InSpectre Gadget:
Inspecting the Residual Attack Surface of Cross-privilege Spectre v2

***Sander Wiebing**, *Alvise de Faveri Tron,
Herbert Bos, Cristiano Giuffrida

*Equal contribution joint first authors

VU VRIJE UNIVERSITEIT AMSTERDAM

VUSec

In Short

- Defenses rely on mitigating Spectre ‘gadgets’

- For the first time, we **precisely reason about exploitability**
  ➔ New approach to analyze Spectre gadgets

- **New Spectre-V2 Attack** with *native* gadgets: **Native BHI**
  ➔ Leaking kernel memory at 3.5 kB/sec on latest Intel CPUs

Spectre Gadget

```
if (attacker < size) {
    uint64_t secret       = array[ attacker ];
             secret       = secret & 0xFF;
             secret       = secret << 12;
    uint64_t transmission = base[ secret ];
}
```

Spectre Gadget

```
ptr->foo();
void foo(void) {

   uint64_t secret       = array[ attacker ];

            secret       = secret & 0xFF;

            secret       = secret << 12;

   uint64_t transmission = base[ secret ];

}
```

So, why is it important to find these gadgets?

Finding Gadgets: High-Level Data-Flow Approach

```
uint64_t secret       = array[ attacker ];
         secret       = secret & 0xFF;
         secret       = secret << 12;
uint64_t transmission = base[ secret ];
```

BHI results: >1000 *potential* gadgets

➔ **exploitability** highly uncertain

Finding Gadgets: Pattern-Based Approach

```
uint64_t secret       = array[ attacker ];
         secret       = secret & 0xFF;
         secret       = secret << 12;
uint64_t transmission = base[ secret ];
```

Intel engineers results for BHI: **0 exploitable gadgets**

➔ Non-standard gadgets can be exploitable, but other **fine-grained requirements have to hold**

Our Approach: In-Depth Inspection

What is the range?

```
uint64_t secret       = array[ attacker ];
         secret       = secret & 0xFF;
         secret       = secret <<
uint64_t transmission = base[ secret ];
```

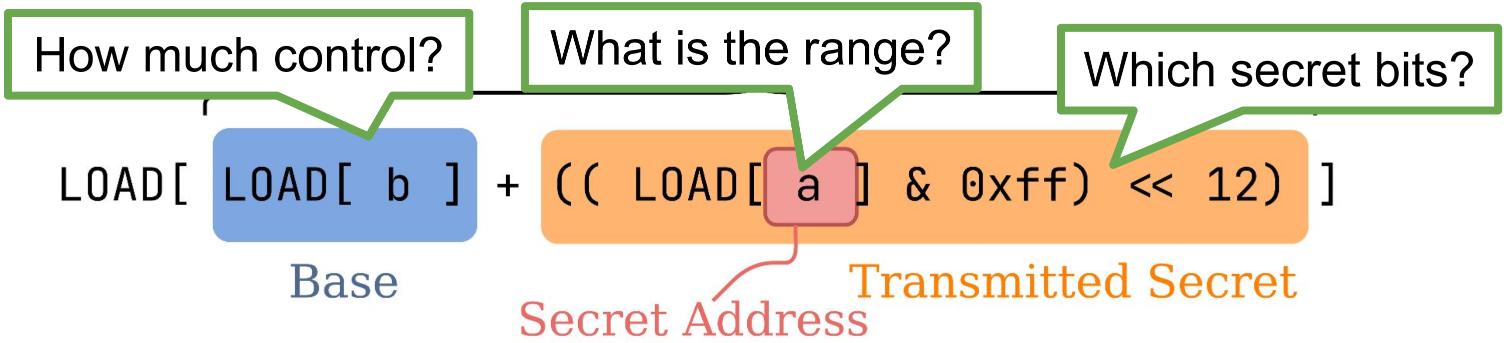
Which secret bits?

How much control?

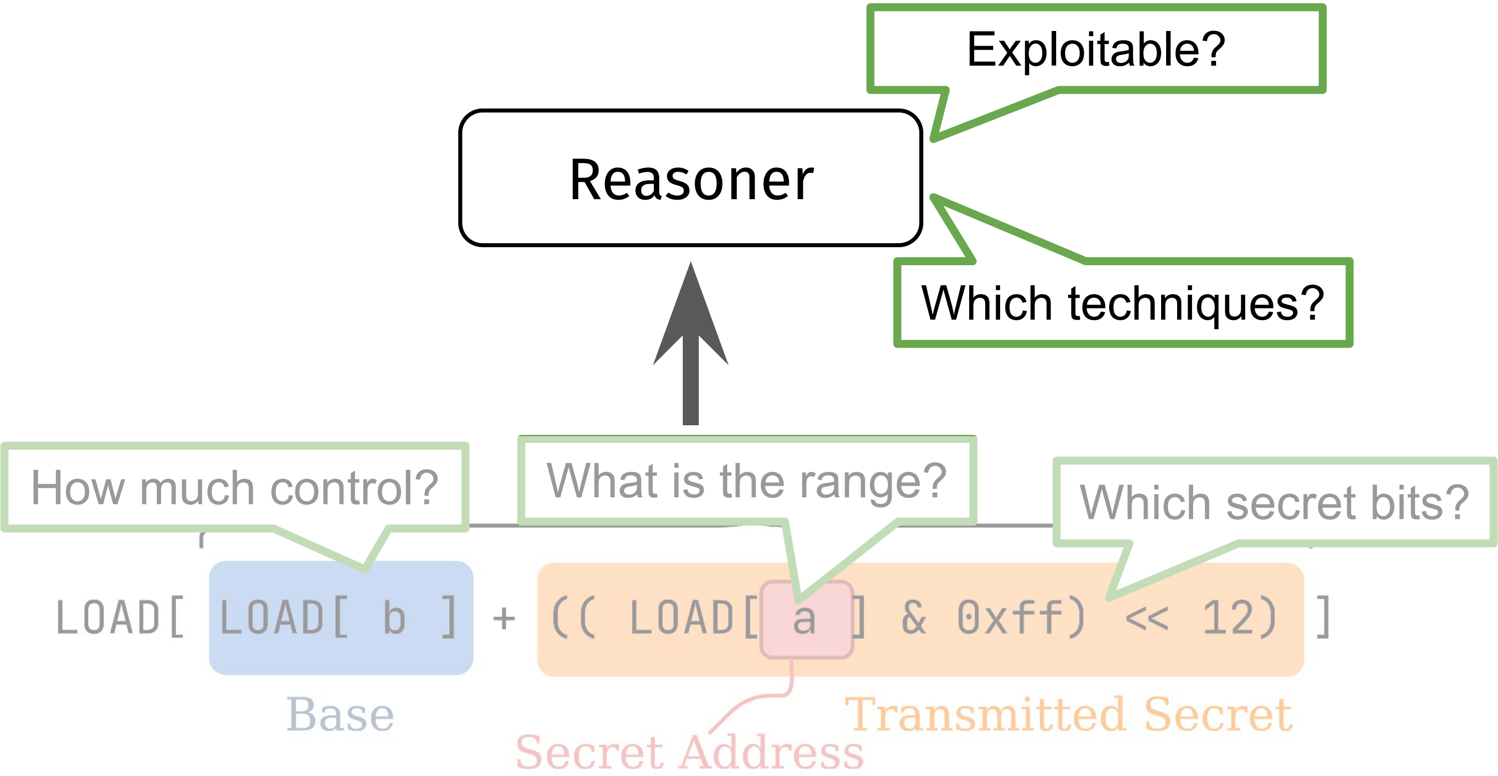
At which position?

Our Approach: In-Depth Inspection

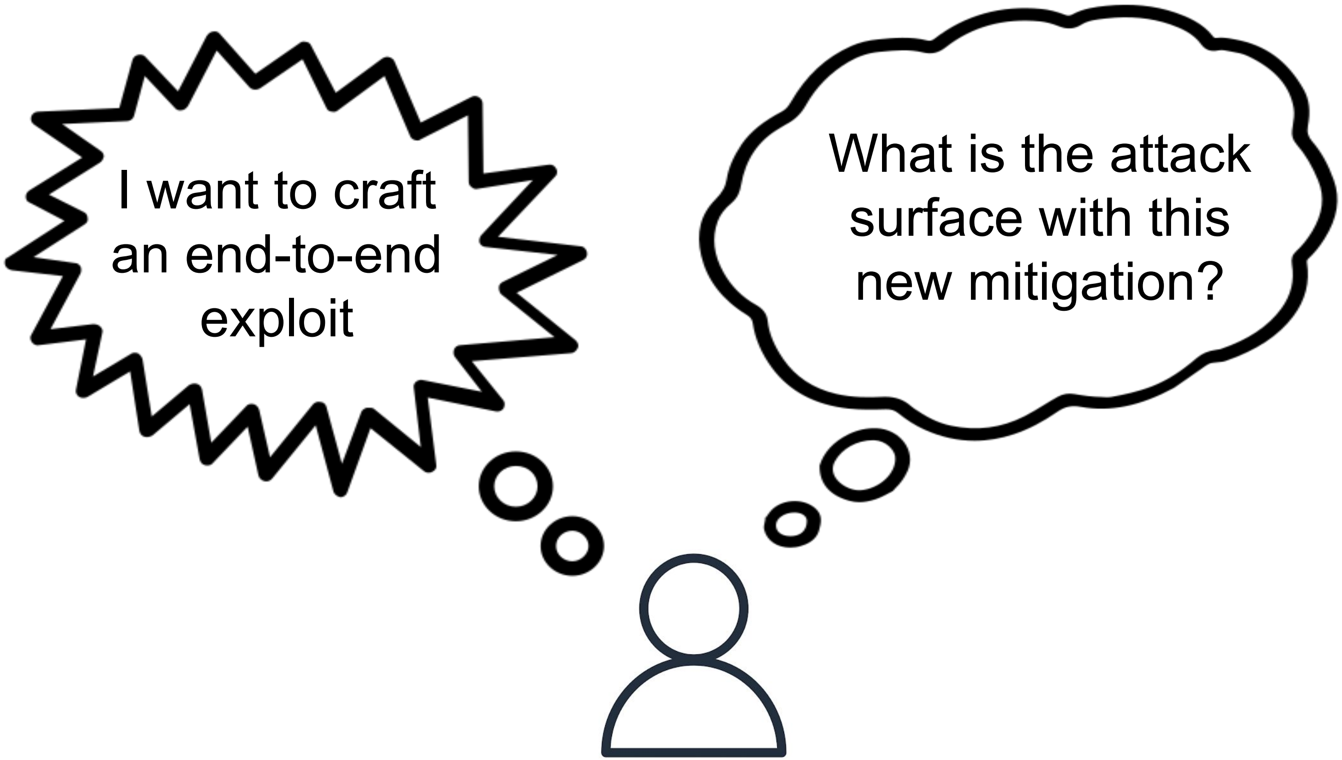
```
uint64_t secret       = array[ attacker ];
         secret       = secret & 0xFF;
         secret       = secret << 12;
uint64_t transmission = base[ secret ];
```



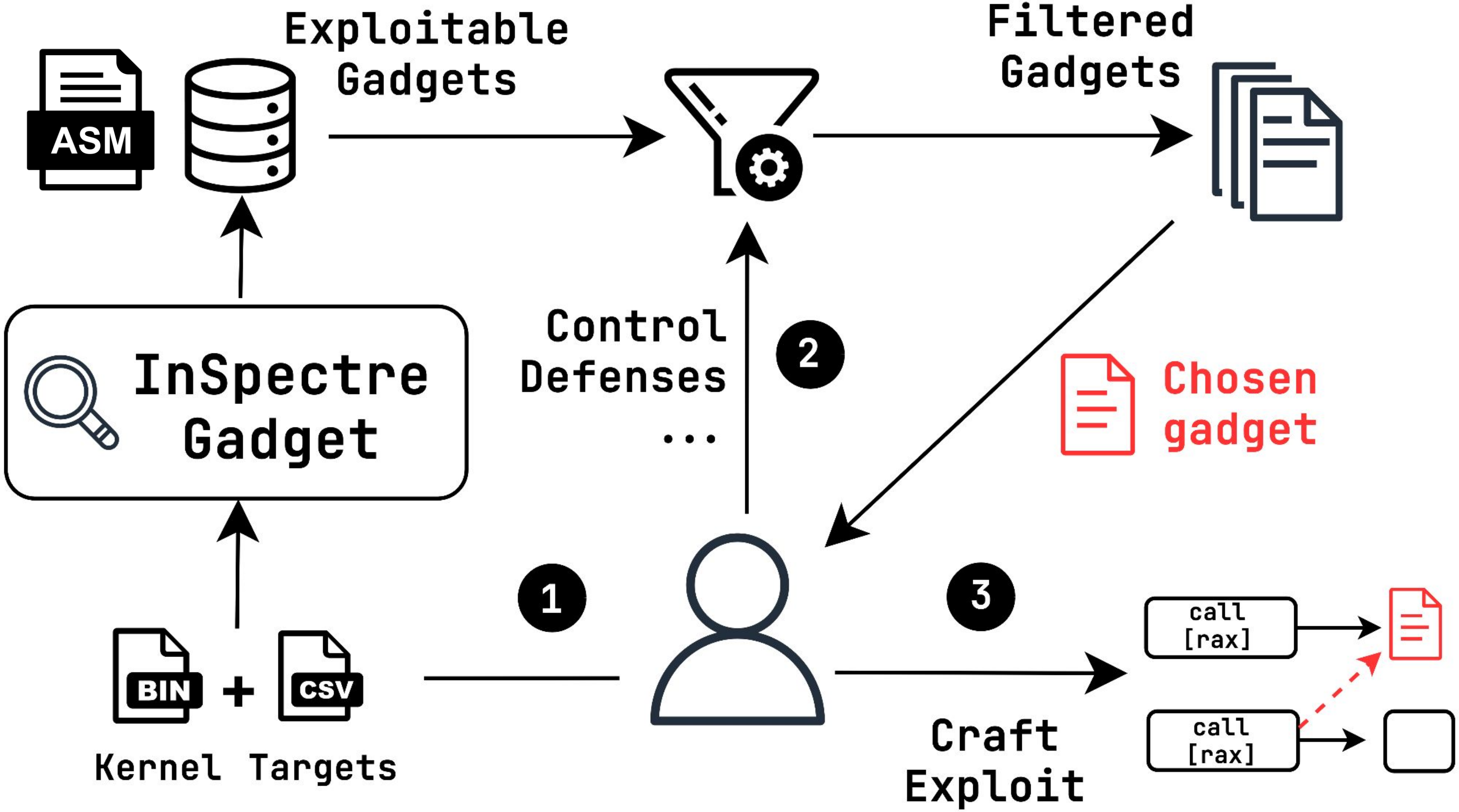
Our Approach: In-Depth Inspection



InSpectre Gadget Workflow



InSpectre Gadget Workflow



Results Analyzing Linux Kernel

- We found >1500 Spectre-V2 gadgets

- **Native BHI end-to-end exploit** on Linux kernel:
  - **Bypassing all deployed mitigations** on latest Intel CPUs
  - Leaking arbitrary kernel memory at 3.5 kB/sec

- **New mitigations** deployed by Intel and Linux Kernel developers
  - Both software and hardware

InSpectre Gadget Demo

```
wiebingsj@vusec:~/inspectre-gadget-results$
```

```
wiebingsj@vusec:~/inspectre-gadget-results$ ls gadgets.db
gadgets.db
wiebingsj@vusec:~/inspectre-gadget-results$
```

```
wiebingsj@vusec:~/inspectre-gadget-results$ ls gadgets.db
gadgets.db
wiebingsj@vusec:~/inspectre-gadget-results$ sqlite3 gadgets.db -cmd ".mode column"
"select * from gadgets
where exploitable = 'True'
LIMIT 10;" | less -S -F
```

```
required_regs    secret_low   secret_high   exploitable   required_solutions                >
---------------  ----------   -----------   -----------   -------------------------------->
['<BV64 rdi>']   6.0          23.0          True          ['perform_training']              >
['<BV64 rdx>']   0.0          15.0          True          ['leak_secret_near_valid_bas>
['<BV64 rdi>']   0.0          4.0           True          ['can_perform_sliding', 'per>
['<BV64 rdi>']   4.0          35.0          True          ['can_perform_known_prefix',>
['<BV64 rdi>']   4.0          24.0          True          []                                >
['<BV64 rdi>']   0.0          63.0          True          ['can_perform_known_prefix',>
['<BV64 rdi>']   3.0          34.0          True          ['can_perform_known_prefix',>
['<BV64 rsi>']   0.0          63.0          True          ['can_perform_known_prefix',>
['<BV64 rsi>']   0.0          7.0           True          ['can_perform_sliding']           >
['<BV64 rdi>']   3.0          18.0          True          []                                >
~
~
~
~
(END)
```

```
ons                                             uuid
----------------------------------------------  ------------------------------------
ing']                                           a2715a53-5f25-4455-91e5-4e932870a37b
ear_valid_base', 'perform_training']            ab7e93db-aab5-4137-bb3b-072f33c0a047
liding', 'perform_training']                    49d5da90-6d90-445d-b37b-48a1dc010cc0
nown_prefix', 'leak_secret_near_valid_base']    3d507c87-a8b3-4b6a-b929-664f10185eaf
                                                d4c05554-87bb-4454-bf63-7a404dd2c66d
nown_prefix', 'perform_training']               49e79ead-4910-4713-8835-1c6f5230b692
nown_prefix', 'perform_training']               5bb996d2-d414-4452-a858-c2d306eedb9a
nown_prefix', 'can_adjust_base']                1c2835c5-da16-4879-b829-cf57c1103111
liding']                                        fa992c2e-4431-4999-b49a-59c766bebe6b
                                                a46b4903-59c7-444c-8ec4-69a8318f17c4
~
~
~
~
(END)
```

```
wiebingsj@vusec:~/inspectre-gadget-results$ inspectre show asm 5bb996d2-d414-4452-a
858-c2d306eedb9a
```

```
wiebingsj@vusec:~/inspectre-gadget-results$ inspectre show asm 5bb996d2-d414-4452-a
858-c2d306eedb9a
```

```
----------------- TRANSMISSION -----------------
                  cgroup_seqfile_show:
8114ff30  endbr64
8114ff34  push    rbp
8114ff35  mov     rax, qword ptr [rdi+0x70] ; {Attacker@rdi} -> {Attacker@0x8114ff35}
8114ff39  mov     r8, rsi
8114ff3c  mov     rbp, rdi
8114ff3f  mov     rax, qword ptr [rax] ; {Attacker@0x8114ff35} -> {Attacker@0x8114ff3f}
8114ff42  mov     rsi, qword ptr [rax+0x60] ; {Attacker@0x8114ff3f} -> {Attacker@0x8114ff42}
8114ff46  mov     rdx, qword ptr [rax+0x8] ; {Attacker@0x8114ff3f} -> {Attacker@0x8114ff46}
8114ff4a  mov     rax, qword ptr [rsi+0x58] ; {Attacker@0x8114ff42} -> {Attacker@0x8114ff4a}
8114ff4e  mov     rdi, qword ptr [rdx+0x60] ; {Attacker@0x8114ff46} -> {Attacker@0x8114ff4e}
8114ff52  test    rax, rax
8114ff55  je      0x8114ff67 ; Not Taken   <Bool LOAD_64[ LOAD_64[ LOAD_64[ LOAD_64[ rdi + 0x70 ]]
                                              + 0x60] + 0x58] != 0x0>
8114ff57  movsxd  rax, dword ptr [rax+0x9c] ; {Attacker@0x8114ff4a} -> {Secret@0x8114ff57}
8114ff62  mov     rdi, qword ptr [rdi+rax*0x8+0x8] ; {Attacker@0x8114ff4e, Secret@0x8114ff57} -> TRANSMISSION
8114ff67  mov     rax, qword ptr [rsi+0x98]
8114ff6e  test    rax, rax
8114ff71  je      0x8114ff7f

------------------------------------------------
uuid: 5bb996d2-d414-4452-a858-c2d306eedb9a
```

```
wiebingsj@vusec:~/inspectre-gadget-results$ inspectre show asm 5bb996d2-d414-4452-a
858-c2d306eedb9a
```

```
----------------- TRANSMISSION -----------------
                  cgroup_seqfile_show:
8114ff30  endbr64
8114ff34  push    rbp
8114ff35  mov     rax, qword ptr [rdi+0x70] ; {Attacker@rdi} -> {Attacker@0x8114ff35}
8114ff39  mov     r8, rsi
8114ff3c  mov     rbp, rdi
8114ff3f  mov     rax, qword ptr [rax] ; {Attacker@0x8114ff35} -> {Attacker@0x8114ff3f}
8114ff42  mov     rsi, qword ptr [rax+0x60] ; {Attacker@0x8114ff3f} -> {Attacker@0x8114ff42}
8
8114ff4a  mov     rax, qword ptr [rsi+0x58] ; {Attacker@0x8114ff42} -> {Attacker@0x8114ff4a}
8114ff4e  mov     rdi, qword ptr [rdx+0x60] ; {Attacker@0x8114ff46} -> {Attacker@0x8114ff4e}
8114ff52  test    rax, rax
8114ff55  je      0x8114ff67 ; Not Taken   <Bool LOAD_64[ LOAD_64[ LOAD_64[ LOAD_64[ rdi + 0x70 ]]
                                           + 0x60] + 0x58] != 0x0>
8114ff57  movsxd  rax, dword ptr [rax+0x9c] ; {Attacker@0x8114ff4a} -> {Secret@0x8114ff57}
8114ff62  mov     rdi, qword ptr [rdi+rax*0x8+0x8] ; {Attacker@0x8114ff4e, Secret@0x8114ff57} -> TRANSMISSION
8114ff67  mov     rax, qword ptr [rsi+0x98]
8114ff6e  test    rax, rax
8114ff71  je      0x8114ff7f

------------------------------------------------
uuid: 5bb996d2-d414-4452-a858-c2d306eedb9a
```

```
mov     rax, qword ptr [rdi+0x70] ; {Attacker@rdi} -> {Attacker@0x8114ff35}
```

```
wiebingsj@vusec:~/inspectre-gadget-results$ inspectre show asm 5bb996d2-d414-4452-a
858-c2d306eedb9a
```

```
----------------- TRANSMISSION -----------------
                  cgroup_seqfile_show:
8114ff30  endbr64
8114ff34  push    rbp
8114ff35  mov     rax, qword ptr [rdi+0x70] ; {Attacker@rdi} -> {Attacker@0x8114ff35}
8114ff39  mov     r8, rsi
8114ff3c  mov     rbp, rdi
8114ff3f  mov     rax, qword ptr [rax] ; {Attacker@0x8114ff35} -> {Attacker@0x8114ff3f}
8114ff42  mov     rsi, qword ptr [rax+0x60] ; {Attacker@0x8114ff3f} -> {Attacker@0x8114ff42}
8114ff46  mov     rdx, qword ptr [rax+0x8] ; {Attacker@0x8114ff3f} -> {Attacker@0x8114ff46}
8114ff4a  mov     rax, qword ptr [rsi+0x58] ; {Attacker@0x8114ff42} -> {Attacker@0x8114ff4a}
8
8
8114ff55  je      0x8114ff67 ; Not Taken   <Bool LOAD_64[ LOAD_64[ LOAD_64[ LOAD_64[ rdi + 0x70 ]]
                                             + 0x60] + 0x58] != 0x0>
8114ff57  movsxd  rax, dword ptr [rax+0x9c] ; {Attacker@0x8114ff4a} -> {Secret@0x8114ff57}
8114ff62  mov     rdi, qword ptr [rdi+rax*0x8+0x8] ; {Attacker@0x8114ff4e, Secret@0x8114ff57} -> TRANSMISSION
8114ff67  mov     rax, qword ptr [rsi+0x98]
8114ff6e  test    rax, rax
8114ff71  je      0x8114ff7f

------------------------------------------------
uuid: 5bb996d2-d414-4452-a858-c2d306eedb9a
```


movsxd rax, dword ptr [rax+0x9c] ; {Attacker@0x8114ff4a} -> {Secret@0x8114ff57}

```
wiebingsj@vusec:~/inspectre-gadget-results$ inspectre show asm 5bb996d2-d414-4452-a
858-c2d306eedb9a
```

```
----------------- TRANSMISSION -----------------
                 cgroup_seqfile_show:
8114ff30  endbr64
8114ff34  push    rbp
8114ff35  mov     rax, qword ptr [rdi+0x70] ; {Attacker@rdi} -> {Attacker@0x8114ff35}
8114ff39  mov     r8, rsi
8114ff3c  mov     rbp, rdi
8114ff3f  mov     rax, qword ptr [rax] ; {Attacker@0x8114ff35} -> {Attacker@0x8114ff3f}
8114ff42  mov     rsi, qword ptr [rax+0x60] ; {Attacker@0x8114ff3f} -> {Attacker@0x8114ff42}
8114ff46  mov     rdx, qword ptr [rax+0x8] ; {Attacker@0x8114ff3f} -> {Attacker@0x8114ff46}
8114ff4a  mov     rax, qword ptr [rsi+0x58] ; {Attacker@0x8114ff42} -> {Attacker@0x8114ff4a}
8114ff4e  mov     rdi, qword ptr [rdx+0x60] ; {Attacker@0x8114ff46} -> {Attacker@0x8114ff4e}
```

qword ptr [rdi+rax*0x8+0x8] ; {Attacker@0x8114ff4e, Secret@0x8114ff57} -> TRANSMISSION

```
                                             + 0x60] + 0x58] != 0x0>
8114ff57  movsxd  rax, dword ptr [rax+0x9c] ; {Attacker@0x8114ff4a} -> {Secret@0x8114ff57}
8114ff62  mov     rdi, qword ptr [rdi+rax*0x8+0x8] ; {Attacker@0x8114ff4e, Secret@0x8114ff57} -> TRANSMISSION
8114ff67  mov     rax, qword ptr [rsi+0x98]
8114ff6e  test    rax, rax
8114ff71  je      0x8114ff7f

------------------------------------------------
uuid: 5bb996d2-d414-4452-a858-c2d306eedb9a
```

```
wiebingsj@vusec:~/inspectre-gadget-results$ inspectre show asm 5bb996d2-d414-4452-a858-c2d306eedb9a
8114ff57  movsxd  rax, dword ptr [rax+0x9c] ; {Attacker@0x8114ff4a} -> {Secret@0x8114ff57}
8114ff62  mov     rdi, qword ptr [rdi+rax*0x8+0x8] ; {Attacker@0x8114ff4e, Secret@0x8114ff57} -> TRANSMISSION
8114ff67  mov     rax, qword ptr [rsi+0x98]
8114ff6e  test    rax, rax
8114ff71  je      0x8114ff7f

-------------------------------------------------
uuid: 5bb996d2-d414-4452-a858-c2d306eedb9a
transmitter: TransmitterType.LOAD
Secret Address:
  0x70]] + 0x60] + 0x58] + 0x9c
T
  - Expr:  (0#32 .. LOAD_32[ LOAD_64[ LOAD_64[ LOAD_64[ LOAD_64[ rdi + 0x70]] + 0x60] + 0x58] + 0x9c]) << 0x3
  - Range: (0x0,0x3fffffff8, 0x8) Exact: True
  - Spread: 3 - 34
  - Number of Bits Inferable: 32
Base:
  - Expr:  LOAD_64[ LOAD_64[ LOAD_64[ LOAD_64[ rdi + 0x70]] + 0x8] + 0x60] + 0x178
  - Range: (0x0,0x, 0x1) Exact: True
  - Independent Expr:  LOAD_64[ LOAD_64[ LOAD_64[ LOAD_64[ rdi + 0x70]] + 0x8] + 0x60] + 0x178
  - Independent Range: (0x0,0x, 0x1) Exact: True
Transmission:
  - Expr:  0x8 + LOAD_64[ LOAD_64[ LOAD_64[ LOAD_64[ rdi + 0x70]] + 0x8] + 0x60] + (0x170 + ((0#32 ..
    LOAD_32[ LOAD_64[ LOAD_64[ LOAD_64[ LOAD_64[ rdi + 0x70]] + 0x60] + 0x58] + 0x9c]) << 0x3))
```

- Spread: 3 - 34
- Number of Bits Inferable: 32

Native BHI Demo: Leaking the /etc/shadow File

1 Break KASLR

2 Setup BHI

3 Scan for ‘`root:`’

4 Leak hash

```
Model name: 13th Gen Intel(R) Core(TM) i9-13900K
Linux version: 6.6.0-rc4
Mitigation: Enhanced / Automatic IBRS, IBPB: conditional, RSB filling, PBRSB-eIB
RS: SW sequence
ubuntu@pizza:~/demo/native-bhi$
```

Conclusion

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE
ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL
ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

- **InSpectre Gadget**: in-depth inspection of Spectre gadgets
  - Using knowledge of advanced exploitation techniques
- **Native BHI**: Leaking kernel memory on latest Intel CPUs
- Paper & code available: vusec.net/projects/native-bhi

VU VRIJE UNIVERSITEIT AMSTERDAM

VUSec