Fantastic Crypto Bugs and Where to Find Them

Open Source Cryptography Workshop

Thai Duong

Mar 2023

Agenda

Fantastic Crypto Bugs

Bad Algorithms

- Weak PRNG
- Unauthenticated encryption
- [AES-]GCM
- Password-based encryption
- RSA PKCS v1.5 encryption
- ECDSA

Low level APIs

- OpenSSL/BoringSSL
- Java Cryptography Extension
- PyCrypto
- Golang Crypto

Common Mistakes

- Unauthenticated public keys
- Horton principle violations
- Length extension attacks
- Padding oracle attacks
- Invalid curve attacks
- Non-constant time comparisons

Weak PRNG

- We need unpredictable randomness to do crypto
- Unfortunately, most generators are predictable
    - glibc: **rand()/srand()**
    - Java: java.lang.**Math.random()**
    - Javascript: **Math.random()**
    - Golang: **math/rand**
    - Python: **random.random()**
- You want to find PRNGs that take a **seed** from users. Most of the time it'd be seeded with, well, time

Unauthenticated Encryption

- Block ciphers can only encrypt a fixed size of data. To encrypt more, one needs a block cipher mode of operation
- Unfortunately, most modes are insecure: **ECB, CBC, CTR, CFB,** etc. Depending on the usage, you can easily recover or modify the plaintext
- The most common stream cipher **RC4** also allows you to modify, and, in certain cases (WEP), recover the plaintext

AES-GCM

- This is the most popular authenticated encryption mode, recommended by NIST
- It has many issues though
  - It requires a nonce. If the nonce is reused, it becomes unauthenticated
  - It is NOT key committing, e.g., it’s possible to find AES_GCM(k1, msg1) == AES_GCM(k2, msg2)

Password-based Encryption (RFC 2898)

- Most user passwords have less than 40 bits of entropy, this means **PBE** only provides 40-bit security level

RSA PKCS v1.5 Encryption

- This is an old standard. Judging from the number of attacks found each year, it remains super popular
    - https://eprint.iacr.org/search?q=bleichenbacher
- Many implementations leak side channel information (e.g., timing, errors, etc.) that allows plaintext recovery

ECDSA

- ECDSA requires a **nonce.** If a single nonce is reused to sign two messages, you can recover the private key
- If the nonces of a handful messages are biased or partially leaked, you can also recover the private key
- Recently, someone found that Java accepted (0, 0) as a valid signature for all messages
    - This is due to ECDSA requiring finite field arithmetic which is hard to implement correctly

Where to Find [Fantastic Crypto Bugs]

1. Choose a keyword

- nonce salt key IV password
- MD5 AES RC4 RSA ECDSA
- CBC CRT CFB ECB GCM
- Math.random()
- random.random()
- math/rand
- Cipher.getInstance()
- BadPaddingException

2. Search GitHub

- Your favourite open source projects
- All of GitHub if you are feeling lucky

3. Determine exploitability

- Read and play with the code
- File bugs!

Let’s go find some bugs!

Advanced Techniques

CodeQL

CodeQL helps automate code analysis. It lets you query code as though it were data. You can write a query to find all variants of a crypto vulnerability, and share your query to help others do the same.

CryptoFuzz

Cryptofuzz, well, fuzzes cryptographic libraries and compares their output in order to find implementation discrepancies. It's quite effective and has already found a lot of bugs.

Project Wycheproof

Project Wycheproof tests crypto libraries against known attacks. It provides tons of ready to use of test vectors that have helped found a lot of bugs.