

Hands-on: Cache Side Channel on GnuPG

CS665: Secure Memory Systems

2019-20 - Semester I

Computer Science and Engineering Department

Indian Institute of Technology Kanpur

August 10, 2019

Instruction to build GnuPG

1. Download GnuPG from <https://gnupg.org/ftp/gcrypt/gnupg/gnupg-1.4.13.tar.gz>
2. Extract it
3. `cd path/to/gnupg`
4. Install **lib32-glibc** and **gcc-multilib** (require to build GnuPG for 32bit Architecture)
5. Configure Build system for 32 bit architecture with debugging symbol as follows:
`./configure --build=i686-pc-linux-gnu "CFLAGS=-m32 -g" "CXXFLAGS=-m32 -g" "LDFLAGS=-m32 -g"`
6. `make`

The binary will show up in `path/to/gnupg/g10/gpg`

Creating a Victim Private Key

1. Create a directory (let say it **testconf**)
`mkdir path/to/gnupg/testconf`
2. Change permission to **700** for **testconf**
`chmod 700 path/to/gnupg/testconf`
3. Set an environment variable **GNUPGHOME** with the **testconf** directory
`export GNUPGHOME=path/to/gnupg/testconf`
4. Generate RSA key pair of 2048 bit
`path/to/gnupg/g10/gpg-gen-key`

Select:

RSA and RSA
2048 bit
Never Expires
Name for key: let say it **TestKey**

Encrypt and Decrypt a Message

1. Create a Directory (let say it **testdir**)
mkdir testdir
2. Create a message file (let say **hello.txt** in **testdir** directory)
echo "Hello world" > path/to/testdir/hello.txt
3. Encrypt the message file using gnupg.
path/to/gnupg/g10/gpg -r "TestKey" -e path/to/testdir/hello.txt
4. Decrypt the message file using gnupg.
path/to/gnupg/g10/gpg -d path/to/testfiles/hello.txt.gpg

Functions of interest for Cache Side Channel attack

1. **Square (S)** function located in **mpih-mul.c** file at line **270** (function **mpih_sqr_n()**)
2. **Module (r)** function located in **mpih-div.c** file at line **329** (Loop in default case in **mpihelp_divrem()**)
3. **Multiply (M)** function located in **mpih-mul.c** file at line **121** (**mul_n()**)

To find the virtual address for Functions of Interest we can use **objdump** or **GDB**

Using objdump

Use **objdump** to get the object dump of **gpg** binary as follows:

```
objdump -D -M intel path/to/gnupg/g10/gpg | less
```

After getting object dump of **gpg** search for desired Functions.

Using GDB

Run your **gpg** with **gdb** and place **break points** on desired function.

```
gdb path/to/gnupg/g10/gpg  
br mpih_sqr_n  
br mpihelp_divrem  
br mul_n  
run -d path/to/testdir/hello.txt.gpg
```