# Prime and Probe Cache Side Channel Attack

This is an implementation of a prime-probe side channel attack.

It attacks OpenSSL’s AES-128’s t-table implementation. In order to perform this attack on your own machine, follow the steps outlined below.

## OpenSSL Installation

Trusted Versions of OpenSSL can be found at: https://www.openssl.org/source/old/. This attack will work for most versions, but the specific version I used was version openssl-1.1.0f. After downloading the OpenSSL source, go to the Downloads folder and unzip with:

tar -xvf openssl-1.1.0f.tar.gz

Now we need to configure OpenSSL to use its t-table c implementation as opposed to the assembly implementation default. OpenSSL also needs to be configured with debug symbols and specified to use a shared object as opposed to an .a library. For the appropriate configuration, run:

cd ~/Downloads/openssl-1.1.0f  
./config -d shared no-asm no-hw

For the selected version: 1.1.0f, this configuration will install OpenSSL in the /usr/local/ directory. The configuration parameters specify that we allow for debug symbols (used to locate T-table locations), create a shared object, only use c implementations of aes (to use the t-tables), and to not use any hardware routines. To proceed with the install, run:

sudo make  
sudo make install\_sw

## Finding Cache Hit/Miss Threshold

Our folder also contains a calibration tool to automatically find the threshold for a cache miss / cache hit. The threshold finding method taken from https://github.com/IAIK/flush\_flush.

Simply compile and run the tool with:

gcc calibration.c -o calibration

This should output an appropriate benchmark for each individual machine. Edit the **main.c**, file’s **MIN\_CACHE\_MISS\_CYCLES** constant to the number that was output.

## Finding T-table Addresses

we must find the offset of addresses of the t-tables, with respect to the **libcrypto.so** shared object. To find this, perform the following commands:

cd /usr/lib  
readelf -a libcrypto.so > ~/aeslib.txt

This will deconstruct the **libcrypto.so** file and allow us to find the appropriate address offsets. We will use vim to find the output quickly:

vi ~/aeslib.txt

Search for the t-tables by pressing ‘/’ and typing ‘Te0’. Take note of these offsets, and change the **probe** character array in **main.c** to the appropriate offsets for your specific machine.

## Compile and run the program

Since we have installed OpenSSL in a local directory instead of a system directory, we need to tell the linker to use the appropriate version of OpenSSL. To do this, type in terminal:

export LD\_LIBRARY\_PATH=/usr/local/lib

The command for compiling the main.c file is:

gcc main.c -o main -I/usr/local/include/ssl -L/usr/local/lib -lcrypto

The output should be equal to the key specified in the main.c file.

To run it DO this: ``` sudo taskset -c 4 ./a.sh