# Flush+Reload Attack on RSA (square and multiple algorithm).

Flush+Reload is Cache Side-Channel Attack that was originally proposed by Yuval Yarom and Katrina Falkner of the The University of Adelaide in their paper entitled, Flush+Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack at 23rd USENIX Security Symposium in 2014. This repository contains the code for Flush+Reload attack on Square and Multiple algorithm of RSA as explained by Yuval Yarom in his research paper. We have reproduced this attack in our own settings/system configuration for experiments. Implementation details are provided in the following.

## RSA & Square and Multiple Algorithm:

Rivest–Shamir–Adleman (RSA) is fist and widely used public-private encryption algorithm used for security of information during transmission. Public key is used for encryption of information where a Private key is used to decrypt the ciphers into information, this asymmetric behavior made RSA one of most used algorithm on internet. Square and multiple algorithm is implementation algorithm of RSA on softwares and used in most of available libraries including GnuPG, axTLS, etc.

## AXTLS Library:

We have used axTLS-1.5.4 server application and configure it to use Square and Multiple algorithm for RSA encryption during serving of HTTPS requests. Following commands are used to compile library.

download axTLS-1.5.4 from here

https://sourceforge.net/projects/axtls/files/1.5.4/axTLS-1.5.4.tar.gz/download

./make

and once it installed succesfully.

make menuconfig  
   
 -> BigInt Options  
 -> Untick all algorithms except "Square Algorithm"  
 -> save changes

## Threshold Calculation:

This implantation contains a tool to generate memory access timings for READ operation from both cache and main memory. Threshold is taken as average of cache and main memory access time for this implementation, Threshold varies for different machines. This tool requires gnuplot and its dependencies must be installed on machine before usage. Threshold on our specified machine is 180-cycles.

## Attack:

This attack is single round attack which able to extract full key in single encryptions. After compilations, attack can be run as: This attack requires axTLS server running in different process to work, server can executed using following commands:

copy the folder “sampleKeyForAxtls” from this folder, then go into the \_stage directory then paste it into this directory here open a terminal into this folder, now run these commands:

export LD\_LIBRARY\_PATH= PATH\_TO\_AXTLS\_SERVER/axtls-code/\_stage/  
   
 //below code to run server  
  
 ./axhttpd -p 127.0.0.1:8081 -s 127.0.0.1:8443 -key axtls-root/ssl/server.key -cert axtls-root/ssl/server.crt -w axtls-root/www

once server is running succesfully, attack can be run as:

./spy 127.0.0.1 8443

or attack can be run in loop (20 times) on terminal of ubuntu using run script. This attack have limitations as described in paper by Yarom, thus running in loop will increase chance of getting full key.

./run.sh

## Configurations:

This attack requires some machine specific details, you need to edit ?config? file and set it according to your machine, such as:

AXTLSLIB: Contains path to axTLS Library "libaxtls.so.1" for  
  
LIBCRYPTO\_SQUARE\_OFFSET: offset to instruction which execute many time during single function (bi\_square) call,. e.g., loop starting point  
LIBCRYPTO\_MULTIPLY\_OFFSET: offset to instruction which execute many time during single function (bi\_terminate) call,. e.g., loop starting point  
LIBCRYPTO\_BARRETT\_OFFSET: offset to instruction which execute many time during single function (bi\_subtract) call,. e.g., loop starting point  
  
THRESHOLD: Your machine threshold.  
  
SQUARE\_HIT: No. of traces captured in single square function call to consider as 1 call,. e.g,. 10  
MUTLIPLE\_HIT: No. of traces captured in single multiple function call to consider as 1 call,. e.g,. 10  
  
ORIGINALKEY: Key to compare with retrieved key,. XOR results.

offsets for bi\_square, bi\_terminate and bi\_subtract can be find by disassembling library using following commands:

objdump -d libaxtls.so.1 > analysis