Two methods for exploiting speculative control flow hijacks

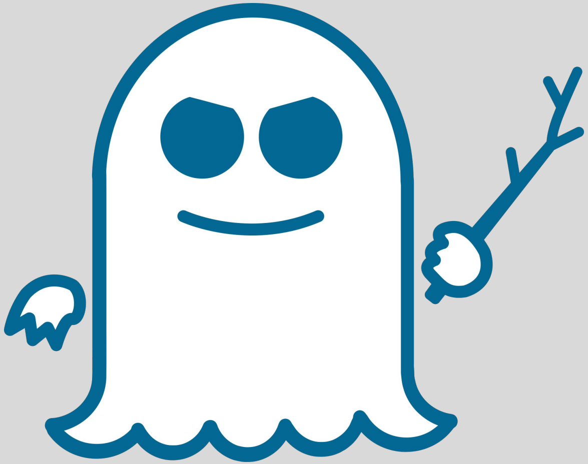
Andrea Mambretti

Alexandra Sandulescu, Matthias Neugschwandtner, Alessandro Sorniotti, Anil Kurmus

Northeastern University

IBM Research

Early 2018

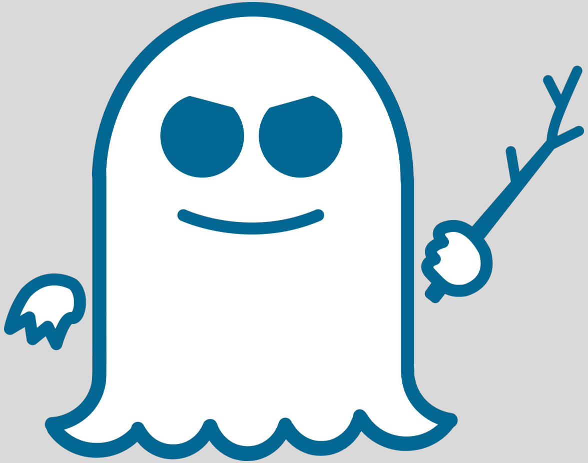


intel

ARM

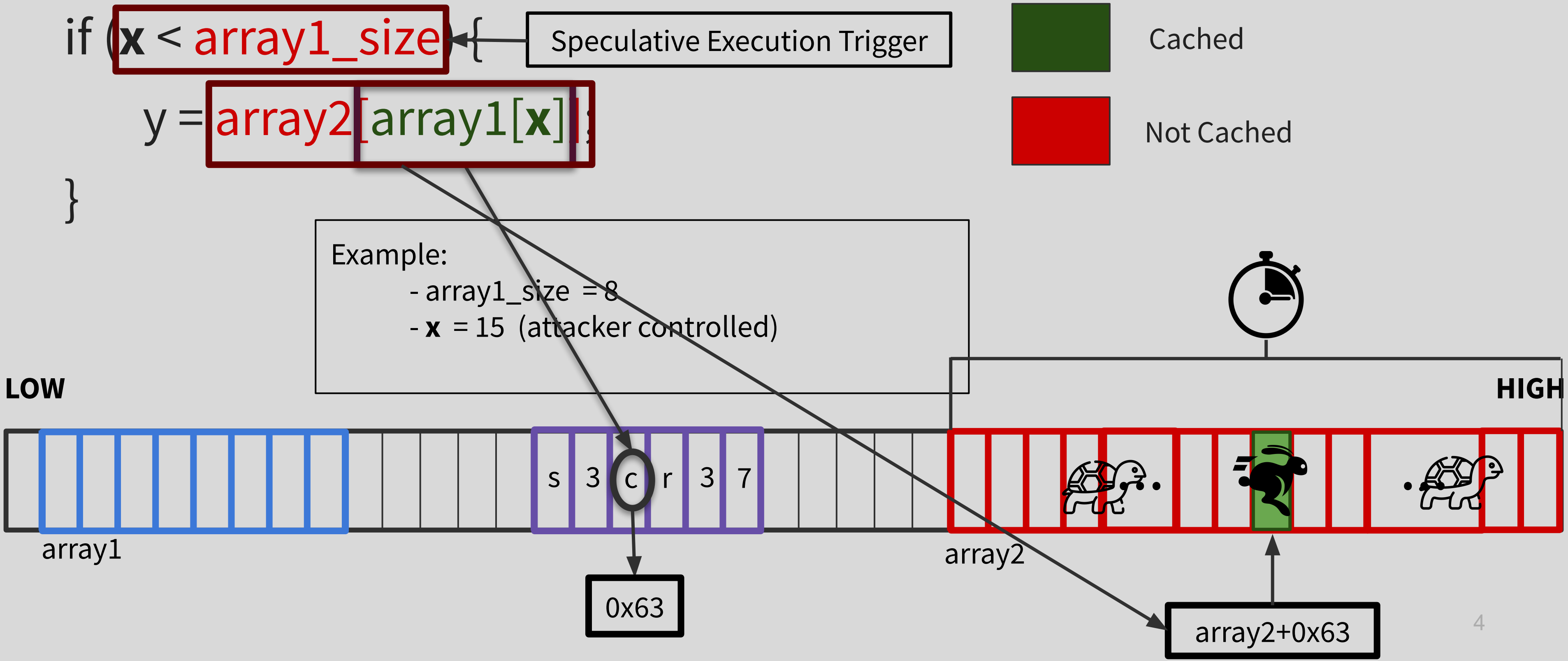
AMD

Multiple vulnerabilities



CVE	Variant	Name
2017-5753	Spectre v1	Bounds Check Bypass
2017-5715	Spectre v2	Branch Target Injection
2017-5754	Meltdown	Rogue Data Cache Load
2018-3640	Spectre v3a	Rogue System Register Read
2018-3639	Spectre v4	Speculative Store Bypass
2018-3665	Spectre-FP	Lazy FP State Restore
2018-3693	Spectre v1.1	Bounds Check Bypass Store

Spectre v1 - Bounds Check Bypass



Speculative CFH Attack Breakdown

Attacker injection

(e.g. Branch Predictor Training)

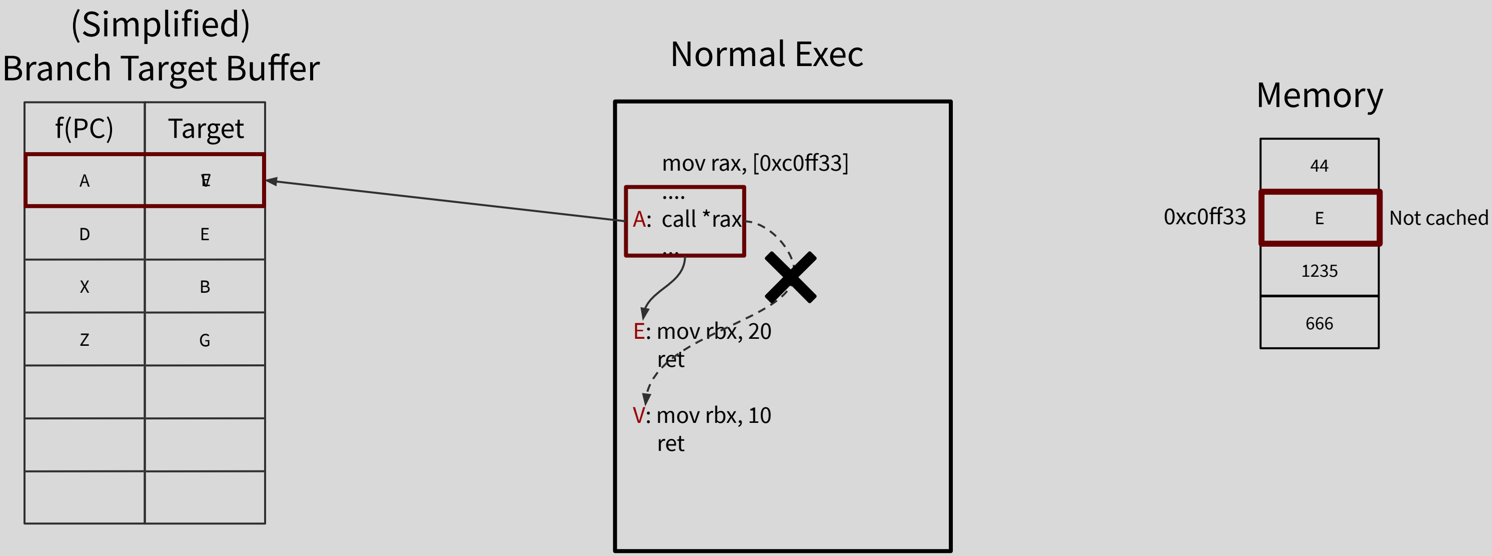
Speculative Control Flow Hijack

lure the victim to execute the vulnerable code

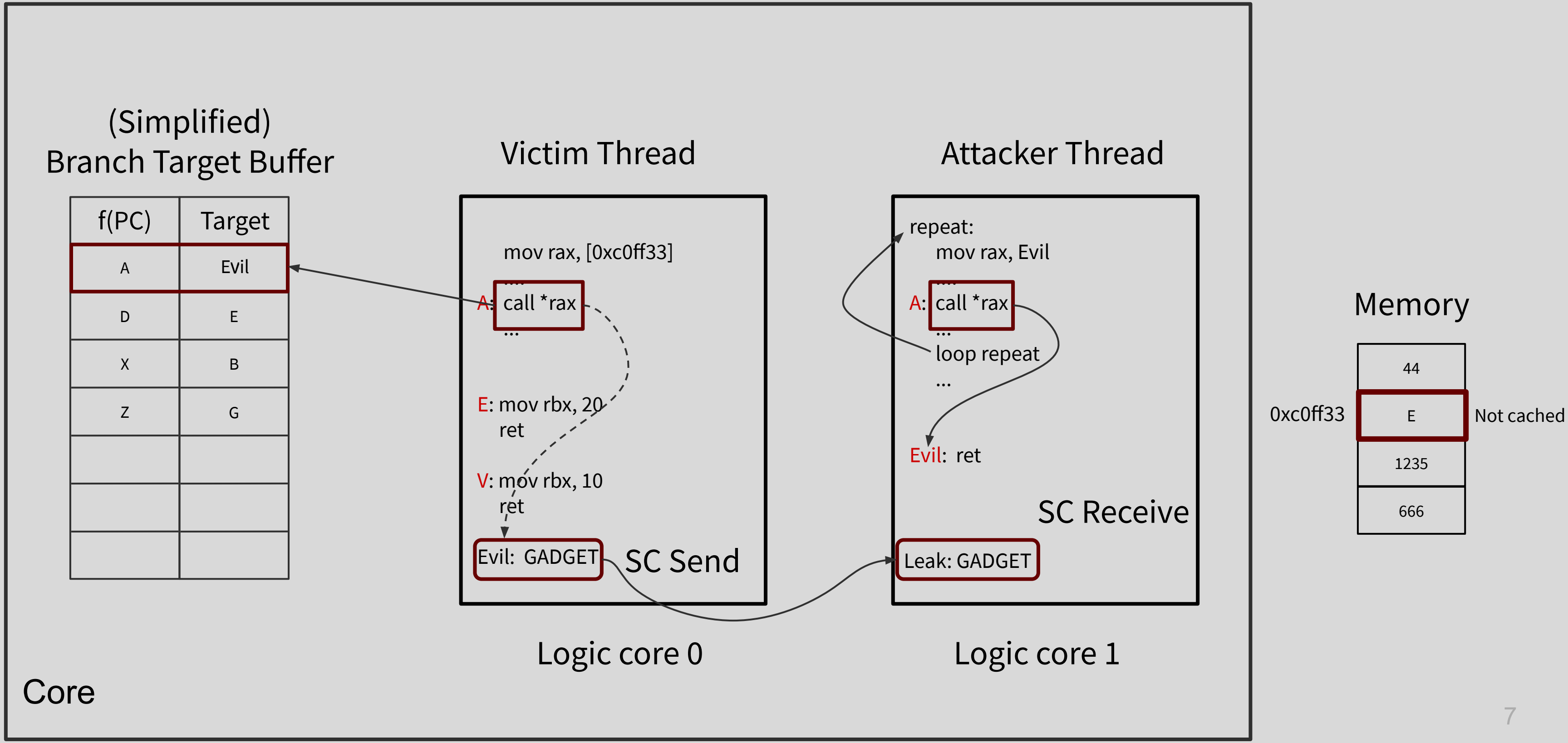
Side Channel Send gadget executed inside the victim

Side Channel Receive gadget executed inside the attacker

Branch Target Buffer



Spectre v2 - Branch Target Injection (BTI)



Control Flow Hijack - Gadget

Spectre v2 and other CF hijack techniques uses Spectre v1 gadget as “*side channel send*”

Project Zero Spectre v2 Proof-of-Concept relies on Kernel **e**xtended **B**erkeley **P**acket **F**ilter (eBPF) JIT mechanism to inject a suitable gadget

Are there other (easier to find) gadgets that can be used?

Our Contribution - New SC Send gadgets

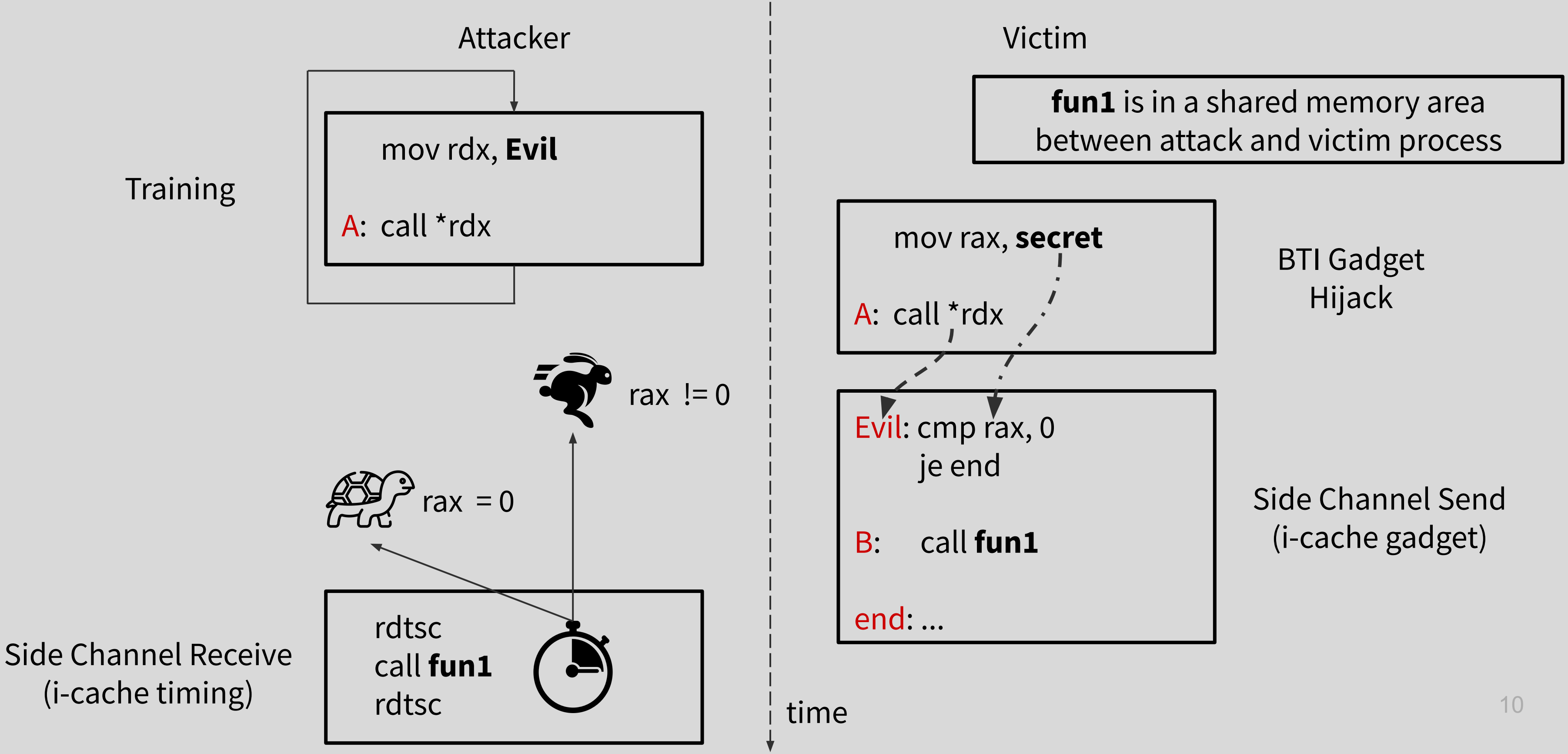
Instruction cache:

timing the execution of a piece of code that is executed conditionally based on a secret

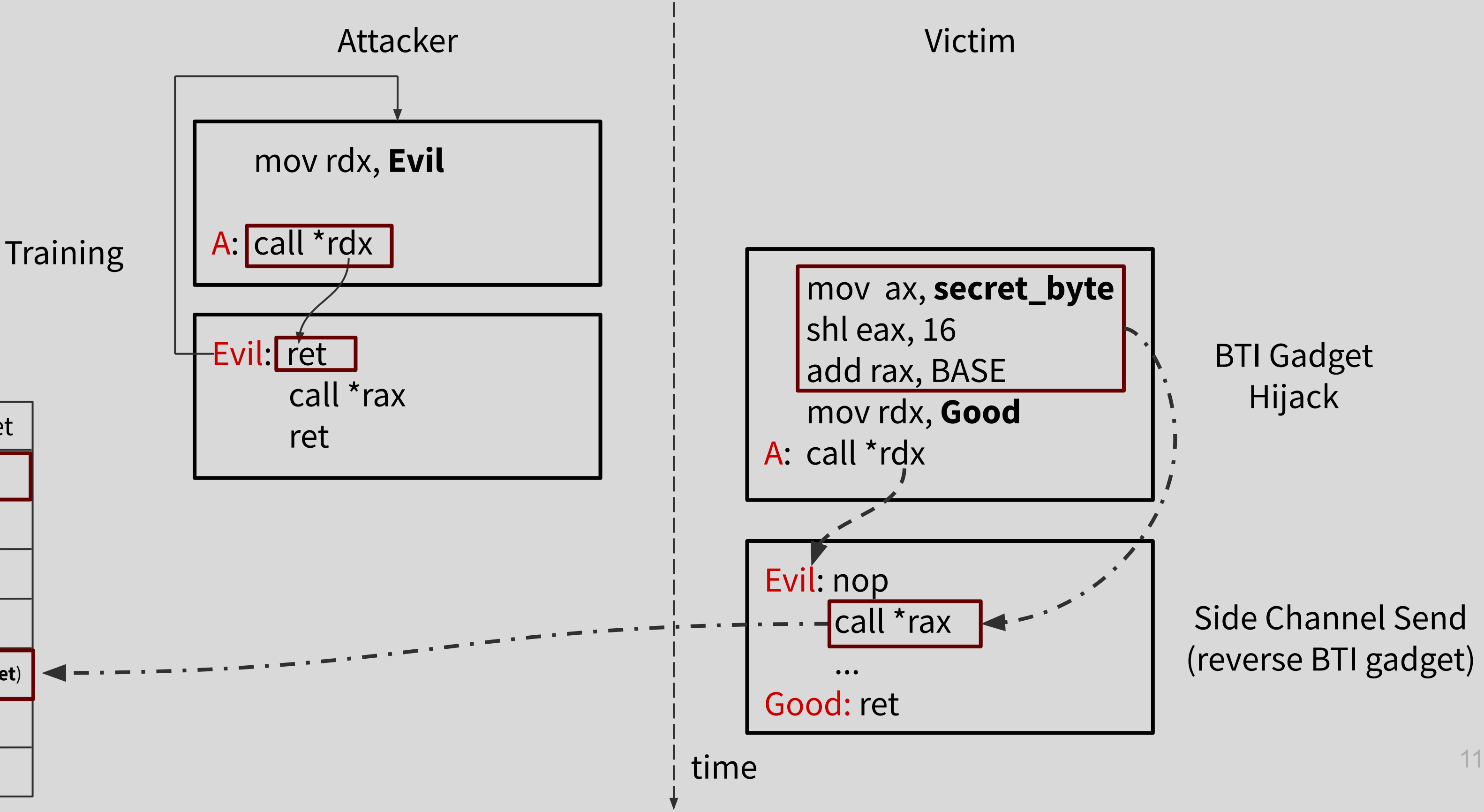
Branch Predictor (Double BTI):

let the victim program train the Branch Predictor using a secret computed value

Instruction Cache - POC

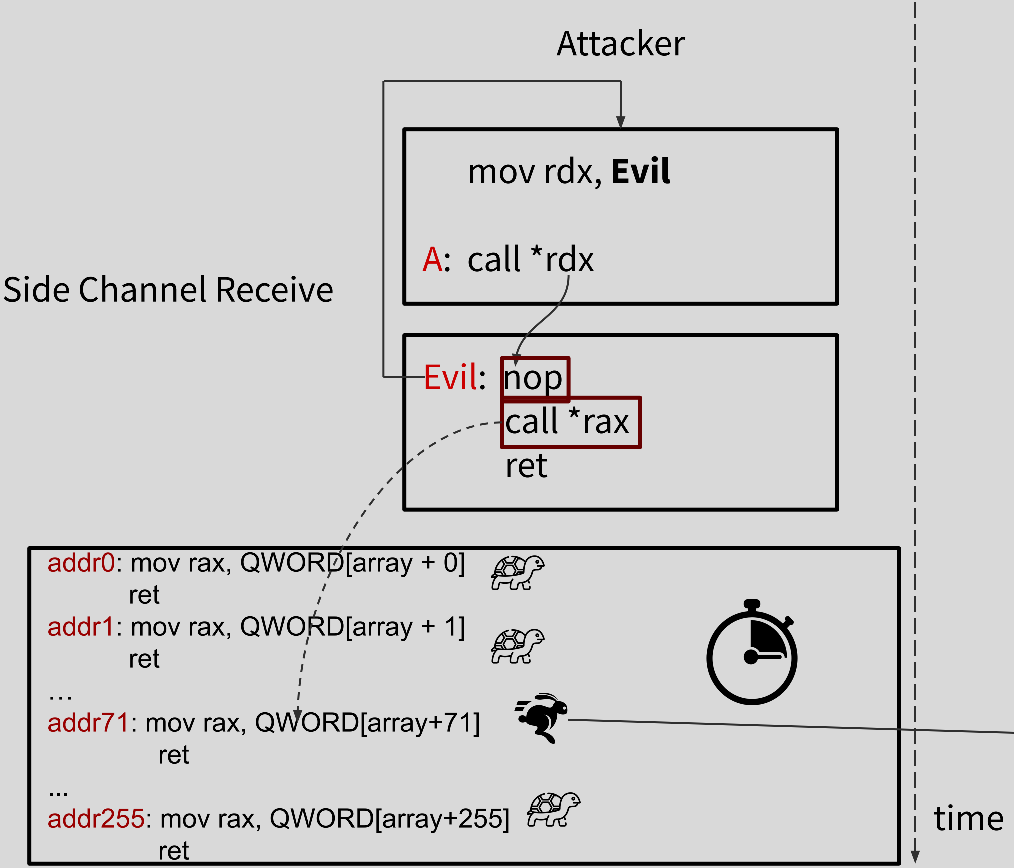


Double BTI - POC Phase 1



f(PC)	Target
A	Evil
D	E
X	B
Z	G
Evil	fun(**secret**)

Double BTI - POC Phase 2



f(PC)	Target
A	Evil
D	E
X	B
Z	G
Evil	fun(**secret**)

fun(**secret**) => addrX with $\{X \in \mathbb{N} \mid X \in [0, 255]\}$

e.g. fun(0) = addr0, fun(255) = addr255

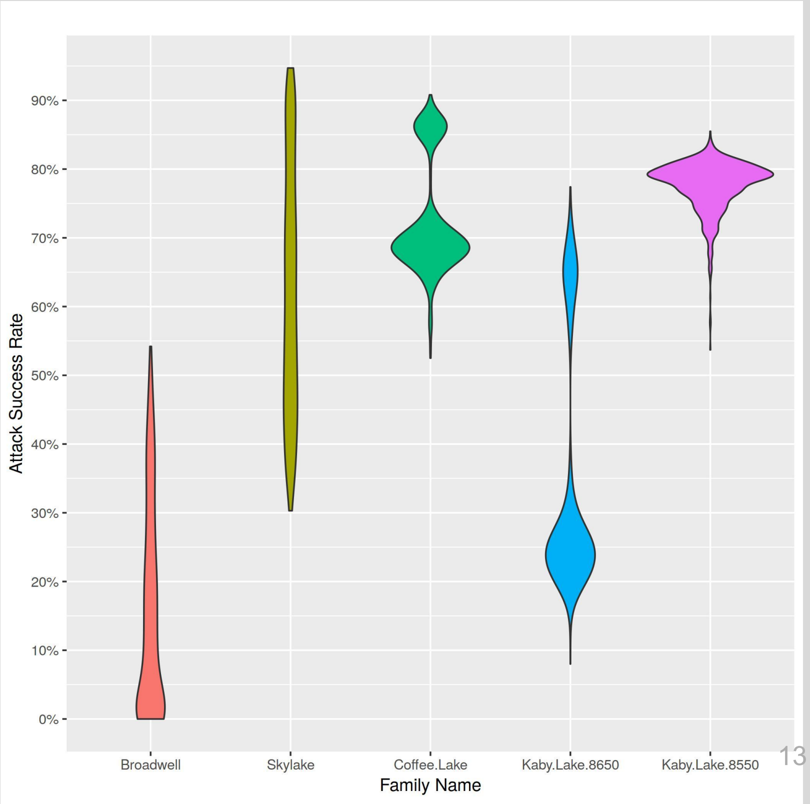
secret_byte = 71 = ‘G’

Results

Icache attack

Secret	Success Rate
0	80.84% +/- 1.37
1	97.29% +/- 0.11

Double BTI



Mitigations

Indirect Branch Restricted Speculation (**IBRS**) and Indirect Branch Predictor Barrier (**IBPB**) does not apply to user-space attacks.

Single Thread Indirect Branch Predictors (**STIBP**) mitigates our attacks

Current **STIBP** default setting leaves to the application the burden of requesting the protection through either **SECCOMP**, or the **prctl** interface.

Retpoline stops our attacks, though the application has to be recompiled with it

Conclusions

We introduced two new SC send gadgets and tested them in BTI attacks (applicable to other Control Flow Hijack attacks, e.g. ret2spec)

Through the **I-cache** gadget we can leak **1 bit** at the time

Through the **double BTI** gadget we can leak **1 byte** at the time with very good signal

Current mitigations do not protect applications unless specifically requested

Questions?