More Crypto Problems

Software Security

Weak Random Numbers

Vocabulary

- Entropy: lack of order or predictability; gradual decline into disorder
  - How random something is

CWEs

- 330 – Use of Insufficiently Random Values
- 331 – Insufficient Entropy
- 334 – Small Space of Random Values
- 335 – PRNG Seed Error
- 338 – Use of Cryptographically Weak PRNG
- 340 – Predictability Problems
- 341 – Predictable From Observable State
- 342 – Predictable Exact Value from Previous Values
- 343 – Predictable Value Range from Previous Values

Keyword – Predictable / Predictability

- Imagine any website with a session ID that's just incremented number
  - Shawn – Regular User – Session 12
  - Dave – Super User – Session 13
  - Shawn session cookie++?

Focus Today:

- Non-Cryptographic Pseudo-Random Number Generators
- Cryptographic Pseudo-Random Number Generators
- True Random Number Generators

srand(time(0)); // nope*

- Historically, before we spent time thinking about computer security, random numbers on computers were used for:
  - Stats
  - …that's it

- Traditional random functions are not good for cryptography because it wasn't an original design consideration

- *modern systems might actually use the same PRNG for random and rand…

Ex1, Tx1, Tx2

- Generate 5 random numbers
- Determine the seed (with one thing done to save time*)
- Re-generate those same 5 numbers again

Ex2

- Better, but still not great
- Ref: https://wiki.sei.cmu.edu/confluence/display/c/MSC30-C.+Do+not+use+the+rand%28%29+function+for+generating+pseudorandom+numbers

Brute Force Isn’t the Only Issue

- Generally, generic PRNGs are predictable even without the seed

- Provided some sequence of numbers and a poor PRNG, you can reasonably guess the next number in sequence

Brute Force Isn’t the Only Issue

- We won’t attempt that today

Cryptographic PRNG

- Security of seed is still of utmost importance
- Brute force aside, a sequence of numbers shouldn’t help an attacker determine the next number in sequence

True Random Number Generators

- True random is hard – but there is random data a computer can survey
  - Keystrokes
  - Mouse movement
  - Screen content (if a user is present)

- Problems
  - attacker with access to same hardware and data
  - mouse movement and the like aren't truly random and can have bias

Linux Example (Not True Random)

- /dev/random
  - Generator gets noise from the system
  - If there's not enough noise, reads of /dev/random will block
    - Arch Linux's documentation states this could cause /dev/random to block indefinitely on a remote server
- /dev/urandom
  - "unlimited" random source
  - Reuses noise if there isn't enough on the system
  - Potentially less entropy than /dev/random
- Both special devices are writable by users

https://en.wikipedia.org/wiki//dev/random

Linux Example (Not True Random)

- Check the amount of entropy available with:
  `cat /proc/sys/kernel/random/entropy_avail`

https://en.wikipedia.org/wiki//dev/random

Why it all matters

- Predictability is a problem in cryptography

- We won’t dive into the math in this class

Discovery

- Use of non-random numbers or data where random should be used
  - Certain libraries don’t require a seed and will happily use the same one each time

Discovery

- Use of PRNGs
  - Most standard library implementations of random numbers are weak
  - Some are not
  - When in doubt, read the documentation

Discovery

- If CPRNGs are used, make sure good seeds are used
  - OS seeds are probably good
  - Seeds you choose are probably bad

Real World Examples

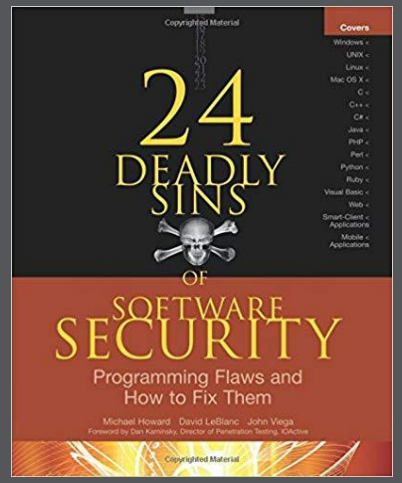
- CVE-2008-0166 – Debian Random Key Generation
  - Description here is great: https://github.com/g0tmi1k/debian-ssh
  - “All SSL and SSH keys generated on Debian-based systems (Ubuntu, Kubuntu, etc) between September 2006 and May 13th, 2008 may be affected.”
- Older: Netscape SSL
  - https://people.eecs.berkeley.edu/~daw/papers/ddj-netscape.html

Summary

- Use the system CPRNG
- If a secure CPRNG fails – don’t fallback to an insecure one
- Seed CPRNGs with a significant amount of random data
  - 64 bits of entropy suggested in 2010, consider more today
- Use hardware RNGs when required

References

- https://Wikipedia.org
- 24 Deadly Sins of Software Security
  - ISBN-13: 978-0071626750



Using the Wrong Crypto

CWEs

- 326 – Weak Encryption
- 327 – Use of a Broken or Risky Cryptographic Algorithm

Never Roll Your Own

- Even folks with degrees in cryptography make mistakes
  - Ref: AES call for proposals produced 15 proposals, 3 were broken before presentation

- Use peer-reviewed crypto

Using a Low-Level Protocol

- Most of the time, you don’t need to deal with the low-level crypto code
- Sending a message? Use TLS; don’t attempt key sharing and encryption directly yourself
- Don’t know how to use TLS with your code? Put a service in front that does*

Using Weak Crypto Primitives

- DES / 3DES
  - Key space is too small
  - Better alternatives available
  - Almost harder to use this than AES at this point
- MD4 / MD5 / SHA1
  - All have documented collisions
  - Can be used for less critical tasks
- RSA keys 1024 bits or less in length should have limited use
  - Random Ref: https://en.wikipedia.org/wiki/RSA_Factoring_Challenge

Incorrect Use of Crypto Primitives

- Using a stream cipher when a block cipher would be a better fit
- Hashing concatenated data
- Use of electronic code book mode with a block cipher
  - Ref: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#ECB
- Encryption of known plain text
- Incorrect hash verification

Ex3, Ex4

Using Wrong Communication Protocol

- Older versions of SSL/TLS should be avoided
- This is well documented
- Ref: https://www.stigviewer.com/stig/microsoft_internet_explorer_11/2017-12-12/finding/V-64729

Failing to Use a Good Salt

- Salt is random data, as discussed last week
- It can be used when deriving keys for cryptographic uses
- If it is not random or changed for each communication, intercepted traffic could have observable changes

- Same is true for picking a not random and/or small initialization vector
  - Ref WEP attacks: https://en.wikipedia.org/wiki/Initialization_vector#WEP_IV

Use of a Weak Key Derivation Function

- Generally, the password you enter to encrypt or decrypt something isn't actually used directly for the cryptographic operation
  - A derived key is calculated using your input into a slow function that results in a key of the proper size for your algorithm
- Fast KDFs result in poor keys
  - May be easier to attack password than key itself

Failing to Use an Integrity Check

- No matter the cipher, check if the message received is valid before attempting decryption – always

Failure to Use Agile Encryption

- Yes, I hate that word too
- Any crypto you use should be easily adapted

Discovery

- Search your source for weak libraries
  - MD4/MD5/SHA1
  - ECB
  - DES / 3DES
  - RC4
- Odd function call sequences
  - Decrypt just calls encrypt – ex5.py
- Any use of a custom crypto library is suspect

Discovery

- Any use of a lower level algorithm should be investigated, but may not be a problem

- Concatenation of data immediately before a hash is used for verification when the data should be validated separately

The Trend

- Code review and a thorough understanding of your application are necessary to be effective at finding these issues

- Tools can help, but they can’t fix everything

Remediation

- Replace the bad algorithms with modern algorithms
  - Use well tested libraries that include good KDFs
- Use existing higher-level libraries instead of primitives
- Use good random number generators
- Check the integrity of your data

Remediation

- Write code that can be easily updated to use different/new crypto

Do

- Use latest versions of TLS
- Use random salt
- Use random IV for chained block ciphers
- Use appropriate algorithms
  - AES for encryption
  - SHA-2 family for hashing

Do Not

- Roll your own
- Hash concatenated data
- Build your own protocol when a higher level will work fine
- Use SHA1
- Use DES / 3DES
- Use RC4
- Use ECB

References

- https://Wikipedia.org
- 24 Deadly Sins of Software Security
  - ISBN-13: 978-0071626750