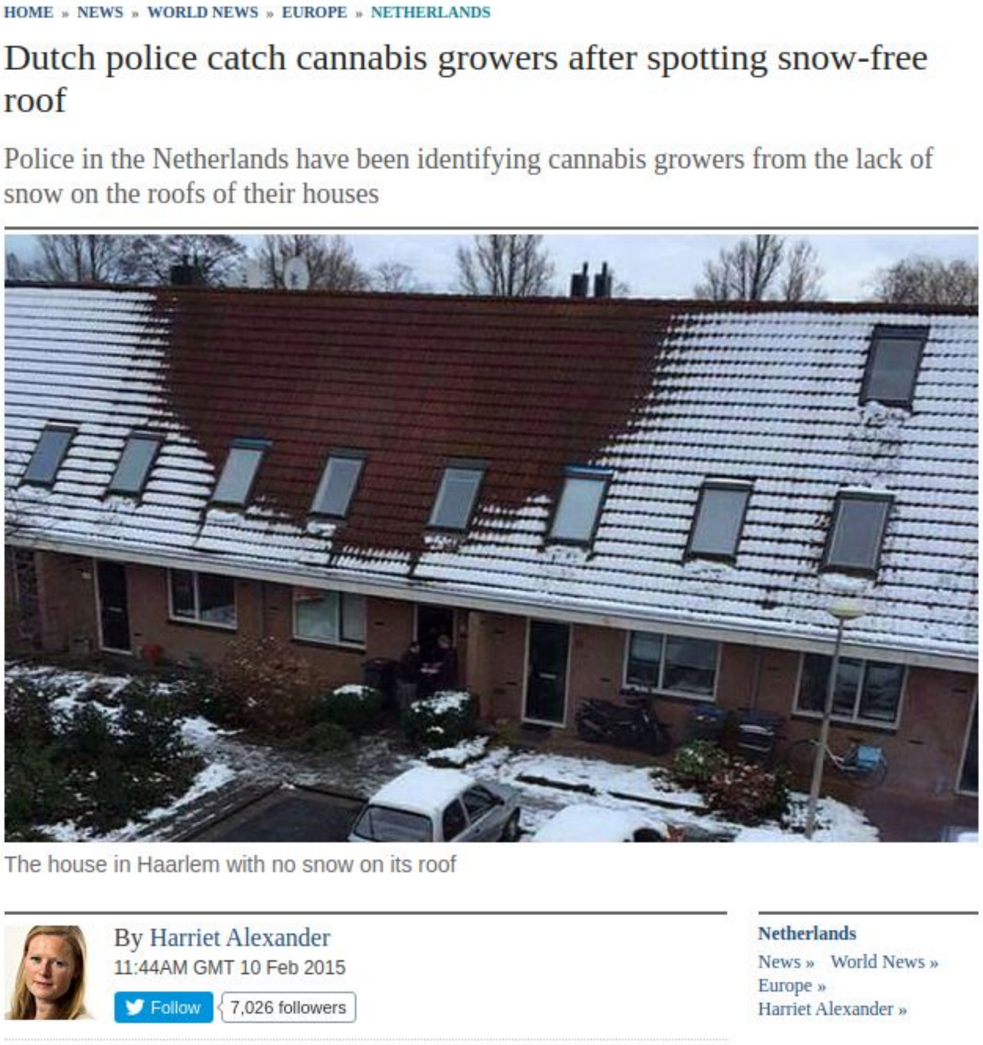
Assignment 2: Flush + Reload

Hardware Security

What



HOME » NEWS » WORLD NEWS » EUROPE » NETHERLANDS

Dutch police catch cannabis growers after spotting snow-free roof

Police in the Netherlands have been identifying cannabis growers from the lack of snow on the roofs of their houses

The house in Haarlem with no snow on its roof

By Harriet Alexander

11:44AM GMT 10 Feb 2015

Follow 7,026 followers

Netherlands

News » World News »
Europe »
Harriet Alexander »

What

usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack

Yuval Yarom and Katrina Falkner, *The University of Adelaide*

https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/yarom

This paper is included in the Proceedings of the 23rd USENIX Security Symposium.

August 20–22, 2014 • San Diego, CA

ISBN 978-1-931971-15-7

Open access to the Proceedings of the 23rd USENIX Security Symposium is sponsored by USENIX

FLUSH+RELOAD: a High Resolution, Low Noise, L3 Cache Side-Channel Attack

Yuval Yarom Katrina Falkner
The University of Adelaide

pages between non-trusting processes
hod of reducing the memory footprint
ystems. In this paper we demon-
eakness in the Intel X86 processors,
processes to information leaks. We
AD, a cache side-channel attack
his weakness to monitor access
pages. Unlike previous cache
H+RELOAD targets the Last-
essors with three cache lev-
program and the victim do
core.
f the FLUSH+RELOAD
rivate encryption keys
PG 1.4.13. We tested
d processes in a sin-
cesses running in
the attack is able
key by observ-

system
n pro-
based
d li-
be
n-

from the shared use of the processor cache. When a process accesses a shared page in memory, the contents of the accessed memory location is cached. Gullasch et al. [29] describes a side channel attack technique that utilises this cache behaviour to extract information on access to shared memory pages. The technique uses the processor's `clflush` instruction to evict the monitored memory locations from the cache, and then tests whether the data in these locations is back in the cache after allowing the victim program to execute a small number of instructions.

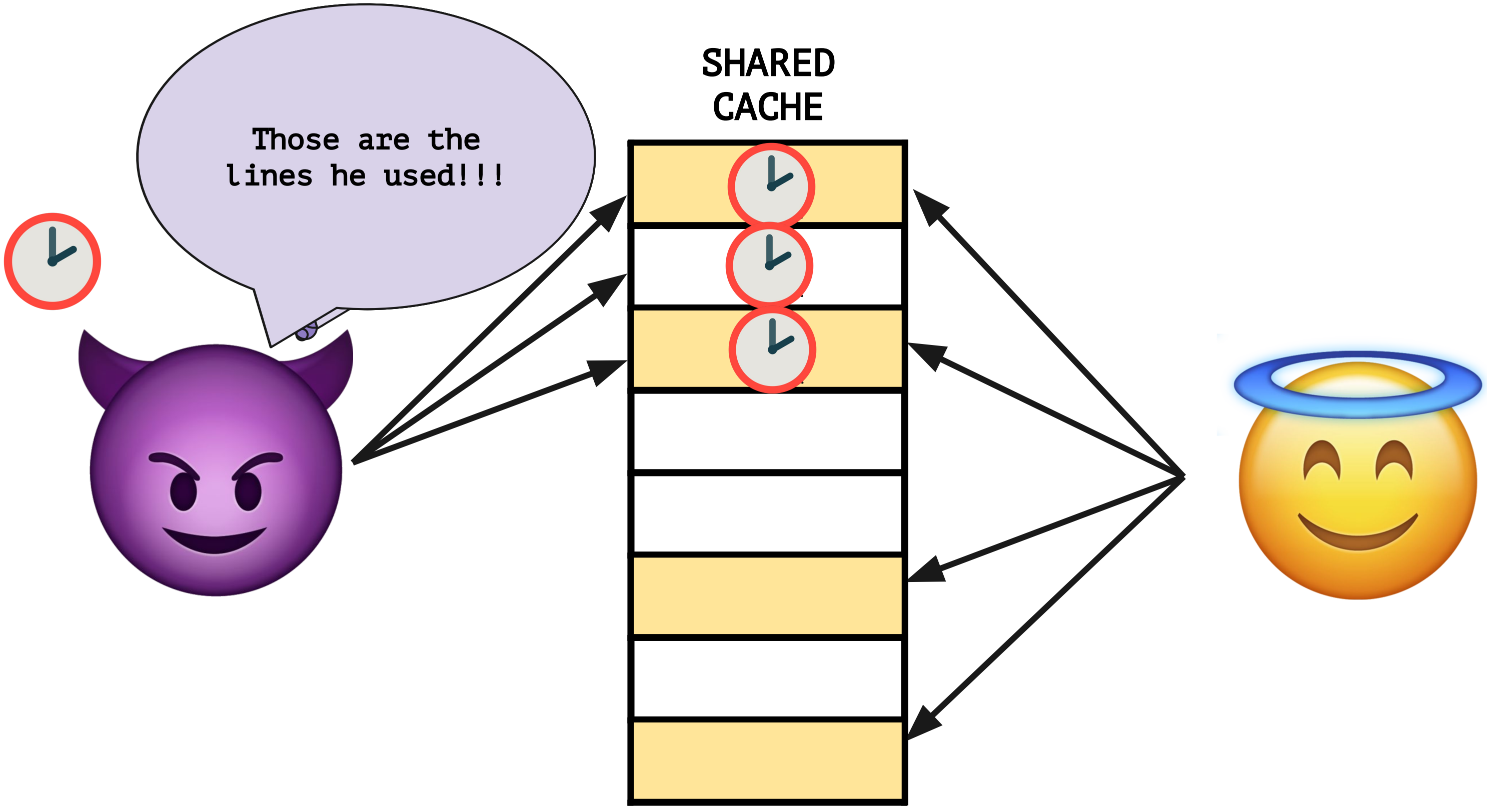
We observe that the `clflush` instruction evicts the memory line from all the cache levels, including from the shared Last-Level-Cache (LLC). Based on this observation we design the FLUSH+RELOAD attack—an extension of the Gullasch et al. attack. Unlike the original attack, FLUSH+RELOAD is a cross-core attack, allowing the spy and the victim to execute in parallel on different execution cores. FLUSH+RELOAD further extends the Gullasch et al. attack by adapting it to a virtualised environment, allowing cross-VM attacks.

Two properties of the FLUSH+RELOAD attack make it more powerful, and hence more dangerous, than prior micro-architectural side-channel attacks. The first is that the attack identifies access to specific memory lines, whereas most prior attacks identify access to larger classes of locations, such as specific cache sets. Consequently, FLUSH+RELOAD has a high fidelity, does not suffer from false positives and does not require additional processing for detecting access. While the Gullasch et al. attack also identifies access to specific memory lines, the attack frequently interrupts the victim process and as a result also suffers from false positives.

The second advantage of the FLUSH+RELOAD attack is that it focuses on the LLC, which is the cache level furthest from the processors cores (i.e., L2 in processors with two cache levels and L3 in processors with three). The LLC is shared by multiple cores on the processor die. While some prior attacks do use th

[F+R paper](#)

What



Once upon a time

*“We need stronger crypto algorithms to protect our communications” **

Herbert J. Bos, 12/09/2001

*very unknown but truthful quote

Once upon a time

T-tables are fast!!
I’ll implement my own crypto!!

T-table crypto (kinda)

```
uint64_t table[256] = (0x6ef72dc68d9d5af9, ... ,0x32676ab64008ac79
                       0x262981b7a097ac2c, ...  ,0x0ac84dbcb82c748a
                                 ...                 ...
                       0x01a892a87c2acc27, ...  ,0x62377c01b1db094c
                       0x12426129dd7123a8, ...  ,0x537ce5189a75db1f)
```

```
hjb_encrypt(char in[8], char key[8], char out[8]){
   for (int i = 0; i < 8; i++) {
     out[i] = do_something(table[in[i] ⊕ key[i]]);
   }
}
```

Assignment

0x3 Stages:

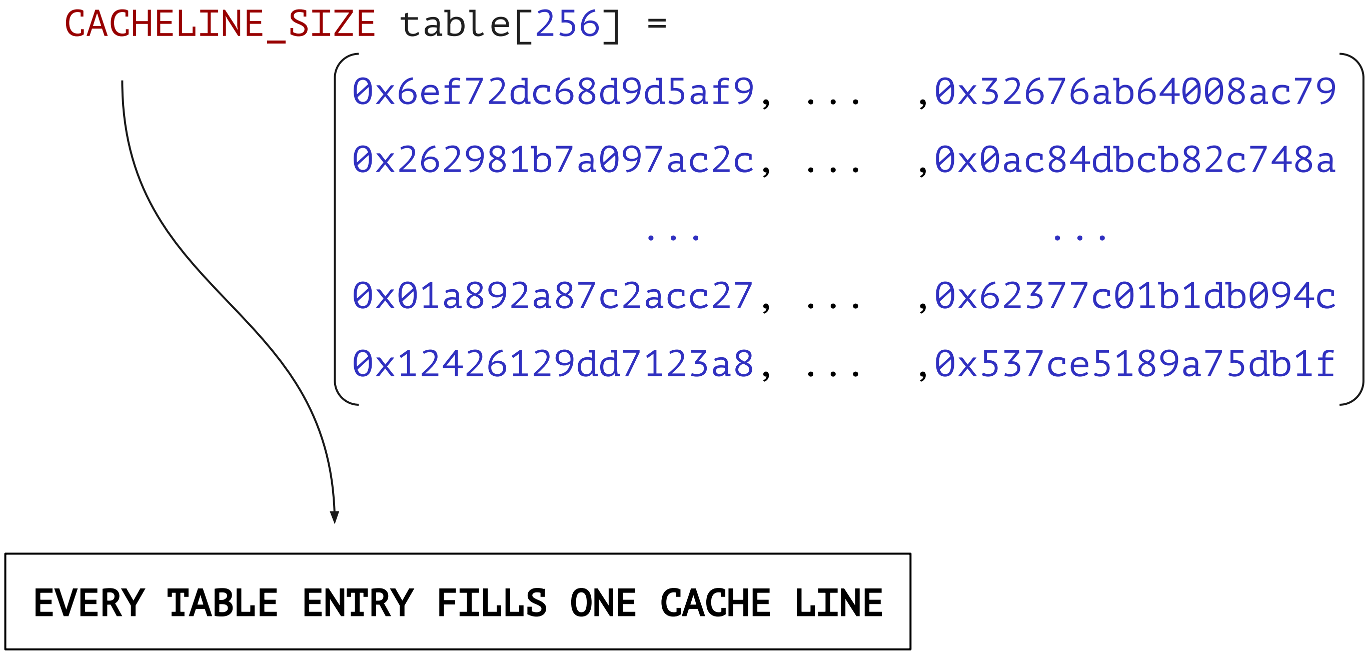
- Easy crypto
- Evict + Reload
- Hardened crypto

Assignment

Stage #1
Easy Crypto

T-table crypto (kinda)

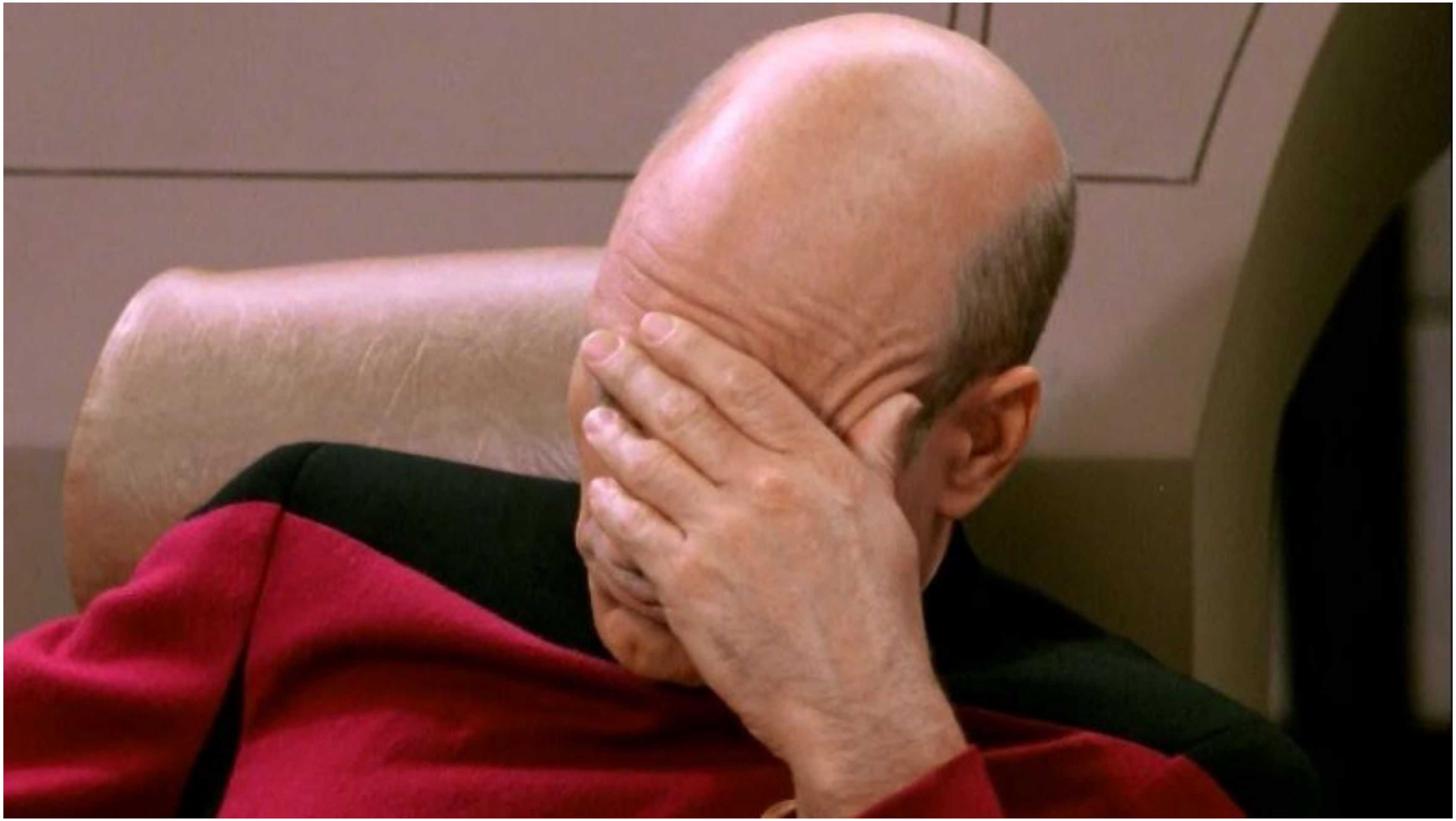
… Herbert didn’t really understand CPU caches



```
CACHELINE_SIZE table[256] =
          ⎛0x6ef72dc68d9d5af9, ...  ,0x32676ab64008ac79⎞
          ⎜0x262981b7a097ac2c, ...  ,0x0ac84dbcb82c748a⎟
          ⎜         ...                  ...           ⎟
          ⎜0x01a892a87c2acc27, ...  ,0x62377c01b1db094c⎟
          ⎝0x12426129dd7123a8, ...  ,0x537ce5189a75db1f⎠
```

EVERY TABLE ENTRY FILLS ONE CACHE LINE

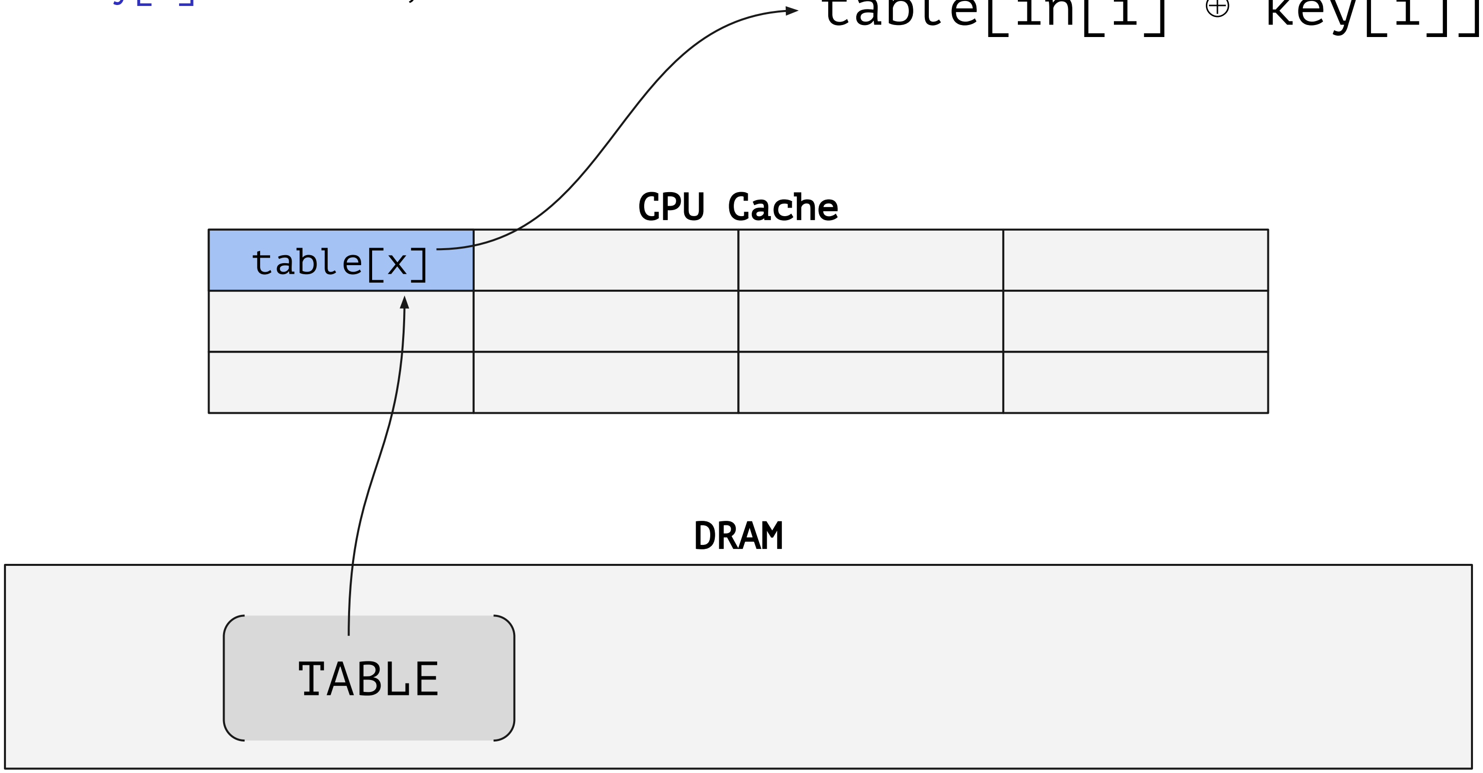
T-table crypto (kinda)



FLUSH + RELOAD

```
char in[8] = controlled_data;
char key[8] = secret;
```

```
for (i in 0..7)
    table[in[i] ⊕ key[i]]
```



CPU Cache

table[x]

DRAM

TABLE

FLUSH + RELOAD

- You have two options:
    - **v.1** Flush + Reload all table
    - **v.2** Flush + Reload table[const]

FLUSH + RELOAD v.1

Flush + Reload all table

```
// remove the table from the cache
for (i in 0..255)
      clflush(table[i])

// bring back table[secret] to the cache
hjb_encrypt(...)

// time every access
for (i in 0..255)
      time(table[i])
```

Tells you which table entry was loaded by hjb_encrypt()

v.1 Challenges

- **Prefetcher:** Optimizes memory accesses to hide latency. (You will see cache hits where you don't have them)

- **Unknown key bytes order:** You will recover the 8 bytes belonging to the key but you don't know the order.
  You can bruteforce this (if you defeat the prefetcher 👿)

FLUSH + RELOAD v.2

Flush + Reload table[const]

```
for (i in 0..7): //for each position of in[8]
    reset(results) //A 256B buffer for the values hit counts
    for (val in 0..256): //try all the possible byte values
        in[i] = val
        for (round in 0..10K) {
            in[other_bits] = rand()%256
            clflush(table[0])
            hjb_encrypt(in, ...)
            if (time(table[0]) < CACHED_THRESHOLD)
                results[i]++
        }
        check_probability(results)
```

FLUSH + RELOAD v.2

Flush + Reload table[const]

Assignment

Stage #2

Evict + Reload

Stage #2

Since HJBCrypto is too easy, we make it difficult for you and we remove **clflush**.

You need to force the table entries out of the caches by means of eviction.

EVICT + RELOAD

Simply fill up the caches

Assignment

Stage #3

Hardened Crypto

T-table crypto v.2

Herbert realised his idea of CACHELINE_SIZE entries was not smart 🤷‍♀️

He introduced 2 fixes:

- Two tables (for two rounds)
- uint64_t table entries (Multiple entries per cacheline)

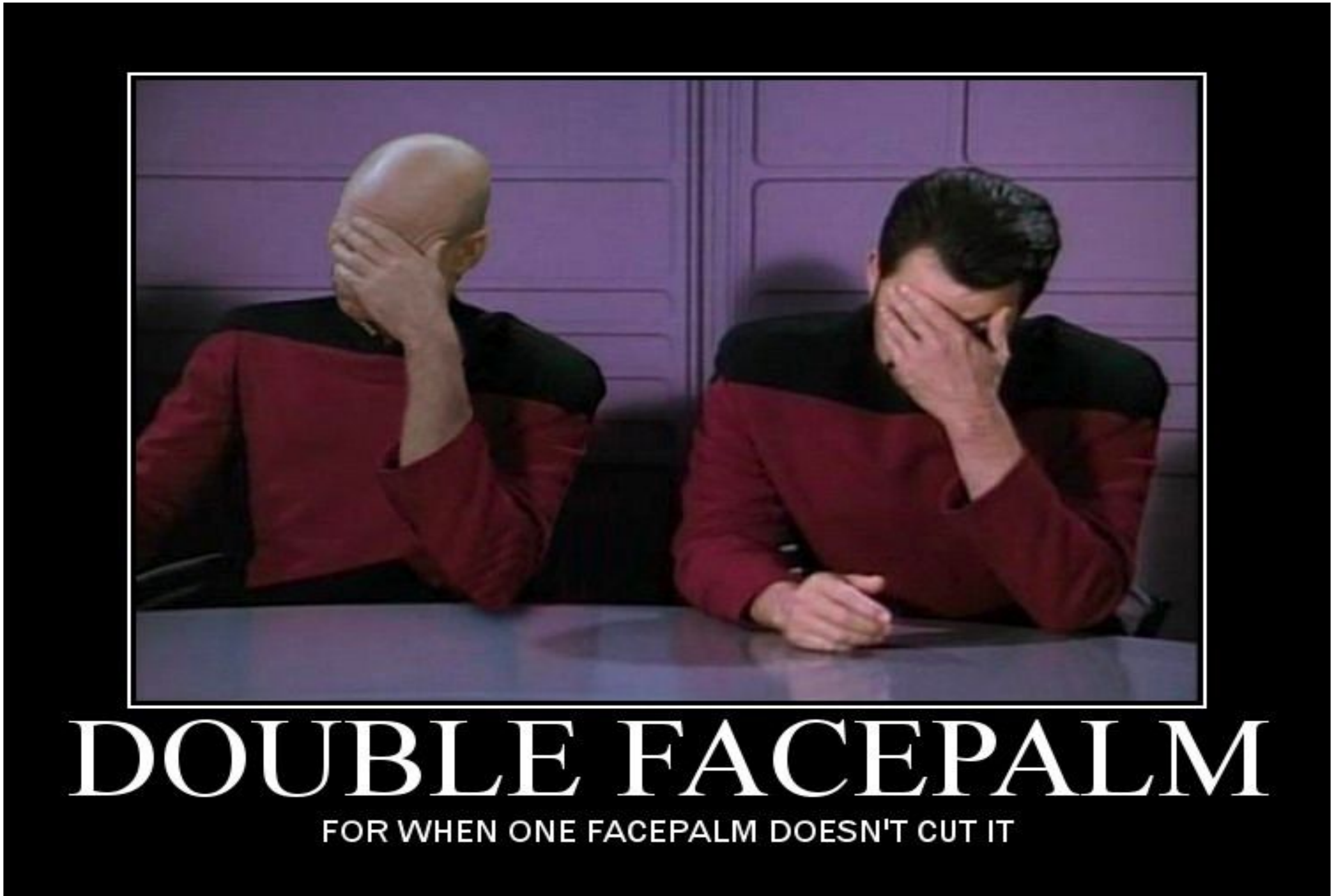
T-table crypto v.2

Unfortunately he didn’t really think this through ...

```
for (i in 0..7) {
    // first round
    val = Te0[(key[i] & 0xf0) ⊕ in[i]];
    ...
    // second round
    val = Te1[(key[i] << 4) ⊕ in[i]];
    ...
}
```

You can leak the key in pieces of 4 bits

T-table crypto v.2



HJB’s Long tales

*“Coming from a systems background,
he drifted into security a few years ago
and never left. Even so, he still does not
understand crypto, and hides this by saying
that he prefers to stay on the systems'
side of security.” **

Herbert J. Bos, Blackhat 2016

*only truthful quote of this slides deck

Deliverable

./attack

Perform a **FLUSH + RELOAD** attack by default.

-DEVICT enables **EVICT + RELOAD**.

-DHARDENED enables the **Hardened version**.

NOTE: Check your code on our cluster before submitting, this is where we will grade it.

Tip: You can set the secret using -DSECRET=0x6162636465666768 to try different keys.

Grading & Deadline

- Deadline:
  - Deadline **Tuesday Nov 5, 23:59**
    Delays: -1pt per late day
- Grading:

  7 ⇒ Stage #1 Flush + Reload

  9 ⇒ Stage #2 Evict + Reload

  11 ⇒ Stage #3 Hardened Crypto

Questions?

- Discussion board on Canvas
    - Help each other
    - Don’t give away your solution