
      -request to delete content
      -generally no body for delete request
      -generally target a resource using only the URI
      -authentication and authorization are essential
- HTTP Response
  - Common Status Codes

- 200 - OK
  201 - created
  304 - cached
  400 - bad request
  401 - unauthorized
  404 - not found
  500 - server error

- Summary
  - the verb for a request designates the type of action a client is trying to take, is it the servers responsibility to deal with this properly
  - the status code on a response designates how a server handles the request, 200 and 404 are the most common
  - the four major types of requests are GET, POST, PUT, DELETE

7. Web 3: Persistence

- HTTP
  - -layer 7 protocol
    -specifies allowable metadata and content of messages
    -does NOT specify how messages are transmitted
    -STATELESS: does not remember the previous request/response cycle
- AAA
  - Authentication - the person is who they say they are
  - Authorization - person is allowed to do X, Y, Z
  - Accounting - what did the person actually do
- Cookies & sessions
  - no need to authenticate for every request
  - small files, often including unique identifiers that web servers send to browsers. These cookies then can be sent back to the server each time your browser requests a new page. It's a way for a website to remember you, your preferences, and your habits online
  - -used to maintain state
    -usually a random string of chars
    -sent in the header (know how to manipulate cookies in requests)
    -curl -b or edit directly in DevTools under Application Tab in Chrome
  - User-Agent

- -tells the server what kind of HTTP Client is making the request (ex: mozilla, curl, googlebot...)
  -find it under DevTools, under the Network tab, in the Headers section

8. App Services & Protocols

- Basic tools
  - netcat
    - nc
      used for just about anything involving TCP, UDP, or other sockets
  - telnet
    - teletype network
      plain txt comms
      Port 23
- Common protocols
  - FTP - port 21
    SMTP -port 25
    SSH - port 22
    HTTP - port 80
    - FTP - look for default logins or anonymous FTP
    - SMTP - look for enumerating users/valid email addresses
      Send/receive email
    - SSH - encrypted tunnel between machines
      look for keys or access to authorized keys
  - GIT - version control system
    - keeps history of files and commits
      set up SSH keys with GitHub
- Layer 7
  - supports end-user applications and processes. This layer is closest to the end user and is wholly application-specific.
- Summary
  - app layer is user facing
  - tons of protocols, many can be expiated
  - internet was not designed with encryption

9. Windows CLI

- cmd.exe

- -case sensitive
  -run as admin - elevate privileges
  -help <command>
  -<command> /?
  -flags use '/' instead of '-'
  -cls = clear

- Navigation
  - -Drives - 'C'
    -cd - change directory
    -dir - list files
  - dir
    /a - show hidden files - can also be used to restrict types of files display
    /p - show files one screen at a time
    /q - show file ownership
  - cd <directory> or chdir <directory>
    -uses '\' (backslash) as the path delimiter
    -drives are treated as separate hierarchies, switch drives by typing the letter, e.g. "D:" or using the /d flag
- create/mod/del
  - new file
    type nul > file.txt
    echo foo > file.txt
  - file contents
    type file.txt
  - mv/cp
    move (src) (dst)
    copy (src) dst)
  - delete
    del file.txt
    rmdir directory
- Search
  - 'find' is the Windows 'grep'
    -must use quotes for the search term
    - find "file" file.txt
- Mgmt
  - 'net <option>'
    lets you manage different system settings

- <option> - the subsystem you want to manage
- net [accounts | computer | config | continue | file | group | help | helpmsg | localgroup | name | pause | print | send | session | share | start | statistics | stop | time | use | user | view
- net subsystems
  -User/Group - used to add, delete, and manage global users/groups on servers
  -Share - used to create, remove, and otherwise manage shared resources (folders) on the computer
  -Start/Stop - used to start/stop Windows services

10. Linux

NC

- create a listener
  nc -nlvp <port number>
- connect to a listener
  nc <url> <port>

SSH

- -Remote Shell hosted on a server
  -Encrypted
  -Syntax to connect to an SSH Server:
  ssh <username>@<server-address>

SCP

- -Copy data to / from a server on a particular port
  -Copying data to a server:
  scp <path-to-file> <username>@<server-address>:<destination-on-server>
- -Copying data from a server
  scp <username>@<server-address>:<location-on-server> <destination-on-host>

Python

- -Serving up Files with Python
  -Navigate to the directory you want to host
  -Start the python server module:
  python3 -m http.server <port>