Meltdown & Spectre

Tuesday, August 28, 2018 10:59 AM

Meltdown

- Takes advantage of out of order executions
    - Breaks down barrier where process' have isolation of memory
    - Software problem
- Deals w/ how we switch processes
- If order of execution is known, what is being retrieved can be changed
- As long as program is in execution, we can access data we shouldn't be able to access
- Ex: Credit card info on website
    - Info should be protected, but another program could access/steal it
- Fault in how OS' work; hard to fix (branch prediction)

Spectre

- Hardware based problem
- Allowed modification of memory assuming we know the pattern in which a process executes
    - Can write memory in place we don't have/own
- Problem with pre-fetching
    - Pre load into fast memory; no validation to overwrite memory b/c assumption that data is valid

Chapter 1

Thursday, August 30, 2018 11:18 AM

<u>CIA Triad</u>

- Confidentiality
  - Secret info is kept secret
  - Hard to get into
    - Multi authentication
    - Strong passwords
- Integrity
  - Info that is available is accurate and not tampered w/
  - Info verified by multiple sources
  - Secure functions to change data
  - Increasing abstraction
    - By abstraction, more things have access to info; damages confidentiality
- Availability
  - Info is not useful if can't be accessed
  - i.e. social media, school website
- Increase in 1 leads to decrease in others
  - Emphasis of 1 of these leads to problems

<u>ATM</u>

- Ex. 1
  - High Priority
    - Confidentiality
      - Password instead of PIN
        - Higher entropy
          - PIN = 10^4
          - Password & Security requirements
            - Lower & upper case
            - Length
            - Biometrics
            - Cards w/ chips
      - Limit # of ATMs, use only specific ATMs
  - Medium Priority
    - Integrity
      - Regular maintenance / check integrity
  - Low Priority
    - Availability
      - Low # of available ATMs
  - Expensive option; people won't want to use
- Ex. 2
  - High
    - Availability
      - Less staff pay
      - By using ATM, not using person
  - Medium
    - Integrity
      - Frequent validation
  - Low

            - Confidentiality
                - If there's a problem, can be fixed

- Hardware Integrity (Table 1.3)
    - Corrupting a hard drive
    - Hardware gives bad/inaccurate results
    - Intel Pentium Processor v1
        - Measured in MHz (80-160 MHz range)
        - 1997: Math co-processor used for harder math
            - Co-processor had floating point problem
                - Floating point arithmetic is not accurate, and processor gave inaccurate results past the margin of error occasionally

<u>Fundamental Security Design Principles</u>

- Complete Mediation
    - System works by itself but also as part of bigger system
- Open Design
    - Auditable
    - Many eyes seeing system
        - Can catch more bugs
    - Controversial
        - Easier to discover vulnerabilities
- Separation of Privilege
    - Admin accounts shouldn't be used for everyday things
    - Independent security systems
    - No one thing has over reaching privilege
- Least Privilege
    - Only want absolute least amount of privileges possible
- Least Common Mechanism
    - Don't put too much work onto one system
    - Ex: Firewall should only do firewall work
- Psychological Acceptability
    - Something people are willing to undergo to use something
- Isolation
    - Work independently even if working as bigger system
    - Shouldn't be reliant on each other
- Encapsulation
    - Every system should be self contained
- Modularity
    - More control
- Layering
    - Multiple layers of protection
- Least astonishment
    - System should work as expected
    - Not doing anything surprising
    - Should be intuitive

ATM Security Risk

- Easily accessible but have to be physically at it
    - Small attack surface
- Shallow layering
- Medium security risk

- Attack Tree "UT" = Can't do anything about it

Cryptography

Tuesday, September 11, 2018 11:01 AM

- Alan Turing
  - Famous for solving Enigma Machine
    - ~1940s
- Caesar Cypher
  - Take alphabet & shift over letters
    - Shift by 2 = A -> C, B -> D, C -> E
  - Not very strong
  - Total of 26 iterations; not hard to figure out
  - Better to map letters to different, random letters
    - ex: A -> Q, B -> T, C -> L, D -> A

- 2 things that make computers special
  - Crypto
  - Computer games
    - 1st computer game = Space War - MIT
      - Pushed ideas of comps
      - Pushed computer networking: 2 players
- C Lang.?
  - Kernisasan, Ritchie, Thompson
    - Invented UNIX b/c they wanted to play Space War on PDP-5
  - Bell Labs
- Book Cipher
  - Have 2 copies of book, give page number, calculate difference between first letters on page and given cipher
    - Ex: Difference between "The" and "ANF"
- Finite State Machine
  - Basis of computing
  - Computers represent different states, and can change from 1 state to another

<u>Cryptographic Tools</u>
- Symmetric Encryption
  - Univ technique for providing confidentiality for transmitted or stored data
  - Referred to as conventional encryption or single key encryption
  - 2 reqs for secure use
    - Need strong encryption algo
    - Sender & receiver must have obtained copies of secret key in a secure fashion and must keep key secure
  - Every additional key weakens encryption
    - Not recommended to have > 1 key
  - Keeping key secret just as important as message itself
  - Not as secure as other methods
  - Attacking Symmetric Encryption
    - Cryptanalytic Attacks
      - Rely on
        - Nature of algo
        - Knowledge of gen characteristic of plaintext
          - Can eliminate certain things
          - Know must follow some pattern
          - Probably know what you're looking for
          - Limits # of things have to try
        - Some sample plaintext-ciphertext pairs
      - Exploits characteristics of algo to attempt to deduce specific plaintext or key being used
        - If successful, all future & past messages are compromised
    - Brute Force
      - On avg, about 1/2 have to be tried
  - Smaller key size = weaker

- EFF
  - Electronic Freedom Foundation
    - Steve Jackson Games
      - GURPS system
      - Munchkin game
      - Illuminati card game
        - 1980s; ~ time of pc popularity
        - Led to FBI raid
          - SJG won court case; founded EFF

<u>Data Encryption Standard (DES)</u>
- Most widely used encryption scheme

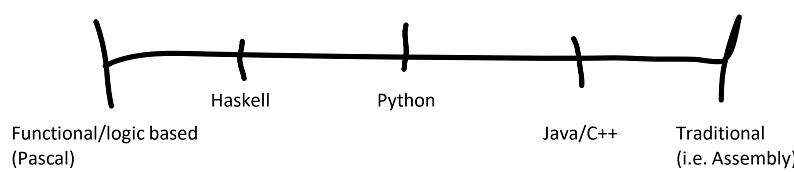
        - FIPS PUB 46
        - Referred to as Data Encryption Algo (DEA)
        - Limited to 56 bit key; 64 bit plaintext
        - Strength concerns
            - Concerns about algo
                - DES = most studied encryption algo in existence
                - Use of 56 bit key
- Not good

Triple DES
- Repeats basic DES algo 3x using 2-3 unique keys
- Attractions
    - 168-bit key length overcomes vulnerability to brute-force attack of DES
    - Underlying encryption algo same as in DES
- Drawbacks
    - Algo is sluggish
    - 64-bit

Advanced Encryption System (AES)
- Needed replacement for 3DES
- NIST called for proposals for new AES in 1997
    - Should have security strength >= than 3DES
    - Significantly improved efficiency
    - Symmetric block cipher
    - 128 bit data and 128/192/256 bit key size
        - vs 56 bit key size for DES and 112 (or 168) for 3DES
    - Scalable!
- New Standard

Practical Security Issues
- Typically symmetric encryption applied to unit of data > single 64 bit or 128 bit block
- Electronic codebook (ECB) mode = simplest approach to multiple-block encryption
    - Each block encrypted using same key
    - Cryptanalysis may be able to exploit regularities in plaintext
    - Added on to help deal w/ multiple blocks
- Modes of operation
    - Alternative techniques dev'd to increase security of symmetric block encryption for large sequences
    - Overcomes weakness of ECB

Pseudo-Random Numbers
- Why can't generate rand nums?
    - Computers = logical
- Side effect of some langs
    - Programming langs on a spectrum
    - Something not built into the language
- Require random block of text and starting place (seed)
    - How to generate 1st num? (More traditional langs)
        - Take system date& time, convert to int, use as offset
            - Assumed programs don't run @ exact same time
            - Can do this due to side effect
        - May be different for functional/logic based side



Block ciphers
- Processes input one block of elements at a time
- Produces output block for each input block
- Can reuse keys
- More common; strongest
- Well defined message; split into blocks
    - Perform operation on each block
- Not fast

Stream Cipher
- Processes input elements continuously
- Produces output one element @ a time
- Faster; good for real-time necessities
- Less secure; good for messages not important later in future

Message Authentication
- How do I know the sender is the person I think it is?
    - Message authentication = "digital seal"
- How do I deal w/ keys that might be compromised?

- Problem w/ sending confidential messages
    - Traditional mail/phone/etc. is unsecure
- Ensure recipient & sender are intended people

- Protects, but doesn't prevent interception
- Protects against active attacks
- Verifies received message is authentic
    - Contents have not been altered
    - From authentic source
    - Timely and in correct sequence
- Can use conventional encryption
    - Only sender & receiver share a key
- Encryption obsufacates message
    - Message authentication doesn’t; instead creates a "likeness" that only I know through hashing

<u>Hash Functions</u>

- Requirements
    - Can be applied to any size block of data
    - Produces fixed length output
    - H(x) is relatively easy to compute for any given x
    - 1-way or pre-image resistant
        - Computationally infeasible to find x such that H(x) = b
        - Very hard to reverse / find x from hash
    - Computationally infeasible to have multiples
    - Collision resistant or strong collision resistance
- Security
    - Ways to attack
        - Cryptanalysis
        - Brute force
    - SHA
- Not encryption b/c always get same result; vulnerability
    - Not complex enough

<u>Public Key Encryption Structure</u>

- Based on math functions
    - 2 keys being used together produce 1 unique key
        - 2 very large prime numbers; 1 public/1 private
- Asymmetric
    - Uses 2 separate keys
    - Public key and private key
    - Public key made public for others to use
- Some form of protocol is needed for distribution
- Plaintext
    - Readable message / data that is fed into algo as input
- Encryption algo
    - Performs transformations on plaintext
- Public & private key
    - Pair of keys, one for encryption, 1 for decryption
- Ciphertext
    - Scrambled message produced as output
- Decryption key
    - Produces original plaintext
- Requirements
    - Computationally easy to create key pairs
    - Easy for sender knowing public key to encrypt messages
    - Easy for receiver knowing private key to decrypt
    - Infeasible for opponent to determine private key from public key
    - Infeasible for opponent to otherwise recover original message
    - Useful if either key can be used for each role

<u>Digital Envelopes</u>

- Protects message w/o needing to 1st arrange for sender and receiver to have same secret key
- Equates to same thing as sealed envelope containing an unsigned letter
- Doesn't encrypt the message, but keeps it safe in a "container"

<u>Symmetric vs Asymmetric Encryption</u>

- Symmetric encryption requires use of shared key
    - Issue = how to send key to other person
        - Asymmetric encryption solves this w/ public/private key (2 keys instead of 1)

Symmetric Encryption & Message Confidentiality

Thursday, September 27, 2018 11:03 AM

Symmetric Encryption

- AKA
  - Conventional encryption
  - Secret-key / single key encryption
- Only alternative before public-key encryption in 1970s
  - Still most widely used alt
- 5 ingredients
  - Plaintext
    - What we want to encrypt
  - Encryption Algo
  - Secret Key
  - Ciphertext
  - Decryption algo
    - Usually reverse of encryption algo

Computationally Secure Encryption Schemes

- Encryption computationally secure if:
  - Cost of breaking cipher > value of info
  - Time req'd to break cipher > useful lifetime of info
- Usually very hard to estimate the amount of effort req'd to break
- Can estimate time/cost of a brute-force attack

- Feistel Network
  - Split crypto into 2 & xor it against key

Block Cipher Structure

- Symmetric block cipher consists of:
  - Sequence of rounds
  - W/ subs & permutations controlled by key
- Parameters & design features
- Most important = block size, followed by:
  - Key size
  - # of rounds
    - 1/2 # rounds = 1/2 strength
  - Subkey generation algo
    - As we process (ex: Feistel), start w/ 1 key, & in next round, increment key
  - Round function
  - Fast software encryption/decryption
  - Ease of analysis

- Data Encryption Standard (DES)
  - Most widely used
  - Triple DES (3DES)
    - Use 3 keys
    - Stronger
- Limiting factor = block size

AES

- Take plaintext
- Everything after 1st key (symmetric) gets "round key"
  - Perform lookup on table to see next step
- 1st round
  - Take entire message block & sub different bytes
  - Shift rows (transposition)
  - Shift cols
  - Lookup/check next round key
  - Repeat for next rounds
- Each starting key results in unique subsequent rounds

Mix Columns and Add Key

- Mix columns
  - Ops on each col individually
  - Mapping each byte to new value that's a function of all 4 bytes in col
- Add Key

Stream Cipher

- Processes input elements continuously -> Key input to a pseudorandom bit generator
- Produces stream of random like nums
  - 1st thing: generate key; key used to produce 1st round
- Unpredictable w/o knowing input key
- XOR keystream output w/ plaintext bytes
- Doesn't protect key since key isn't made until after connection
  - Someone listening in to traffic while making connection can get key

Block Cipher Modes

- CBC
  - Encrypts the XOR of next block of plaintext & previous block of ciphertext
- CFB
  - CBC & ECB
    - XOR against random
- OFB
  - CFB w/ DES
  - Very slow
- Counter (CTR)
  - For every block, come up w/ counter (random # based on key)
  - Increment key for every iteration of block (every block)
  - XOR against counter
  - Counter can be encrypted in message itself
  - After text encrypted, use counter to offset text
  - Agree on rand #, start w/ that #
    - As we process message, xor against counter
  - Can be included in crypto

Key Distribution

- Biggest problem w/ symmetric encryption = getting key to person securely
  - 4 ways
    - Physical meetup
      - Problem: have to meet up
    - Give to 3rd party who delivers it
      - Problem: How to trust 3rd party

        - Transmit key to recipient through encryption; generate new key encrypted w/ old key
            - Problem: Someone w/ old key knows new key
        - Encrypted message to 3rd party who delivers it
            - Problem: How to ensure secure communication between 3rd party & receiver
    - Answer = public/private key

Public-Key Cryptography & Message Authentication

Thursday, October 4, 2018 11:05 AM

Secure Hash Algorithm (SHA)

- SHA-1
  - 160 bit hash values
- SHA-256, 384, 512
  - 256/384/512 bit hash values
  - Same structure as SHA-1 but > security
- Can't undo hash w/o IV

HMAC

- Hashed Message Authentication Code
- General term
- Use hash to create digital signature
  - Make symbolic representation
  - Run message w/ hash to ensure message not tampered
  - Should get same symbolic representation if not tampered
- Basis of TLS
- Design Objectives
  - Use f(x) w/o modifications
  - Preserve original performance of hash f(x) w/o incurring degradation
  - Allow easy replaceability of embedded hash f(x)
  - Use & handle keys simply
  - Well Understood
- Security
  - Low possibility of collisions
  - Birthday Attack (finding collisions) = O(2n/2)
    - Brute Force = $O(2^n)$

RSA Public-Key Encryption

- Best known algo
- Multiply 2 primes for new #; hard to break
  - 1 prime = secret; 1 public
  - Anyone can know public key
- Key formulated from prime multiplication
- Encrypt: $C = M^e \bmod n$
  - n = prime
- Decrypt: $M = C^d \bmod n = (M^e)^d \bmod n = M$
- Sender & receiver know n & e
- Security
  - Brute force
    - Very slow
  - Math attack
    - Try all primes until you find the right key combo
  - Timing attack
    - If know how encryption systems work, know that system produces certain values @ certain times and use to find right key
  - Chosen ciphertext attacks
    - Take advantage of exploits

Diffie-Hellman Key Exchange

- Use 2 primes
  - Shared prime #s
    - Agree on #'s along w/ mod
  - Q
  - Alpha: < q & primitive root of 1
- Select private $X_A < q$
- Calculate public Y_A = alpha^{xa} mod q, & Y_B
- Exchange & compute secret key
- Downside: no built in signatures

Man in the Middle Attack

- Attacker generates 2 private keys & 2 public keys
- Attacker intercepts transmitted key w/ own and sends own/compromised key to receiver
- Parties think they're talking to each other; actually talking to attacker

Exam Review

Tuesday, October 9, 2018 11:03 AM

- Short answer & mult choice
  - 6-7 short answer; ~10 mult
- 1 pg notes
- RSA calculations not needed; know how works
- Priority ranking
- Know articles: Credit denial / spectre/meltdown
  - Meltdown = breakdown in partitions in memory
  - Spectre = able to write to other segments in memory using branch prediction
  - Know overview: how they work, why problem, potential fixes
- Know how to apply every topic
- Know Feistel networks
- CIA triad
  - Confidentiality
    - Secrecy
  - Integrity
    - How reliable
  - Availability
    - Ease of access
- History not needed
- Might need to do simple cryptography