black hat USA 2024

AUGUST 7-8, 2024

BRIEFINGS

Arbitrary Data Manipulation and Leakage with CPU Zero-Day Bugs on RISC-V

Fabian Thomas, Lorenz Hetterich

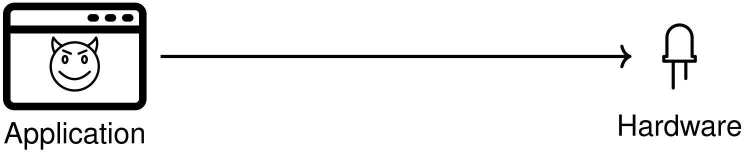
The Advertisement Panel

The Challenge: Operating System

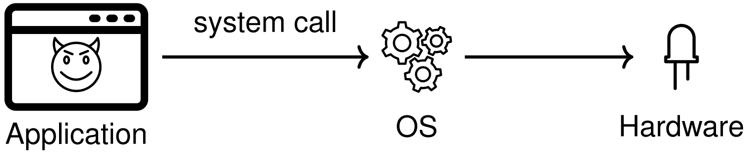
Application

Hardware

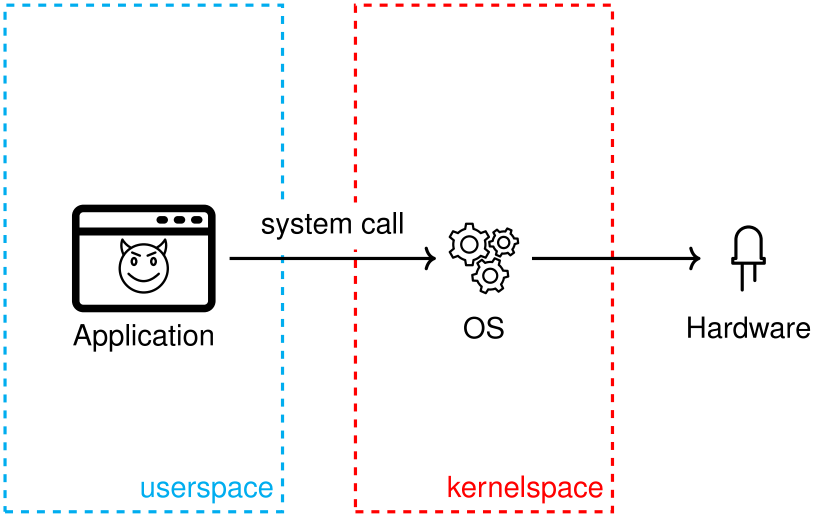
The Challenge: Operating System



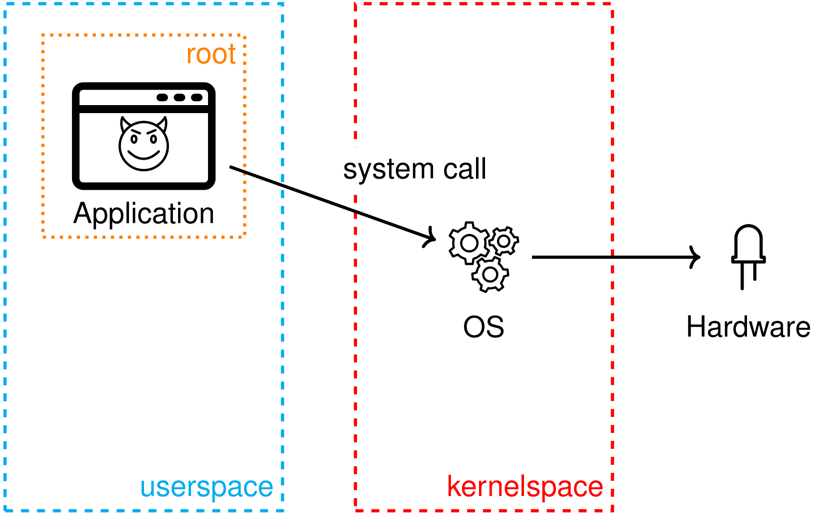
The Challenge: Operating System



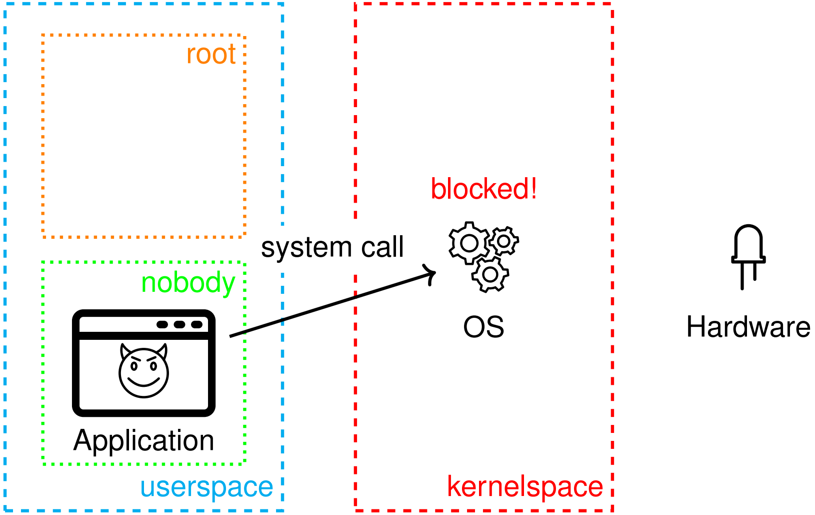
The Challenge: Operating System



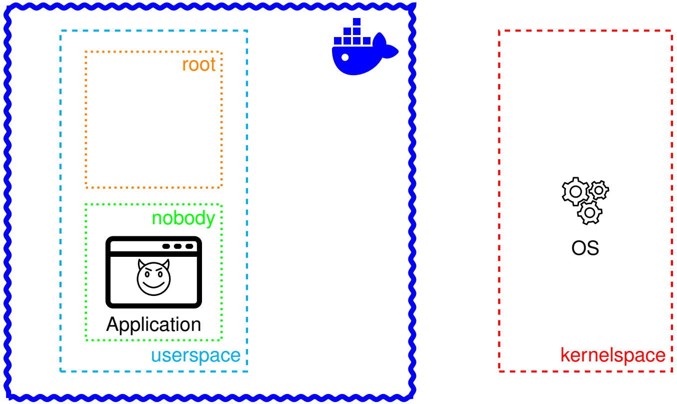
The Challenge: Sandboxing



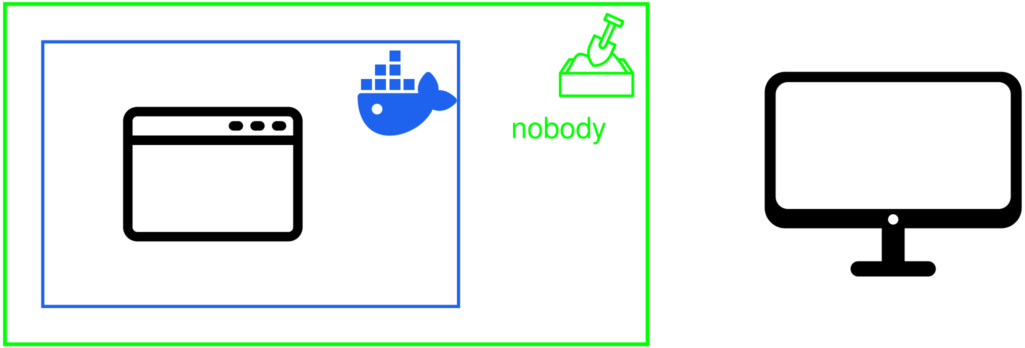
The Challenge: Sandboxing



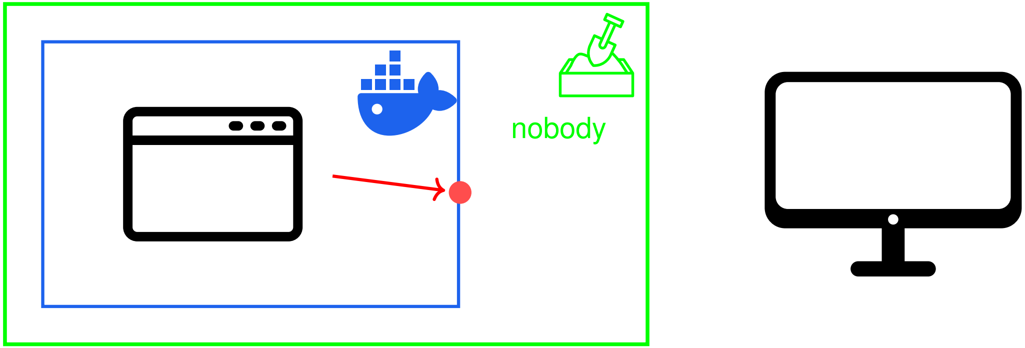
The Challenge: Container



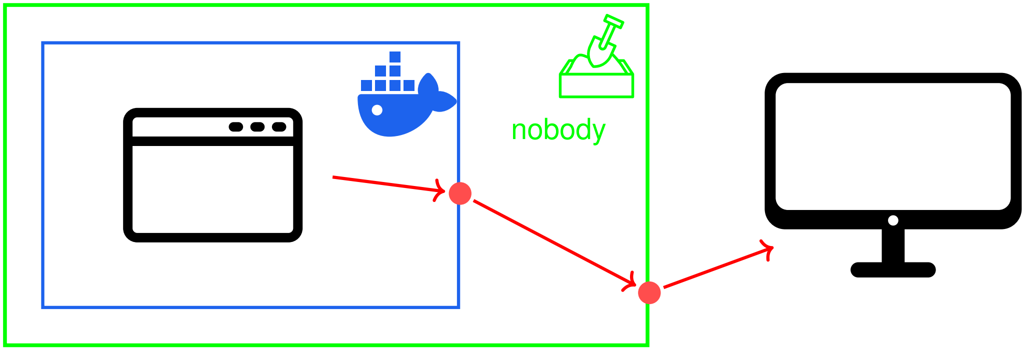
The Challenge: Full Isolation



The Challenge: Full Isolation



The Challenge: Full Isolation



You can't break out. . .

~~You can’t break out. . . .~~: GhostWrite

Who are we?

Fabian Thomas

PhD student @CISPA (Germany)

E-Mail	fabian.thomas@cispa.de
Web	fabianthomas.de
Twitter	@fth0mas

Who are we?

Lorenz Hetterich

PhD student @CISPA (Germany)

E-Mail lorenz.hetterich@cispa.de

Twitter @hetterichlorenz

Who are we?

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

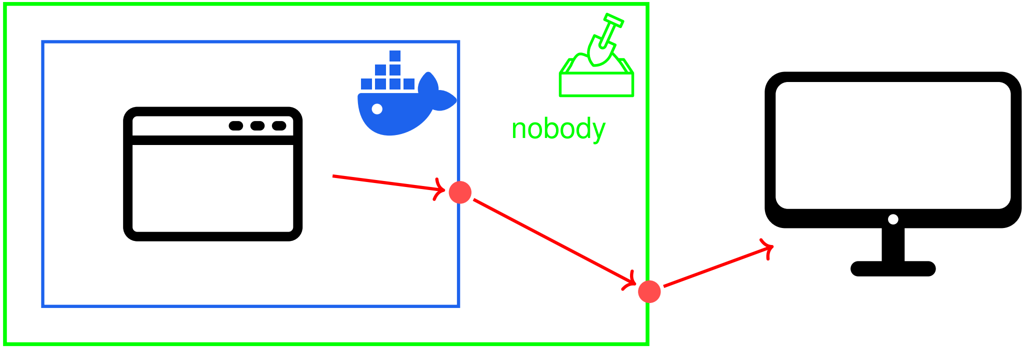
Research Group Schwarz

- Research focus:
  - Hardware vulnerabilities
  - . . . from software
- Recent discoveries:

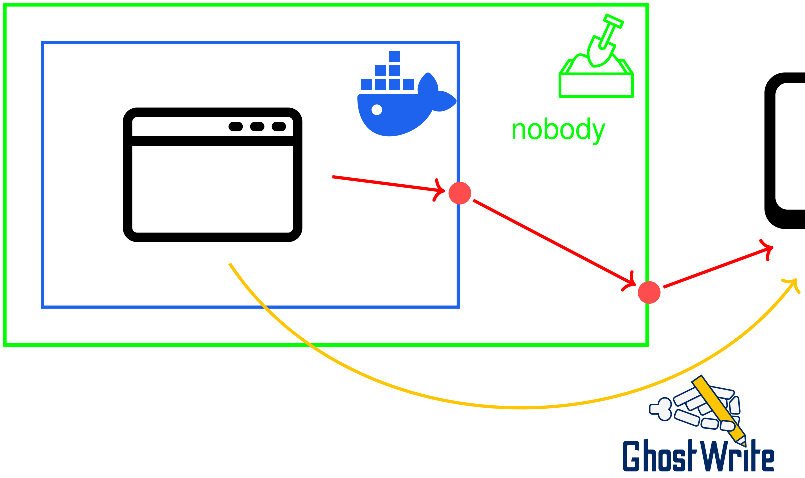
GhostWrite

CACHE WARP

~~Full Isolation~~: GhostWrite



~~Full Isolation~~: GhostWrite



~~Full Isolation~~: GhostWrite

Software World

~~Full Isolation~~: GhostWrite

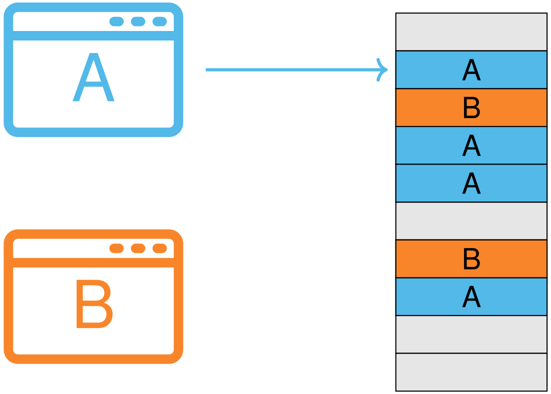
Software World

Hardware World

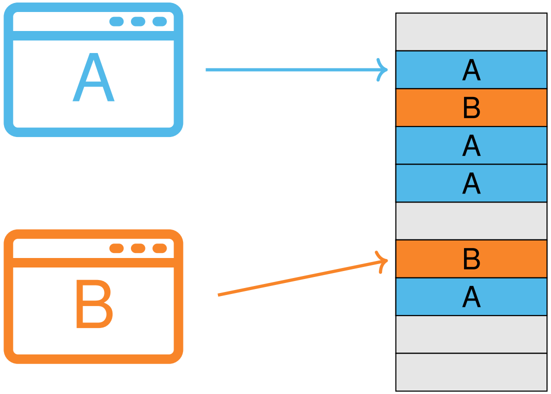
GhostWrite

$\longrightarrow$

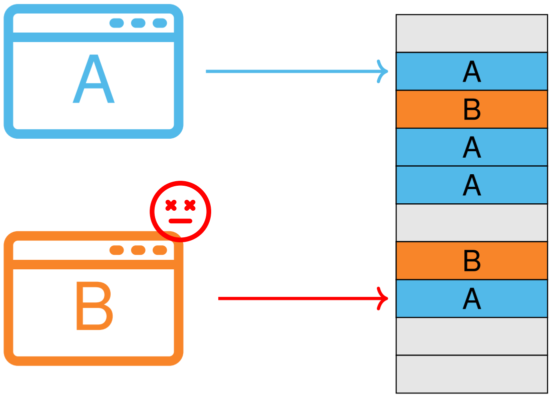
Memory Isolation



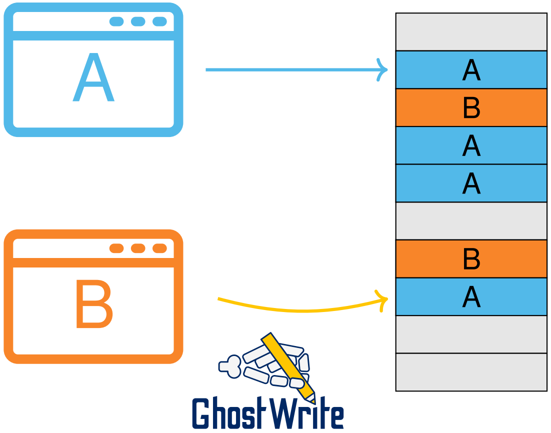
Memory Isolation



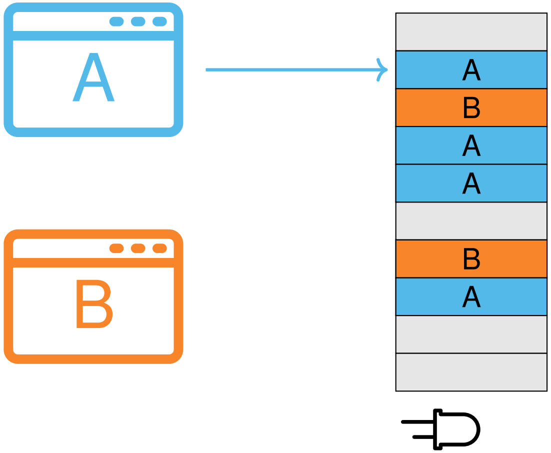
Memory Isolation



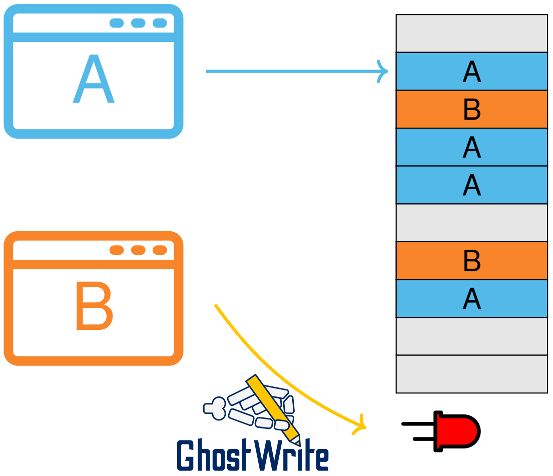
~~Memory Isolation~~: GhostWrite



~~Memory Isolation~~: GhostWrite



~~Memory Isolation~~: GhostWrite



The Exploit

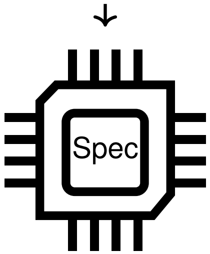
```
mv t0, phys_addr
vmv.v.x v0, value
vsetvli zero, zero, e8, m1
vse128.v v0, 0(t0)
```

The Exploit

```
mv t0, phys_addr
vmv.v.x v0, value
vsetvli zero, zero, e8, m1
vse128.v v0, 0(t0)
```

GhostWrite: Investigation

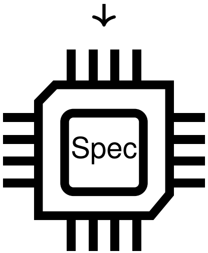
```
vse128.v v0, 0(t0)
```



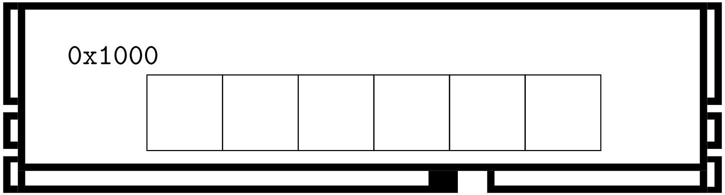
t0	v0	v1	v2	v3	v4	v5
0x1000	R	I	S	C	-	V

GhostWrite: Investigation

```
vse128.v v0, 0(t0)
```

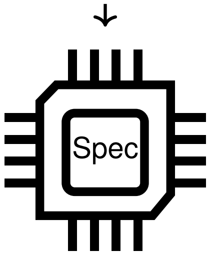


t0	v0	v1	v2	v3	v4	v5
0x1000	R	I	S	C	-	V

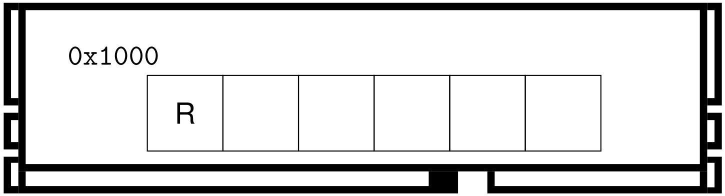


GhostWrite: Investigation

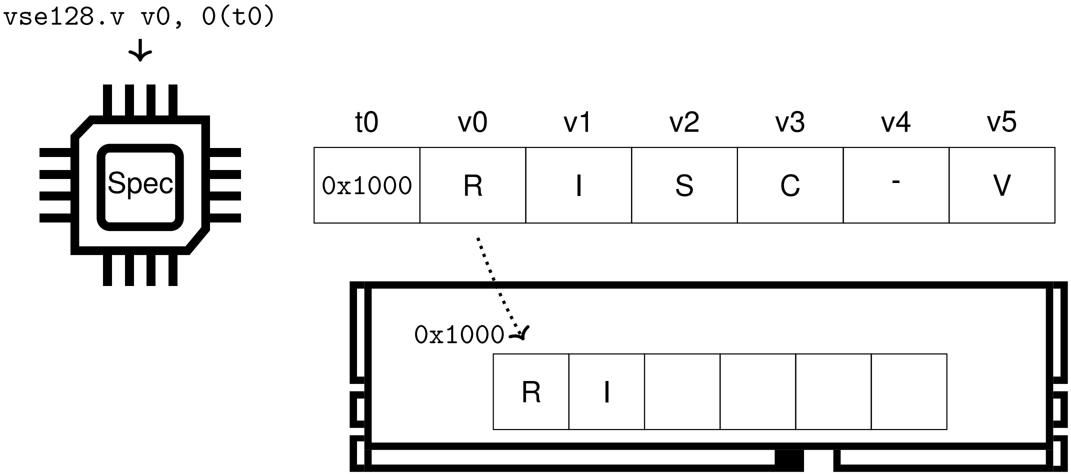
```
vse128.v v0, 0(t0)
```



t0	v0	v1	v2	v3	v4	v5
0x1000	R	I	S	C	-	V

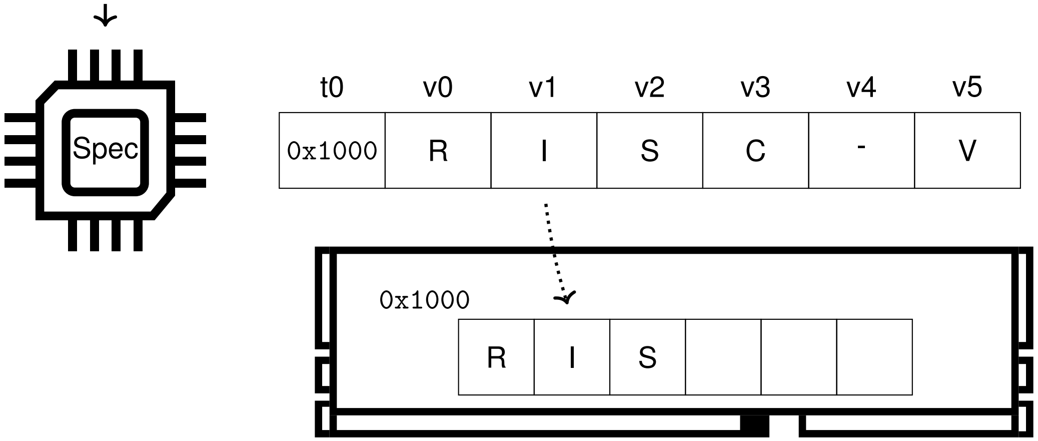


GhostWrite: Investigation



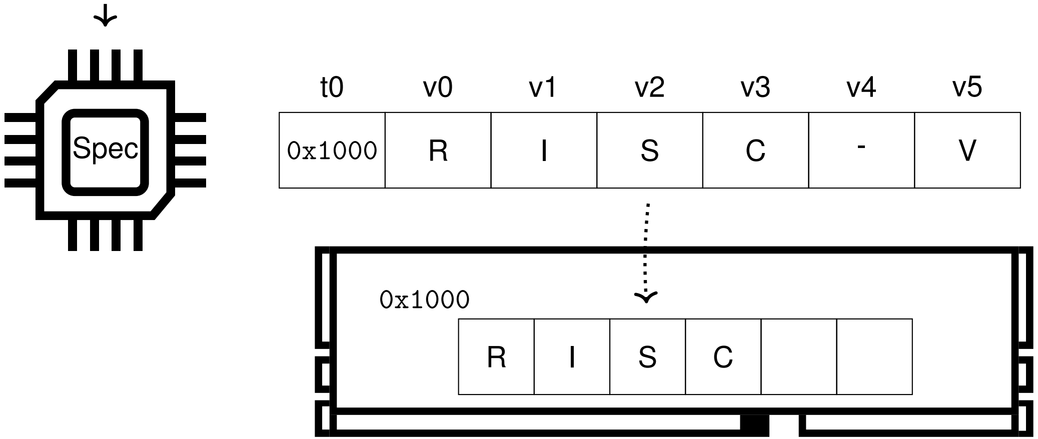
GhostWrite: Investigation

```
vse128.v v0, 0(t0)
```



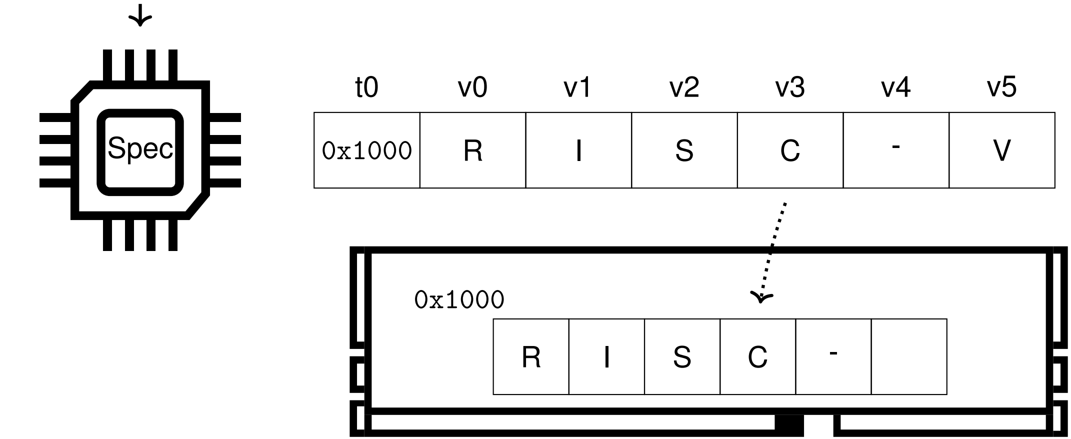
GhostWrite: Investigation

```
vse128.v v0, 0(t0)
```

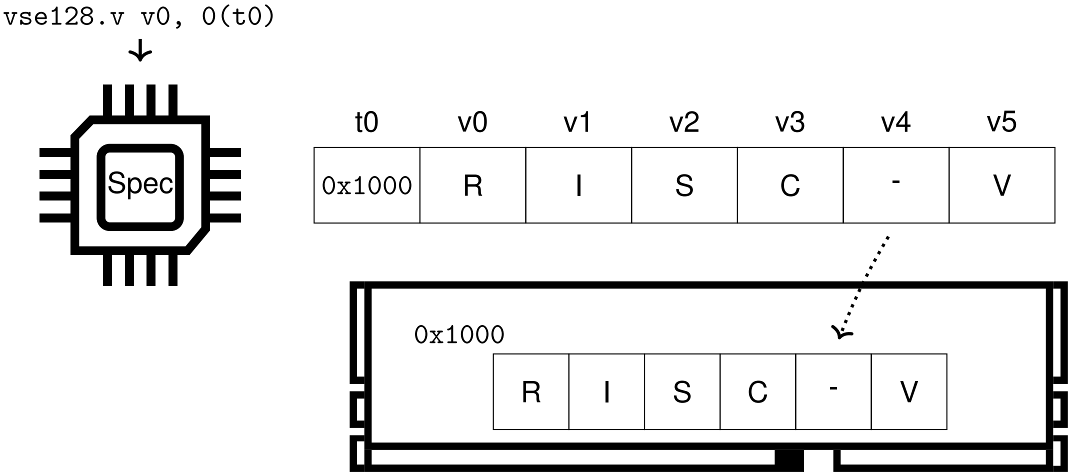


GhostWrite: Investigation

```
vse128.v v0, 0(t0)
```

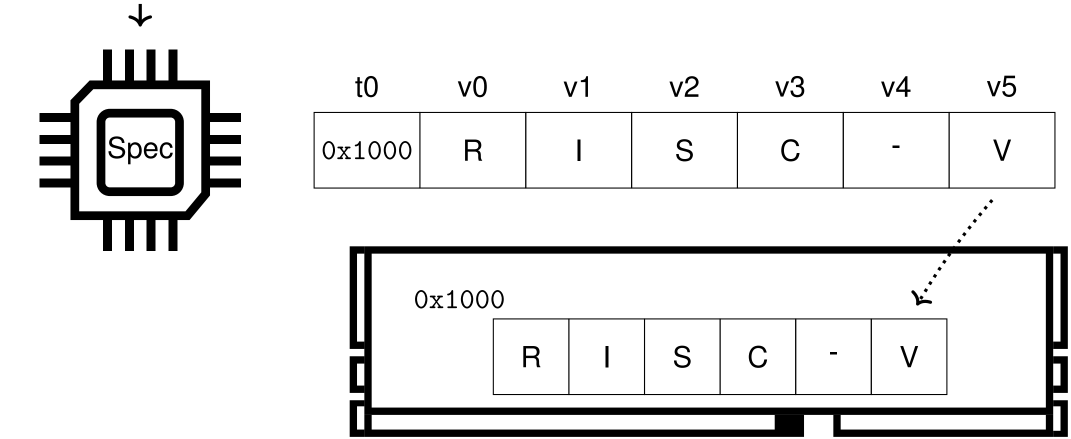


GhostWrite: Investigation



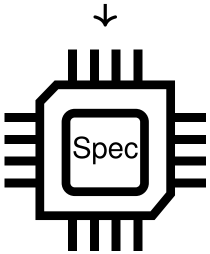
GhostWrite: Investigation

```
vse128.v v0, 0(t0)
```

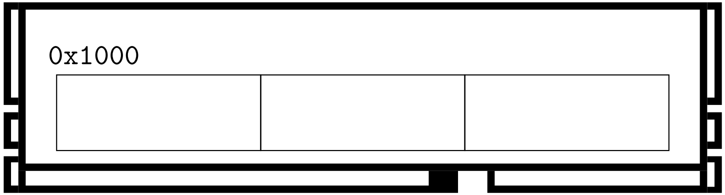


GhostWrite: Vector Instructions

```
vse128.v v0, 0(t0)
```

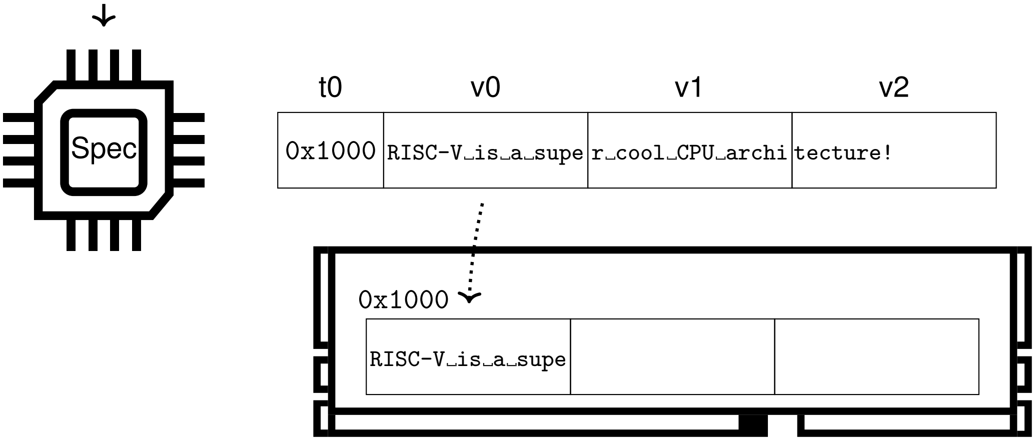


t0	v0	v1	v2
0x1000	RISC-V␣is␣a␣supe	r␣cool␣CPU␣archi	tecture!



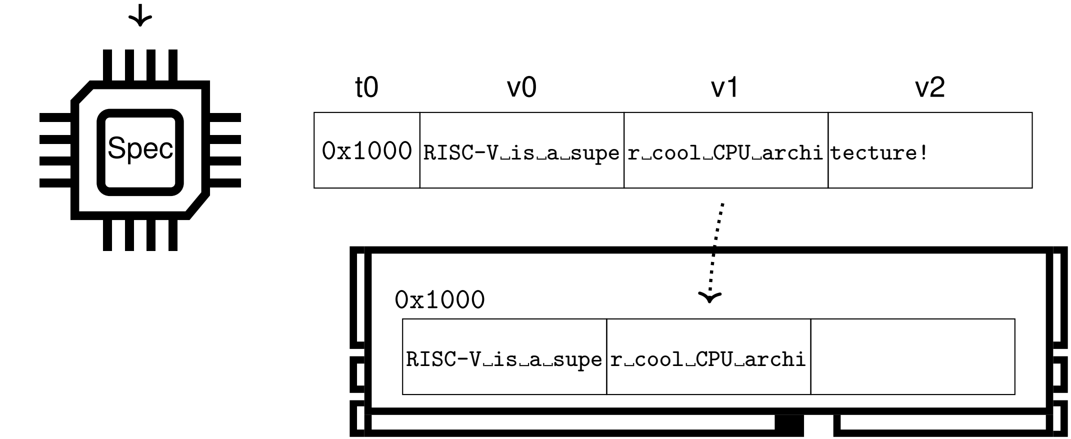
GhostWrite: Vector Instructions

```
vse128.v v0, 0(t0)
```



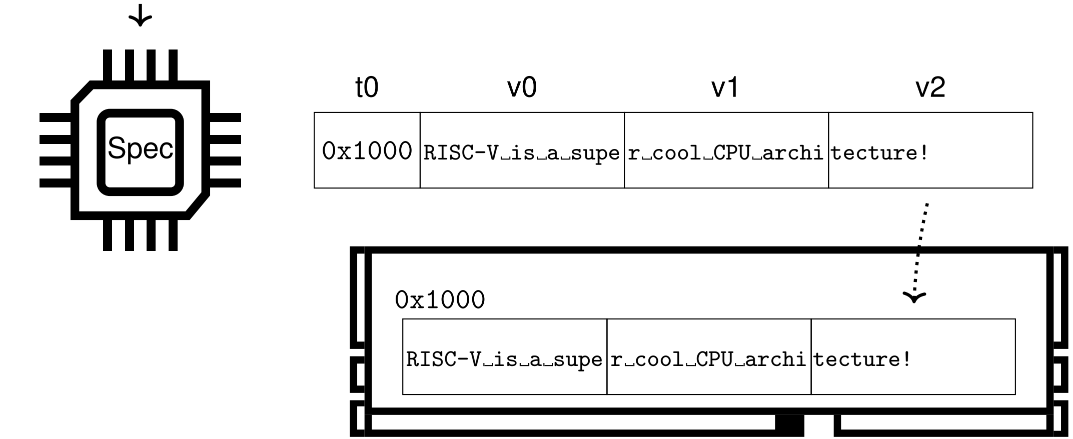
GhostWrite: Vector Instructions

`vse128.v v0, 0(t0)`

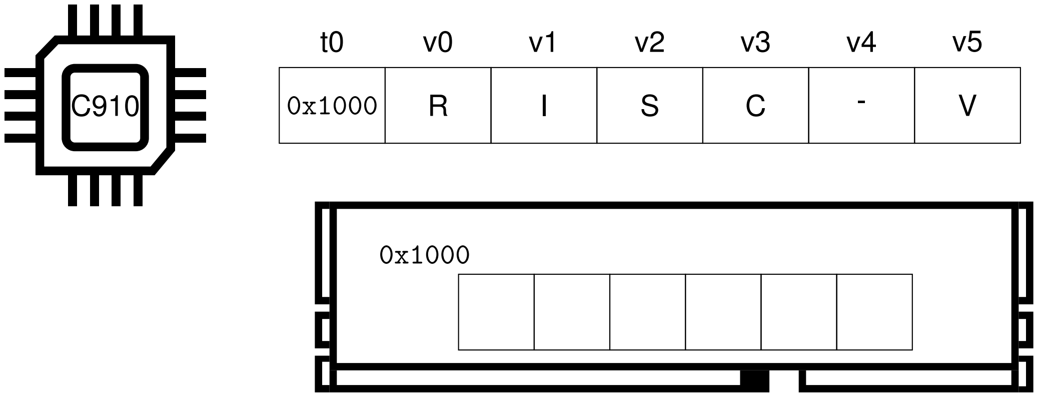


GhostWrite: Vector Instructions

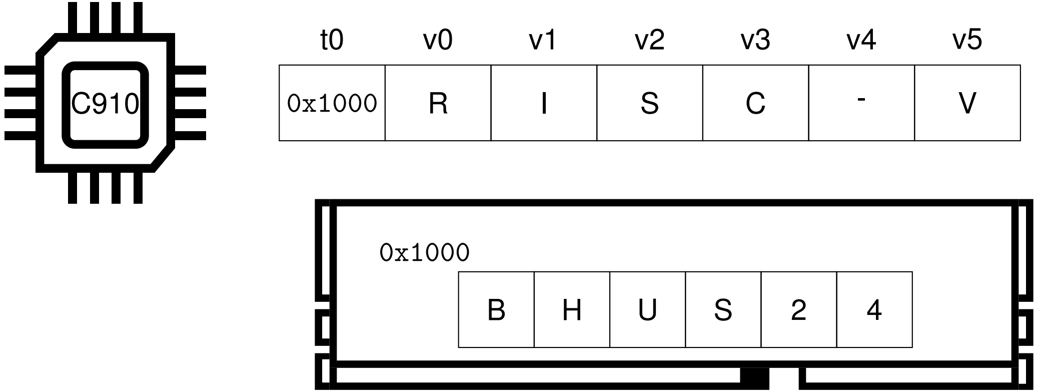
```
vse128.v v0, 0(t0)
```



GhostWrite: Investigation

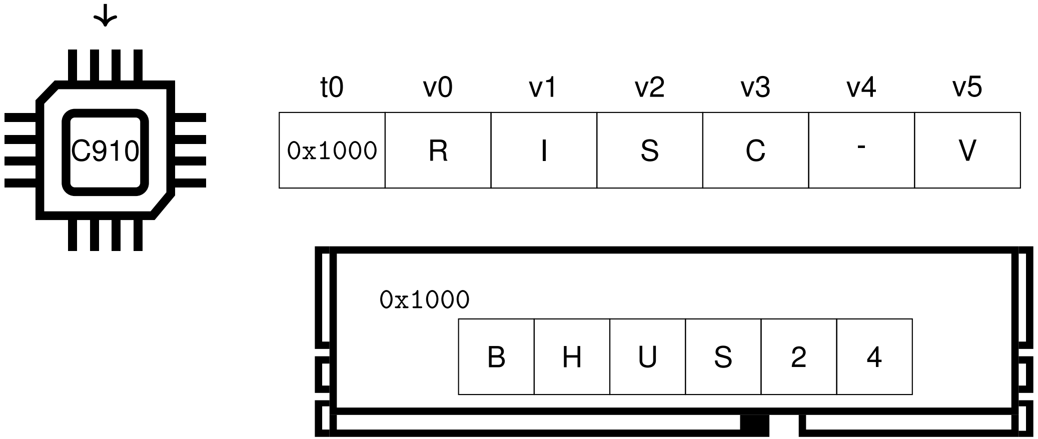


GhostWrite: Investigation

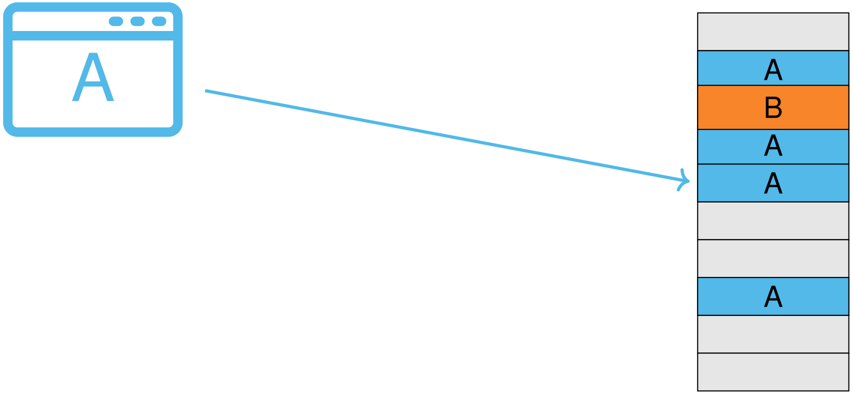


GhostWrite: Investigation

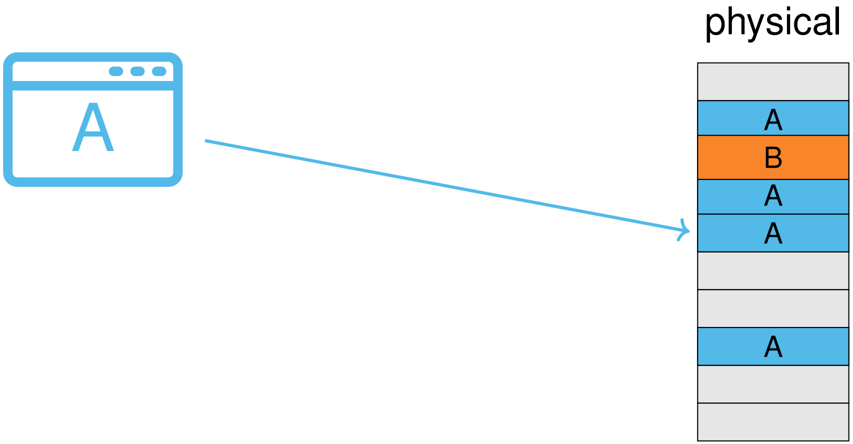
```
vse128.v v0, 0(t0)
```



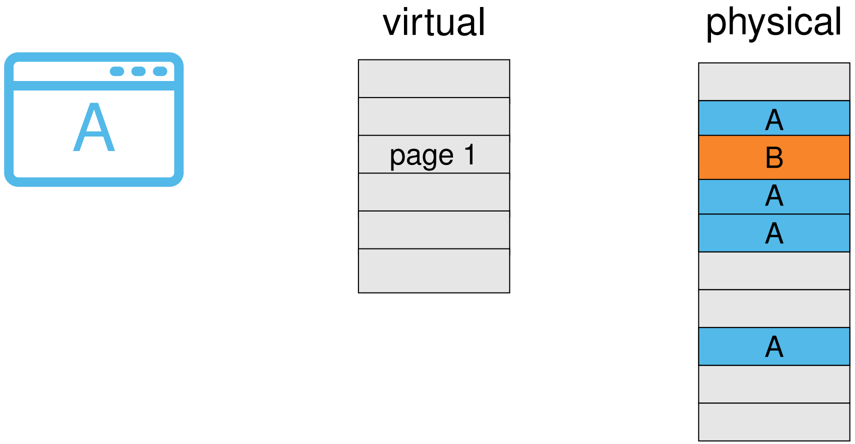
GhostWrite: Virtual Memory



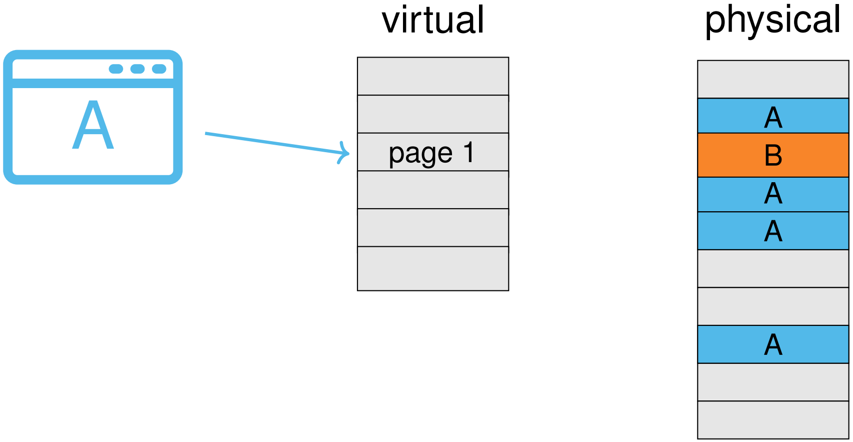
GhostWrite: Virtual Memory



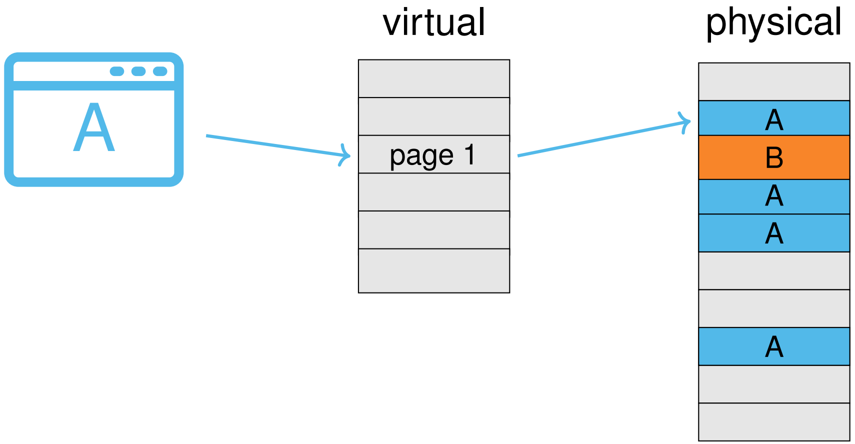
GhostWrite: Virtual Memory



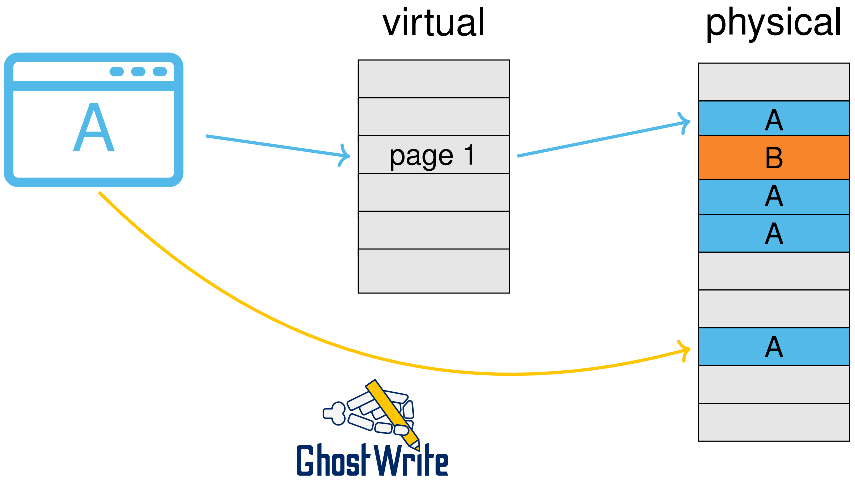
GhostWrite: Virtual Memory



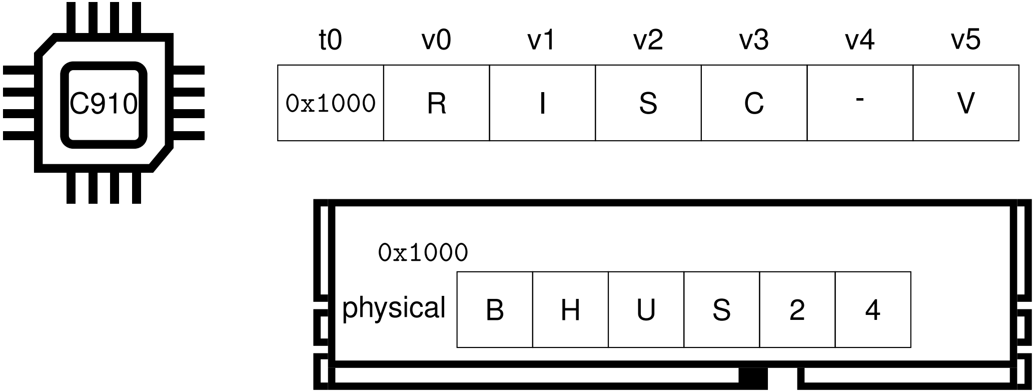
GhostWrite: Virtual Memory



GhostWrite: Virtual Memory

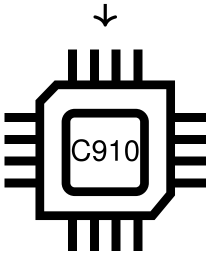


GhostWrite: Investigation

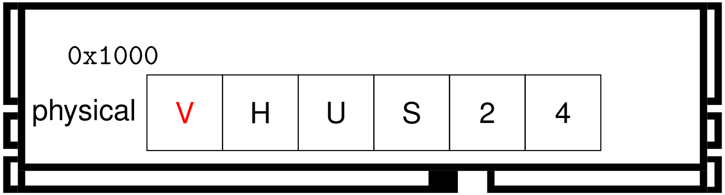


GhostWrite: Investigation

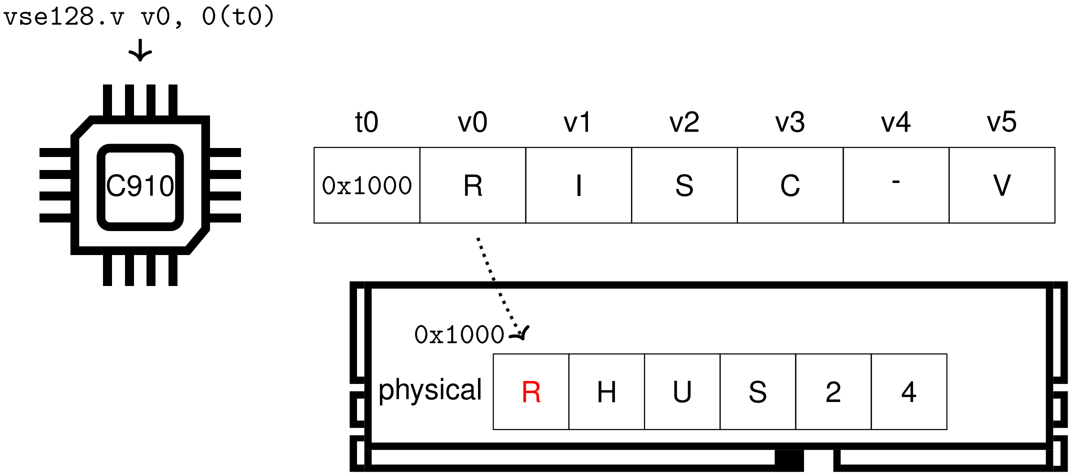
```
vse128.v v0, 0(t0)
```



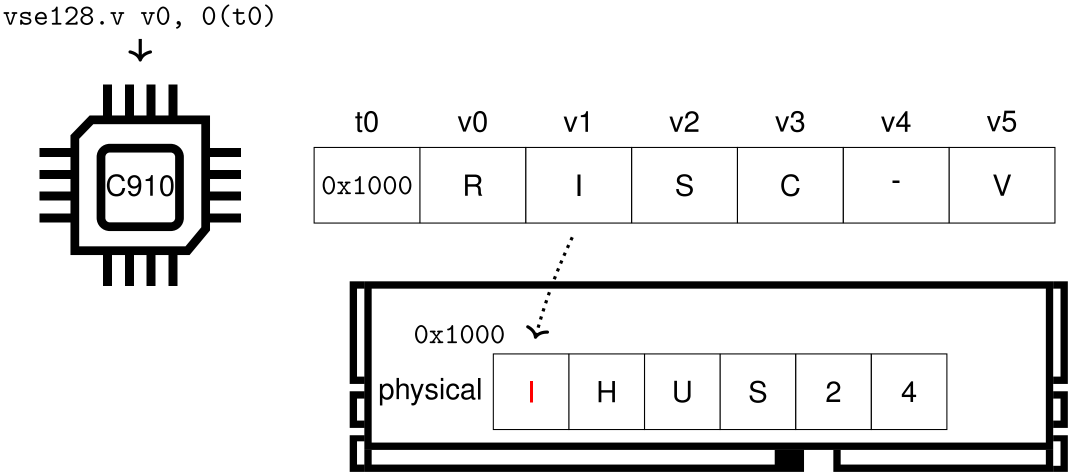
t0	v0	v1	v2	v3	v4	v5
0x1000	R	I	S	C	-	V



GhostWrite: Investigation

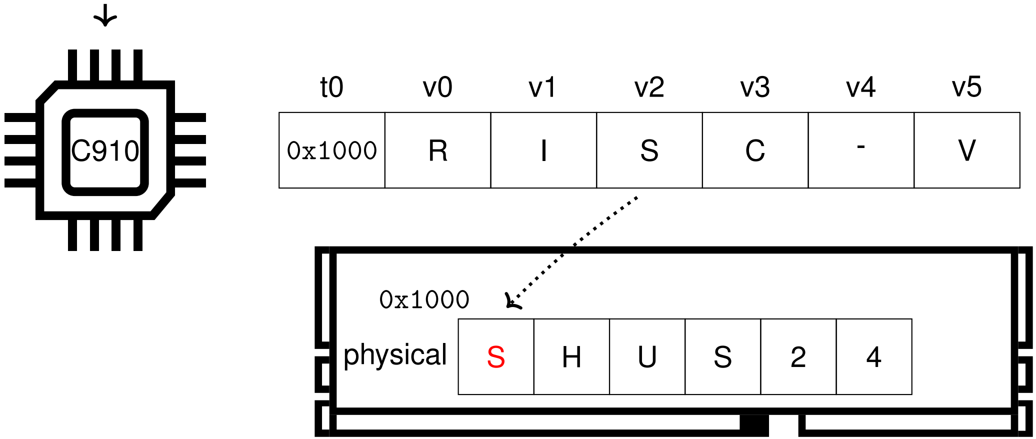


GhostWrite: Investigation



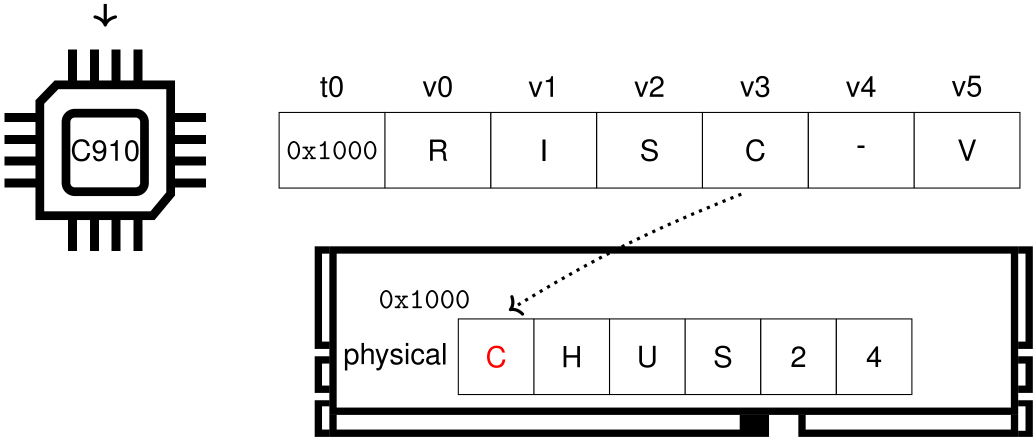
GhostWrite: Investigation

```
vse128.v v0, 0(t0)
```

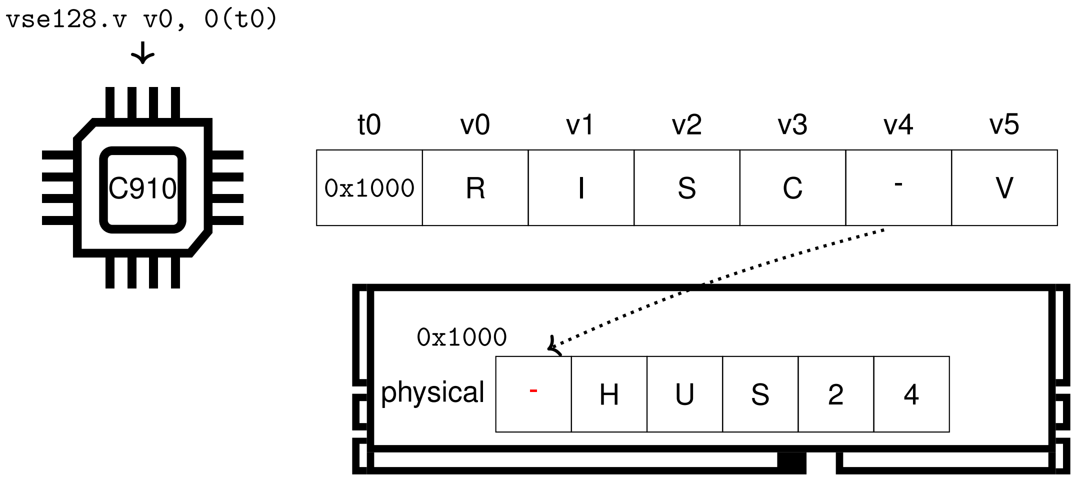


GhostWrite: Investigation

```
vse128.v v0, 0(t0)
```

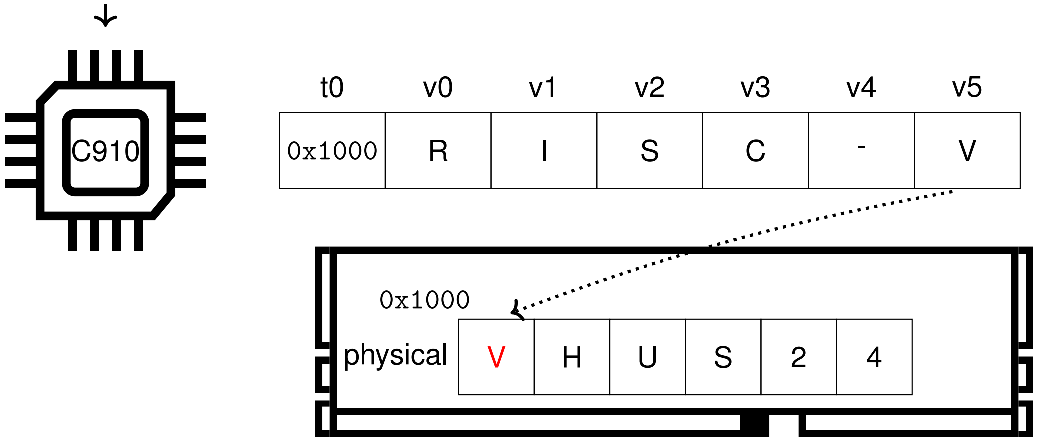


GhostWrite: Investigation



GhostWrite: Investigation

```
vse128.v v0, 0(t0)
```



Is every system vulnerable?

Is every system vulnerable?

x86

arm

RISC-V®

AMD

intel®

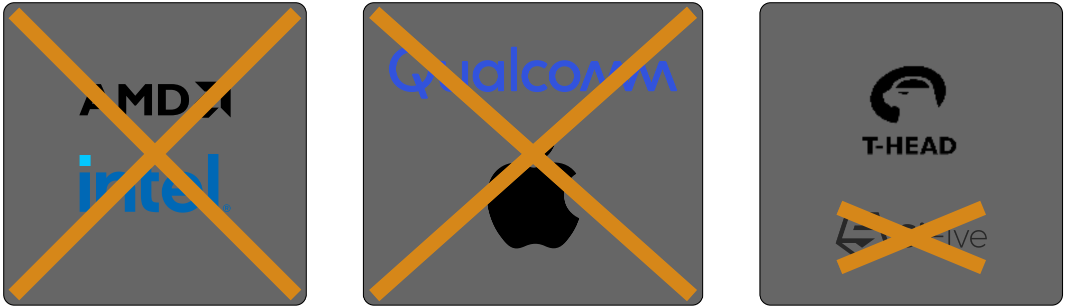
Qualcomm

T-HEAD

SiFive

Is every system vulnerable?

x86 | arm | RISC-V®



T-Head XuanTie C910

- one of the fastest RISC-V CPUs

T-Head XuanTie C910

- one of the fastest RISC-V CPUs
- 4 cores, 2GHz, vector extension

T-Head XuanTie C910

- one of the fastest RISC-V CPUs
- 4 cores, 2GHz, vector extension
- available in the cloud

Scaleway

T-Head XuanTie C910

- one of the fastest RISC-V CPUs
- 4 cores, 2GHz, vector extension
- available in the cloud
- available in laptops

Scaleway

Instruction Set Architecture (ISA)

Specifies behavior

Instruction Set Architecture (ISA)

Specifies behavior

Defines legal programs

Instruction Set Architecture (ISA)

Specifies behavior

Defines legal programs

Licensing fees

Instruction Set Architecture (ISA)

Specifies behavior

Defines legal programs

Licensing fees

Limited customization

RISC-V

open, community-driven

RISC-V

open, community-driven

no licensing fees

RISC-V

open, community-driven

no licensing fees

well designed

black hat USA 2024

RISC-V

open, community-driven

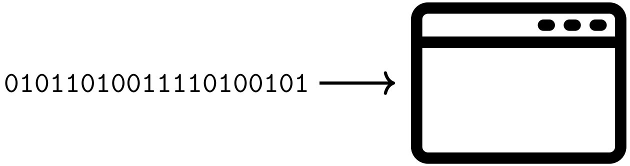
no licensing fees

well designed

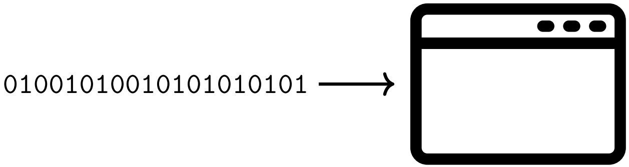
extensible

Software Fuzzing

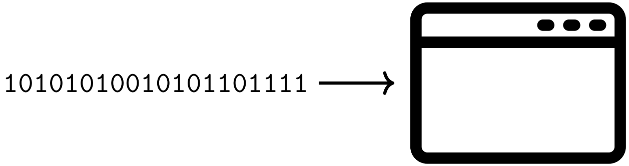
Software Fuzzing



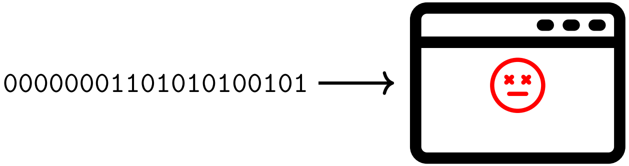
Software Fuzzing



Software Fuzzing

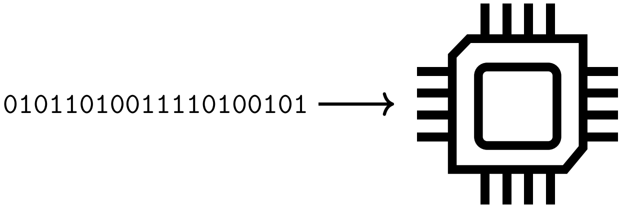


Software Fuzzing

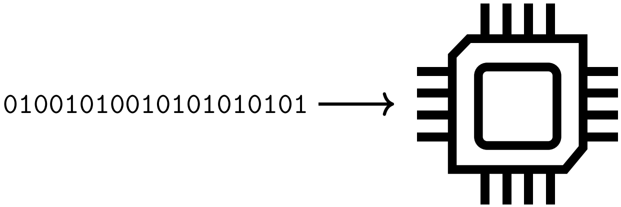


Hardware Fuzzing

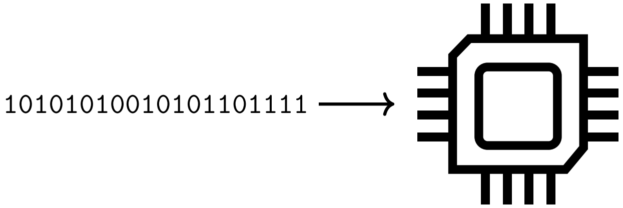
Hardware Fuzzing



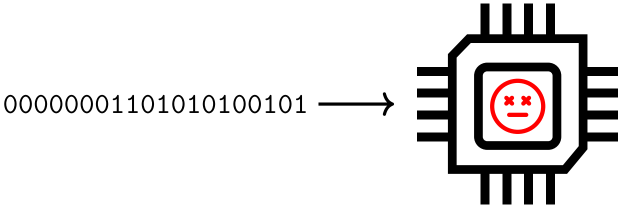
Hardware Fuzzing



Hardware Fuzzing

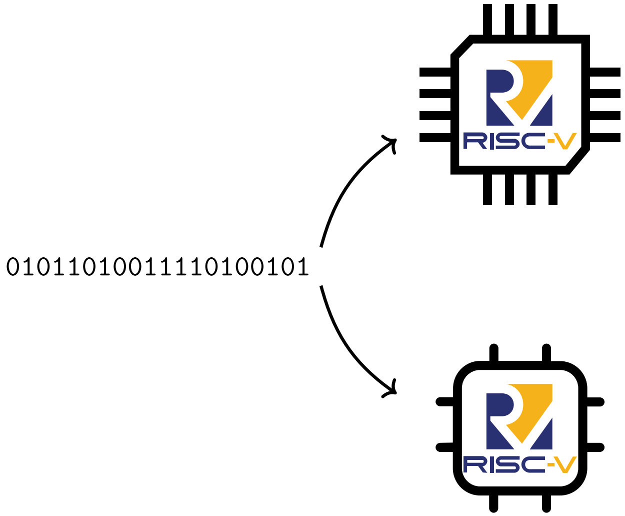


Hardware Fuzzing

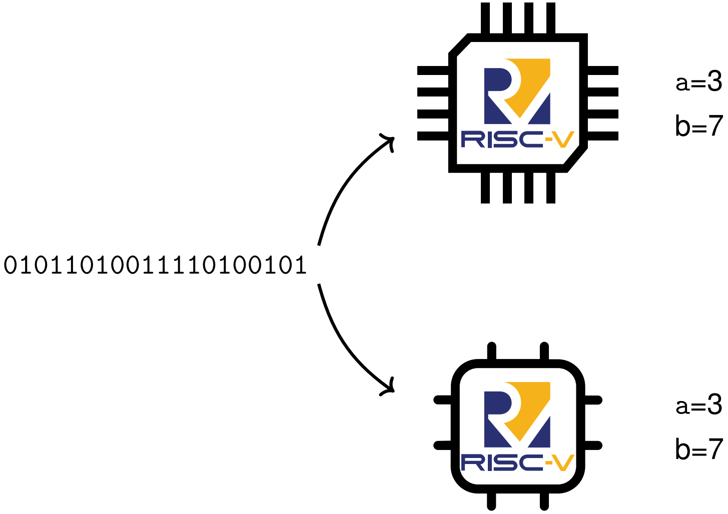


Differential Fuzzing

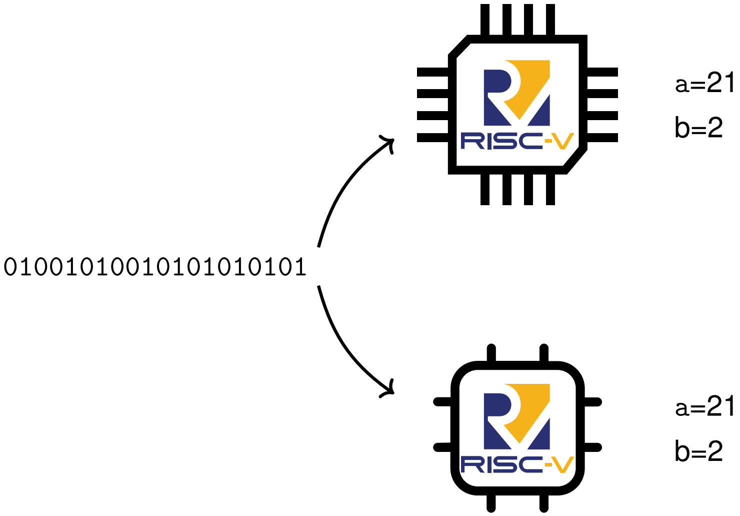
Differential Fuzzing



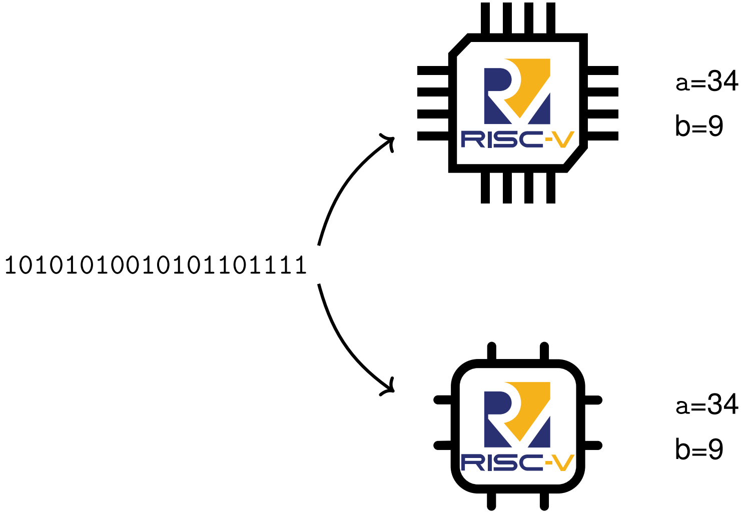
Differential Fuzzing



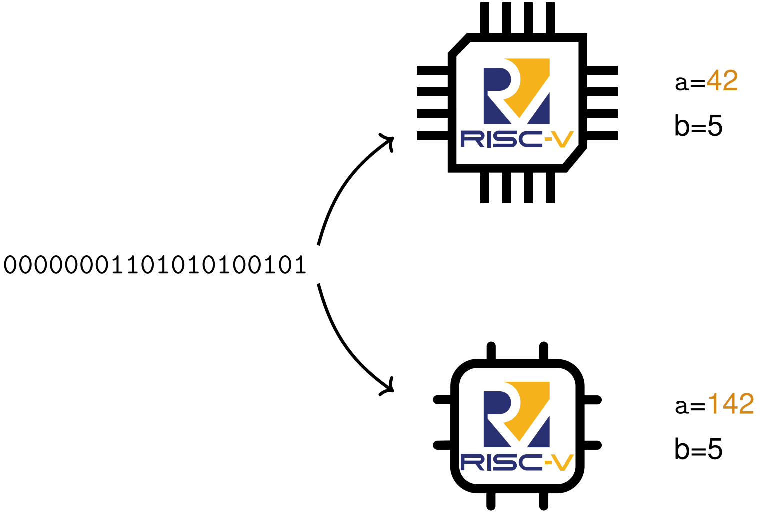
Differential Fuzzing



Differential Fuzzing

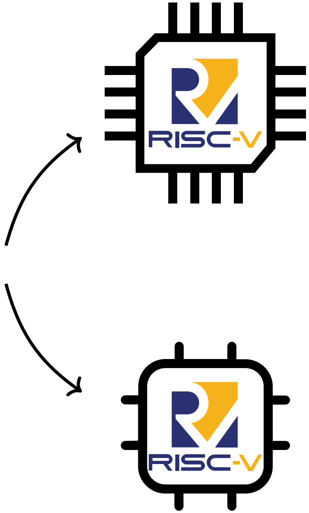


Differential Fuzzing

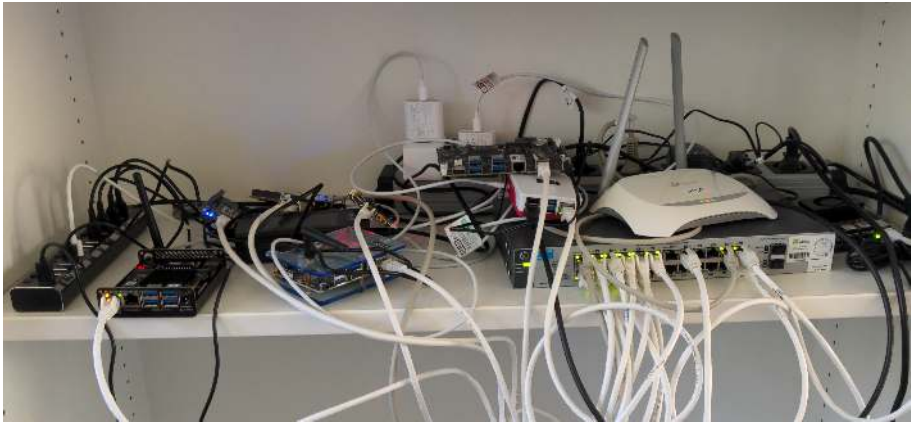


Differential Fuzzing

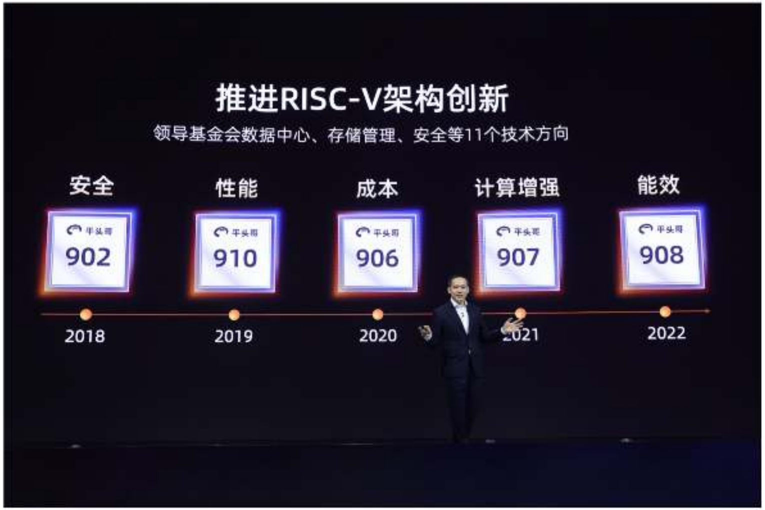
```
fadd f3, f4, f4
li x1, 42


x1: 0
```



Let’s start fuzzing!



C908 and C906



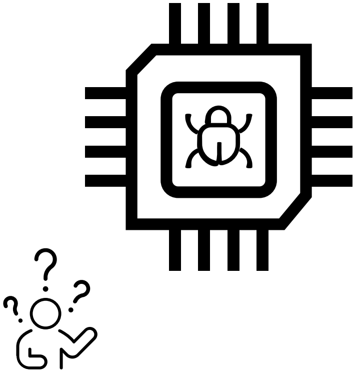
Demo: Freezing the C906

How to fix Hardware Bugs?

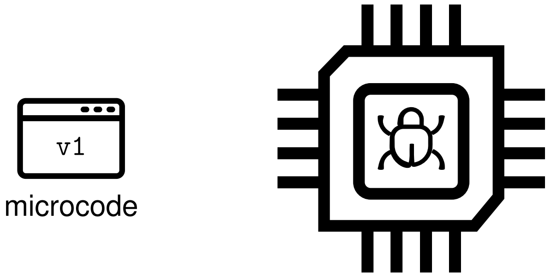
How to fix Hardware Bugs?

How to fix Hardware Bugs?

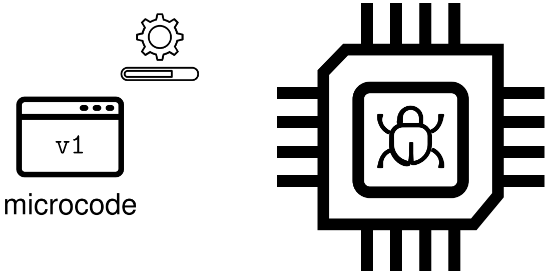
How to fix Hardware Bugs?



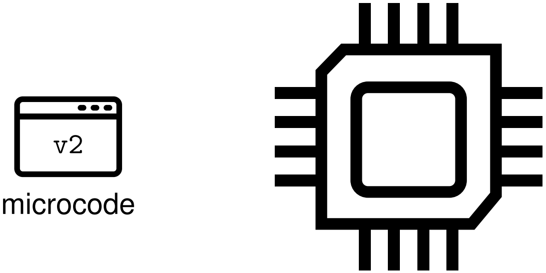
How to fix Hardware Bugs?



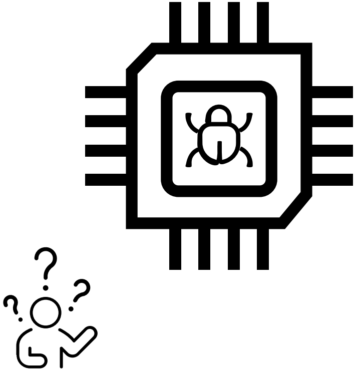
How to fix Hardware Bugs?



How to fix Hardware Bugs?



How to fix Hardware Bugs?



How to fix Hardware Bugs?

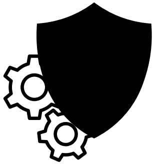
disable vector extension

How to fix Hardware Bugs?

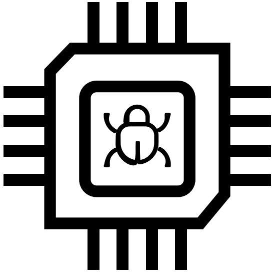
OS

How to fix Hardware Bugs?

disable vector extension



OS



How to fix Hardware Bugs?

OS: disable extension

How to fix Hardware Bugs?

OS: disable extension

up to 33% overhead

lose ∼ 50% instructions

How to fix Hardware Bugs?

OS: disable extension

up to 33% overhead

lose ∼ 50% instructions

How to fix Hardware Bugs?

OS: disable extension

up to 33% overhead

lose ~ 50% instructions

OS: disable extension

How to fix Hardware Bugs?

OS: disable extension

up to 33% overhead

lose ~ 50% instructions

OS: disable extension

up to 77% overhead

lose ~ 50% instructions

How to fix Hardware Bugs?

OS: disable extension

up to 33% overhead

lose ~ 50% instructions

OS: disable extension

up to 77% overhead

lose ~ 50% instructions

How to fix Hardware Bugs?

OS: disable extension

up to 33% overhead

lose ∼ 50% instructions

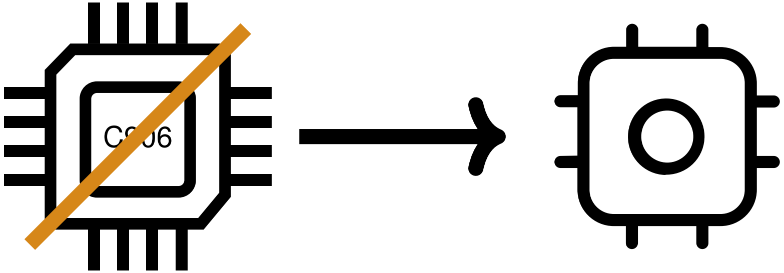
OS: disable extension

up to 77% overhead

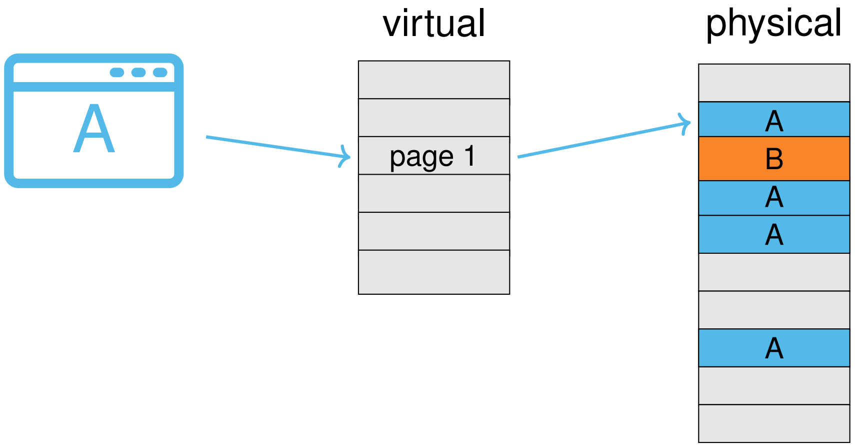
lose ∼ 50% instructions

no mitigation

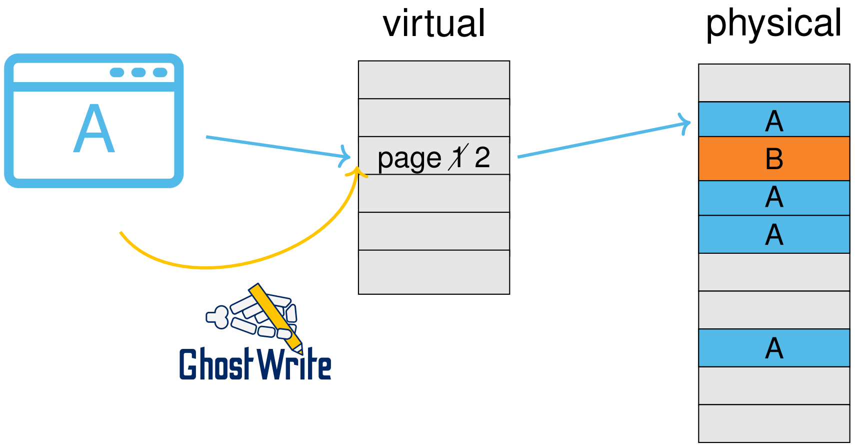
How to the fix C906?



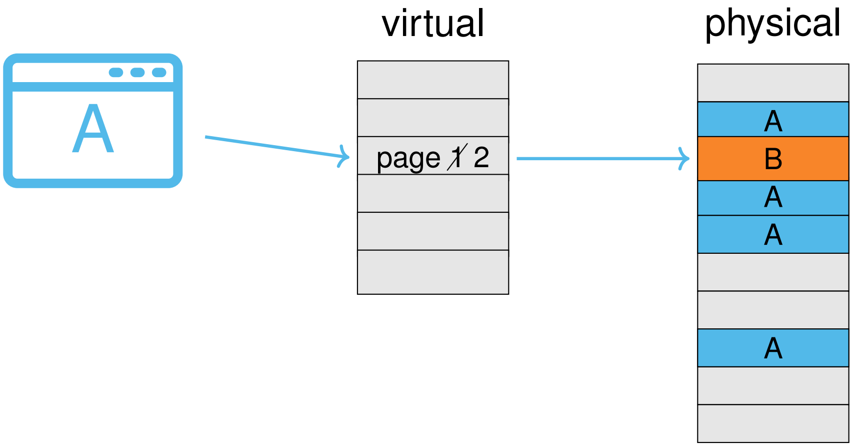
Reading Arbitrary Memory



Reading Arbitrary Memory



Reading Arbitrary Memory



Demo: Reading Arbitrary Memory

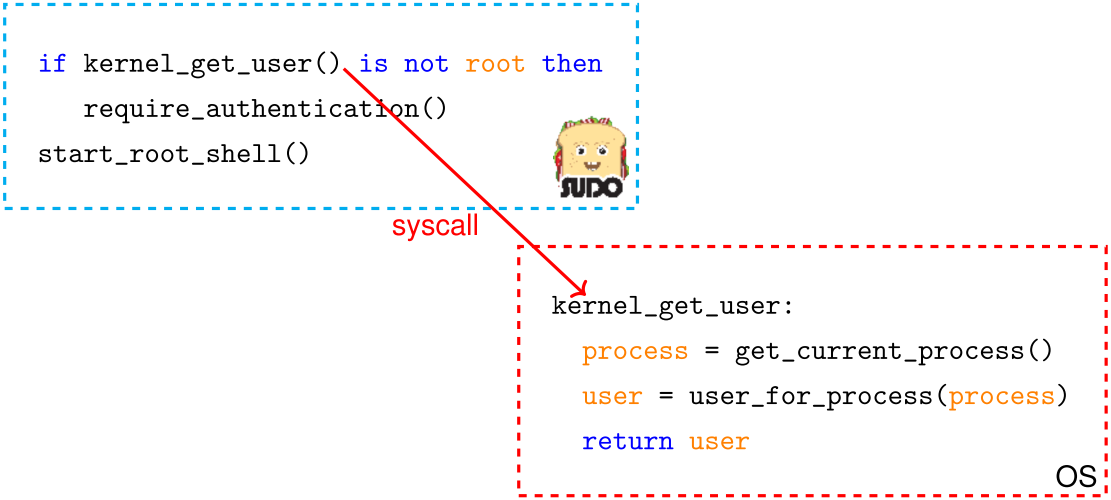
Getting root

```
if kernel_get_user() is not root then
   require_authentication()
start_root_shell()
```

```
kernel_get_user:
   process = get_current_process()
   user = user_for_process(process)
   return user
```

OS

Getting root



```
if kernel_get_user() is not root then
   require_authentication()
start_root_shell()
```

```
kernel_get_user:
   process = get_current_process()
   user = user_for_process(process)
   return user
```

Getting root

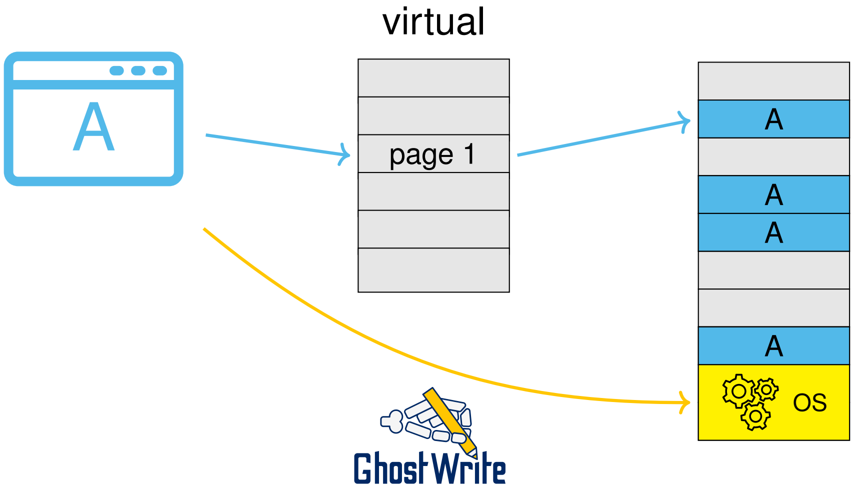
```
if kernel_get_user() is not root then
   require_authentication()
start_root_shell()
```

syscall

```
kernel_get_user:
   process = get_current_process()
   user = user_for_process(process)
   return root
```

OS

Getting root: Patching the Kernel



Demo: Getting root

Is GhostWrite the only bug?

GhostWrite

Is GhostWrite the only bug?

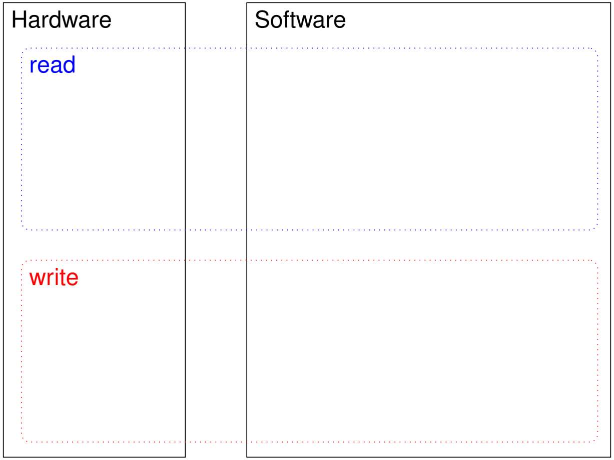
ÆPIC LEAK

CACHE WARP

GhostWrite

MDS

GhostWrite: The Big Picture



GhostWrite: The Big Picture

oftware

write

black hat USA 2017

REGISTER NOW

JULY 22-27, 2017
MANDALAY BAY / LAS VEGAS, NV

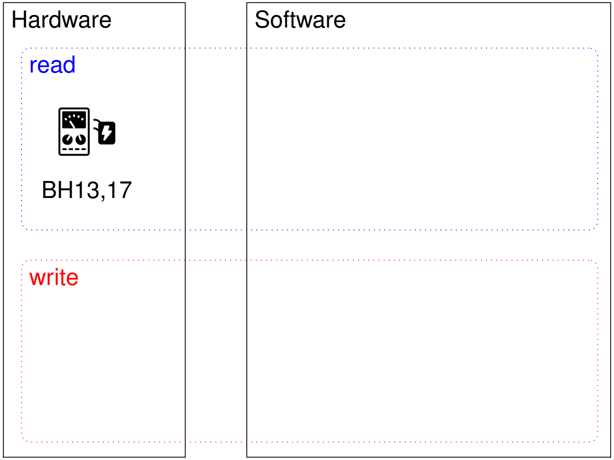
ATTEND | TRAININGS | BRIEFINGS | ARSENAL | FEATURES | SCHEDULE | SPECIAL EVENTS | SPONSORS | PROPOSALS

BACK TO TRAINING

ON THIS PAGE

ADVANCED HARDWARE HACKING: HANDS-ON POWER ANALYSIS & GLITCHING WITH THE CHIPWHISPERER

GhostWrite: The Big Picture



GhostWrite: The Big Picture

Software

write

nccgroup

Implementing Practical Electrical Glitching Attacks

November 2015

VUSec
riscure

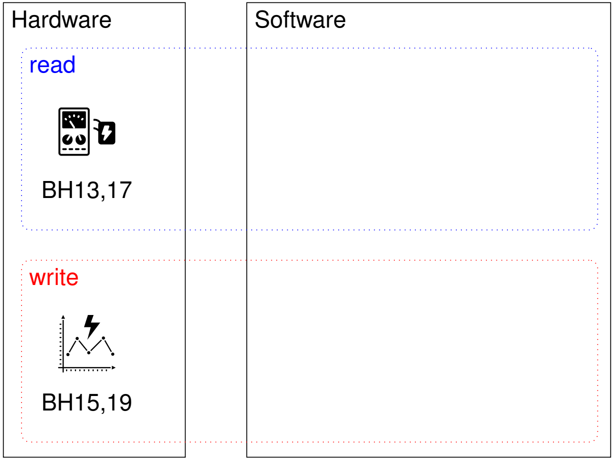
black hat USA 2018

There Will Be Glitches:
Extracting and Analyzing
Automotive Firmware Efficiently

black hat USA 2018

AUGUST 4-9, 2018
MANDALAY BAY / LAS VEGAS

GhostWrite: The Big Picture



GhostWrite: The Big Picture

oftware

Side-Channel Attacks on Everyday Applications

Taylor Hornby

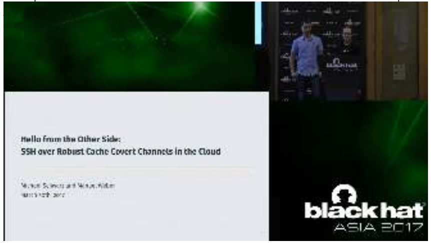
[illegible]

University of Calgary

Zcash

write

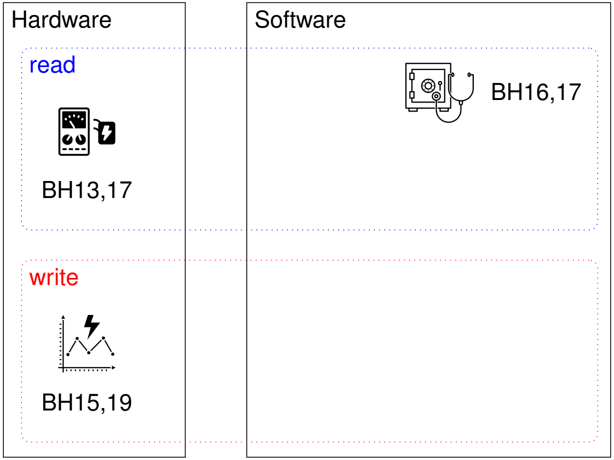
BH15,19



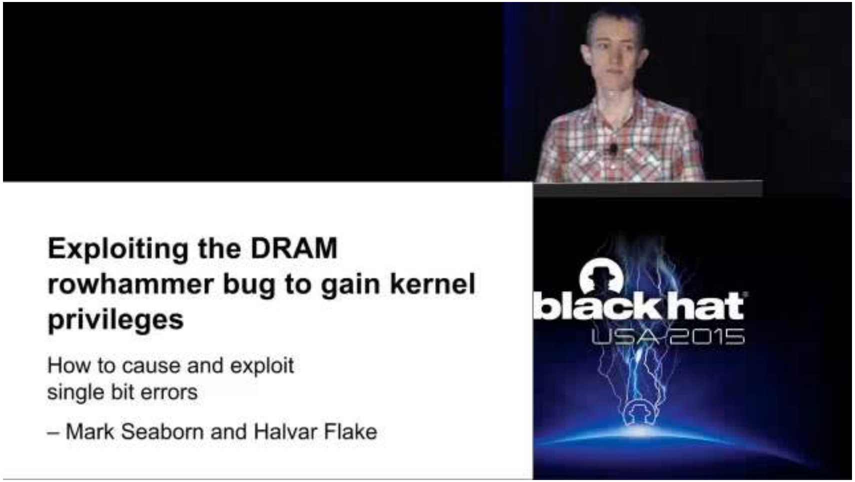
Hello from the Other Side:
SSH over Robust Cache Covert Channels in the Cloud

[illegible]

GhostWrite: The Big Picture



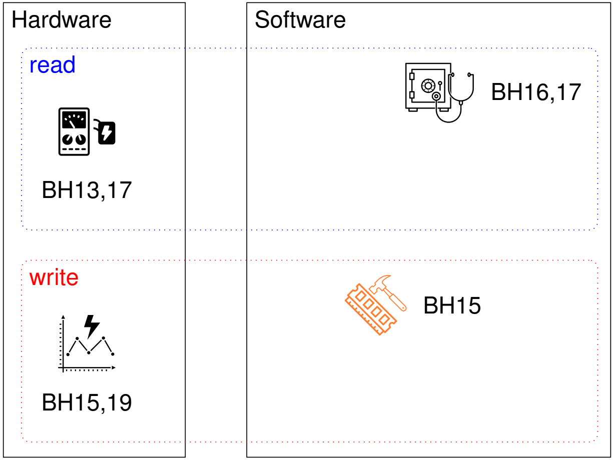
GhostWrite: The Big Picture


Exploiting the DRAM rowhammer bug to gain kernel privileges

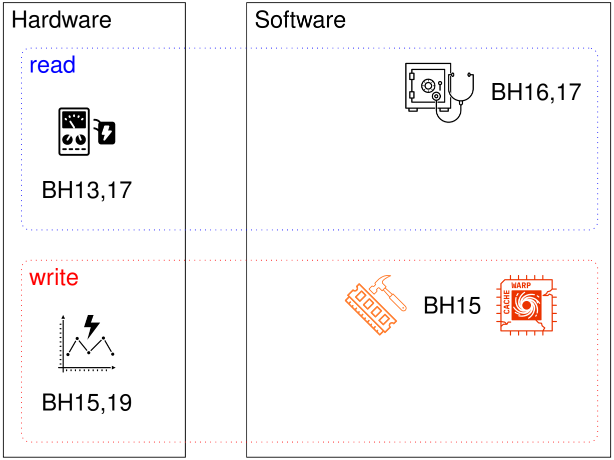
How to cause and exploit single bit errors

– Mark Seaborn and Halvar Flake

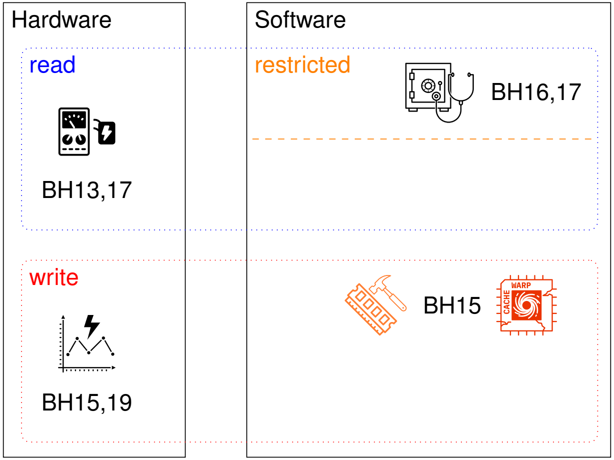
GhostWrite: The Big Picture



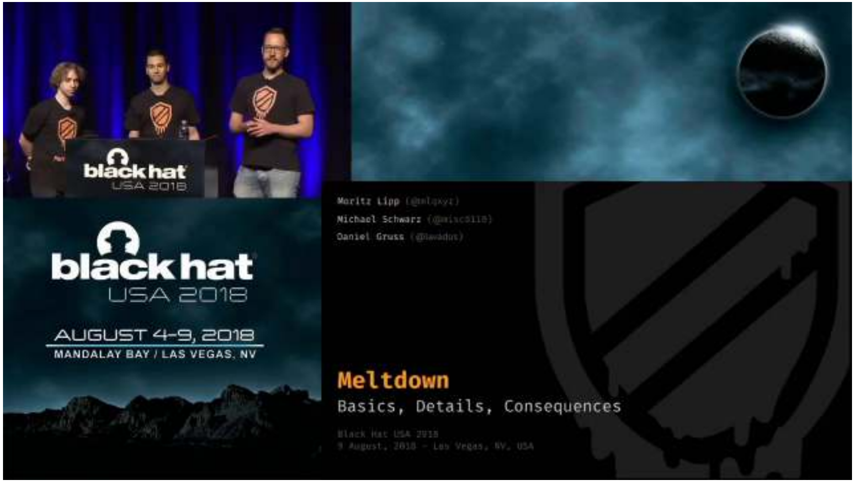
GhostWrite: The Big Picture



GhostWrite: The Big Picture



GhostWrite: The Big Picture



Moritz Lipp (@mlqxyz)
Michael Schwarz (@misc0110)
Daniel Gruss (@lavados)

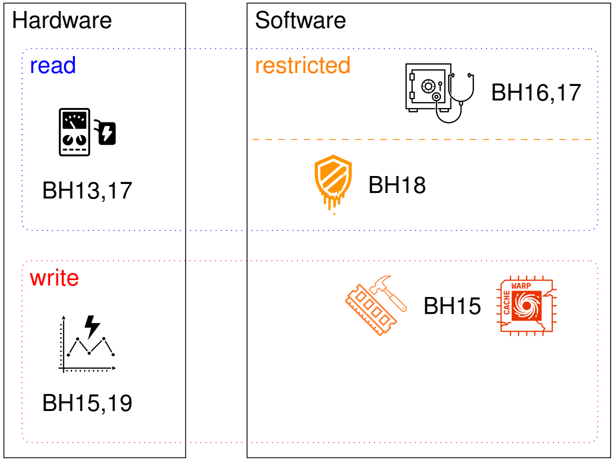
Meltdown

Basics, Details, Consequences

Black Hat USA 2018
9 August, 2018 – Las Vegas, NV, USA

Black Hat USA 2024

GhostWrite: The Big Picture



GhostWrite: The Big Picture

black hat USA 2023

Downfall

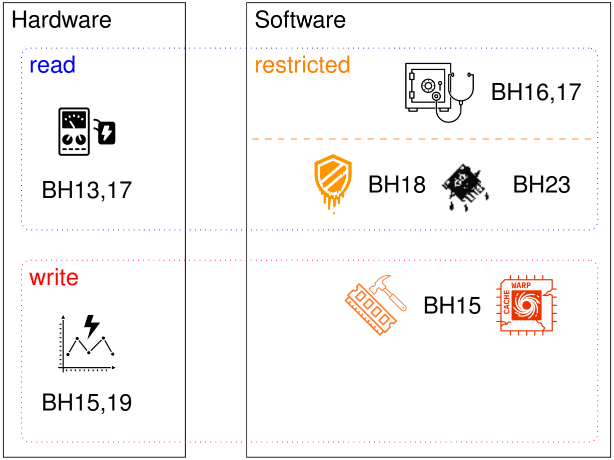
Single Instruction Multiple Data Leaks in Cutting-edge CPUs

Daniel Moghimi – August 9, 2023
BlackHat USA, Las Vegas

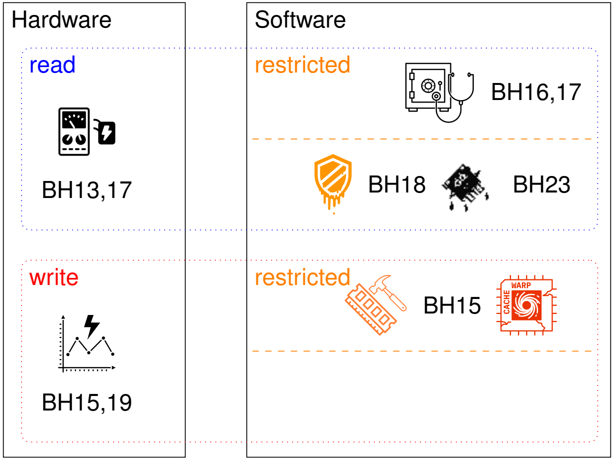
I'm going to talk about downfall, single instruction, multiple data

AUGUST 5-10, 2023
MANDALAY BAY / LAS VEGAS

GhostWrite: The Big Picture

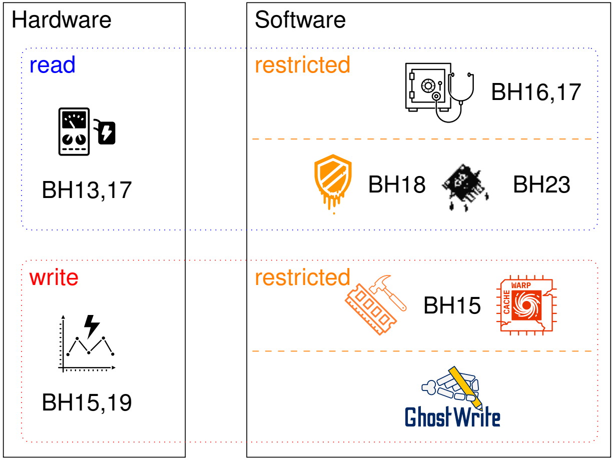


GhostWrite: The Big Picture



black hat USA 2024

GhostWrite: The Big Picture



GhostWrite: Comparison

	Rowhammer	CacheWarp	GhostWrite
Restrictions	*bit flips*		
Speed	😭		
Practicality	🙁		

GhostWrite: Comparison

	Rowhammer	CacheWarp	GhostWrite
Restrictions	*bit flips*	*old state*	
Speed	😭	😃	
Practicality	🙁	😐	

GhostWrite: Comparison

	Rowhammer	CacheWarp	GhostWrite
Restrictions	*bit flips*	*old state*	–
Speed	😭	😃	😃
Practicality	🙁	😐	😃

What can we learn?

RISC-V is great

What can we learn?

RISC-V is great

Only C910

What can we learn?

RISC-V is great

Only C910

Quality control important

What can we learn?

RISC-V is great

Only C910

Quality control important

Configurable hardware

GhostWrite: Overview

ghostwriteattack.com

Microarchitecture Vulnerabilities:

Past, Present, and Future

The End: GhostWrite

- GhostWrite destroys all isolations on C910 RISC-V CPU

@fth0mas @hetterichlorenz

ghostwriteattack.com

The End: GhostWrite

GhostWrite

- GhostWrite destroys all isolations on C910 RISC-V CPU
- Mitigation: disable vector extension, up to 33% overhead

@fth0mas @hetterichlorenz

ghostwriteattack.com

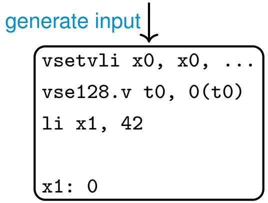
The End: GhostWrite

- GhostWrite destroys all isolations on C910 RISC-V CPU
- Mitigation: disable vector extension, up to 33% overhead
- Hardware bugs are everywhere

@fth0mas @hetterichlorenz

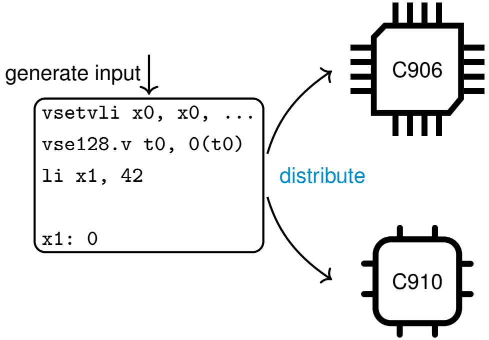
ghostwriteattack.com

GhostWrite: The Framework



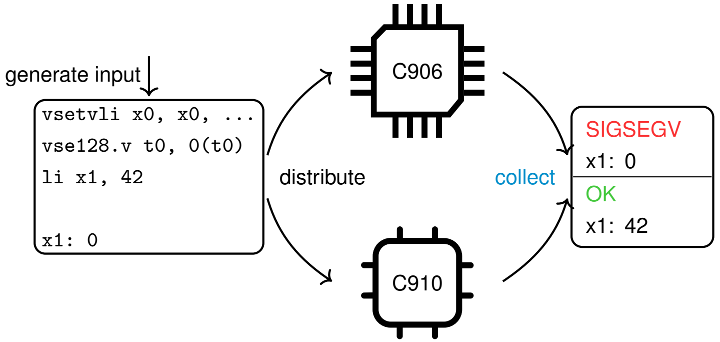
```
vsetvli x0, x0, ...
vse128.v t0, 0(t0)
li x1, 42

x1: 0
```

GhostWrite: The Framework



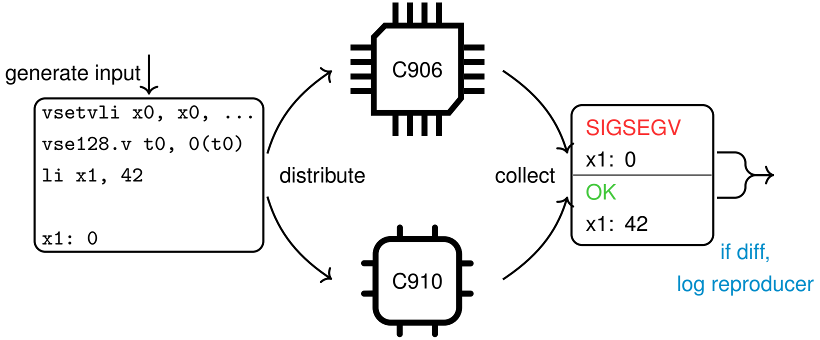
```
vsetvli x0, x0, ...
vse128.v t0, 0(t0)
li x1, 42

x1: 0
```

GhostWrite: The Framework



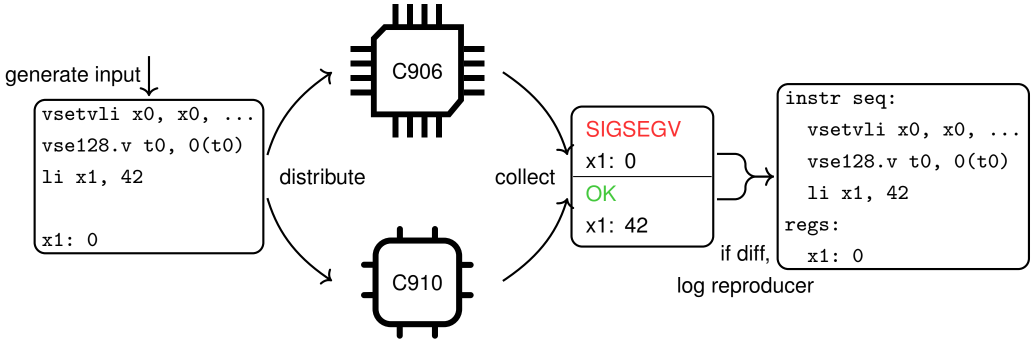
```
vsetvli x0, x0, ...
vse128.v t0, 0(t0)
li x1, 42

x1: 0
```

GhostWrite: The Framework



```
vsetvli x0, x0, ...
vse128.v t0, 0(t0)
li x1, 42

x1: 0
```

GhostWrite: The Framework



```
vsetvli x0, x0, ...
vse128.v t0, 0(t0)
li x1, 42

x1: 0
```

```
instr seq:
  vsetvli x0, x0, ...
  vse128.v t0, 0(t0)
  li x1, 42
regs:
  x1: 0
```