Cryptanalysis Tutorial & Challenge
Secret Sharing or Secret Spoiling

Eric Filiol
efiliol@hse.ru, efll@protonmail.com
https://ericfiliol.site

Thales Digital Factory, France & HSE, Moscow, Russia

November 11^{th}, 2022

Cryptographic Challenge

- This challenge is inspired by a real case I have met in a recent past.
- It may be a very interesting example of what backdoors could/might be among many other possibilities.
- The challenge target is a GMP (GNU Multiple Precision Arithmetic Library) implementation of the Shamir Secret Sharing Scheme.
- GMP library, developed by Torbjörn Granlund has become a widely used standard for the implementation of large numbers (integers, rationals, floats) (see `gmplib.org` for documentation)
  - Similar libraries have been added for IEEE 754 float numbers (`mpfr.org`) or for complex numbers, univariate polynomials over reals ... (see `multiprecision.org` for similar libraries)

.

Inital Step (for Reverse Engineers)

- The Linux binaries have been provided yesterday.
  - The aim was to play the reverse engineering part for anyone interested and daring.
  - Expected result: corresponding source code (assembly or C code).
- Is there any answer from today's audience?

How to get the Source Code

- A C code is provided to you.
  - The file is *BSidesLisbon2022_challenge.tgz* and has a size of 107 689 bytes
  - It contains four files `gost_grasshopper.h`, `rsa_lib_gmp.h`, `sssp.h`, `sssp.c`.
  - To compile it, use `gcc -ansi -O3 sssp.c -o sssp -lgmp`. GMP library must be installed before.
- This implementation is not optimized (for readability purposes) but it is rather close to industrial implementation encountered so far.
- The file is available from now on at the following link `http://e.pc.cd/3ilotalK`. The password is `BSidesLisbon2022*!`.
- On a scale of 1 to 10, the difficulty of this challenge is 5.

Timeline of the Challenge

- From now on, you have to
  - Analyse the source code and find the backdoor.
  - Find how to exploit the backdoor. A short description of the cryptanalysis principle with complexity result is to provide.
- A prize is awarded to the first of you who identifies the nature of the backdoor (a flipper zero device)
- A prize is awarded to the one who explains how to exploit this backdoor with the lowest complexity (a NEO Smart pen N_2)
- Send your answer to `efll@protonmail.com`. My Telegram ID is @Efll62 The internet and protonmail clock will be used as time proof.

Source Code Analysis

- From now on
  - Part I: let us analyze the protocol and explain the code in details. Feel free to ask any questions (expected time 30')
  - Right after, you get 20' to identify the flaw. Questions are still welcome.

Backdoor Identification

- The key point is to generate a message key K which is RSA-encrypted and can be decrypted/accessed if at least $t = 3$ shares are provided thanks to the Shamir Secret Sharing (SSS) protocol. Then we have

```
1                /** Generation of message key K **/
2                sleep(1);
3                gmp_randseed_ui(Random_State, seed);
4                mpz_rrandomb(K, Random_State, 256);
5
```

- So to access any file encrypted with K you have seemingly either to break the RSA encryption or the SSS protocol.
- In fact, the backdoor consists in using the `mpz_rrandomb` function instead of the `mpz_urandomb`.

Random Generation with GMP

`void mpz_rrandomb (mpz_t rop, gmp_randstate_t state, mp_bitcnt_t n)`
Generate a random integer with long strings of zeros and ones in the binary representation. Useful for testing functions and algorithms, since this kind of random numbers have proven to be more likely to trigger corner-case bugs. The random number will be in the range 2^{n-1} to $2^n - 1$, inclusive.

`void mpz_urandomb (mpz_t rop, gmp_randstate_t state, mp_bitcnt_t n)`
Generate a uniformly distributed random integer in the range 0 to $2^n - 1$, inclusive.

Hint for Cryptanalysis

- Now you have to guess and find how to exploit the flaw and solve the challenge.
- The most convincing method will awarded. If more than one solution is proposed, the computational complexity will be used as a tiebreaker.

Backdoor Characterization

- So the randomness quality of `mpz_rrandomb` is awful and produce very structured secret keys. For instance, we have

  $$K = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00000000FFFFFFE0000000000000$$

  with 4 runs of length $53, 27, 32, 144$.
- The best cryptanalytic approach consists in attacking secret keys directly since only very few possible keys are possible. For instance (statistical simulation over $N = 1,000,000$ keys)
  - Around 12.7 % of the key have 3 runs (of zeroes or ones) exactly.
  - Around 63 % of the keys have at most 6 runs.

Backdoor Exploitation

- The principle is very easy. For each possible number of runs, we exhaustively generate and try keys. For a n-run key, we have $2^{8.(n-1)+1}$ different keys to test at most. For instance to generate any 3-run key (around 12.6 % of the keys), here is the pseudo code.

Algorithm 1 Generation of 3-run Keys

```
1: for i from 1 to 254 do                                          ▷ First run
2:     for j from 1 to 256 - i - 1 do                              ▷ Second run
3:         K ← 0-run of length i||1-run of length j||0-run of length (256 − i − j)
4:         K' ←⤻ K                                                 ▷ invert 0-runs and 1-runs
5:         Try K and K'
6:     end for
7: end for
```

- To sum up, to break 63 % of the keys, the complexity attack is $\mathcal{O}(2^{41})$. To break all keys, the complexity is $\mathcal{O}(2^{137})$ (exhaustive search is in $\mathcal{O}(2^{256})$).

Conclusion

Thank you for your attention

Questions & Answers