hexhive

WarpAttack: Bypassing CFI through Compiler-Introduced Double-Fetches

Jianhao Xu[*†], Luca Di Bartolomeo[†], Flavio Toffalini[†], Bing Mao[*] and Mathias Payer[†]

jianhao xu@smail.nju.edu.cn,
{luca.dibartolomeo, flavio.toffalini}@epfl.ch,
maobing@nju.edu.cn, mathias.payer@nebelwelt.net

*

†

Compiler-Introduced Double Fetch

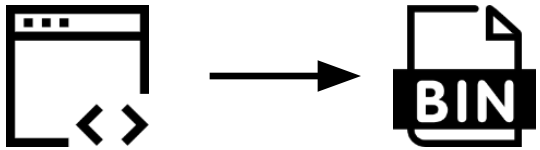
A Linux kernel case (2012):

```
/* commit: 8135cf8b092723dbfcc611fe6fdcb3a36c9951c5 */
switch (op->cmd) {
    case XE
        op-
        bre
    case XE
        //
    defaul
        op-
}
```

```
/* the corresponding assembly code */
cmp DWORD PTR [r13+0x4],0x5
mov DWORD PTR [rbp-0x4c],eax
ja 0x3358 <xen_pcibk_do_op+952>
mov eax,DWORD PTR [r13+0x4]
jmp QWORD PTR [rax*8+off_77D0]
```

Compiler & Correctness-Security Gap

- Compiler



- Correctness-Security Gap

```
// Attempt to scrub the sensitive data saved on stack
memset(secret, 0, sizeof(secret));
return;
```

Correctness ≠ Security

Dead Store

Sensitive Data Scrubbing

Compiler-Introduced Security Issues

- WarpAttack key insight

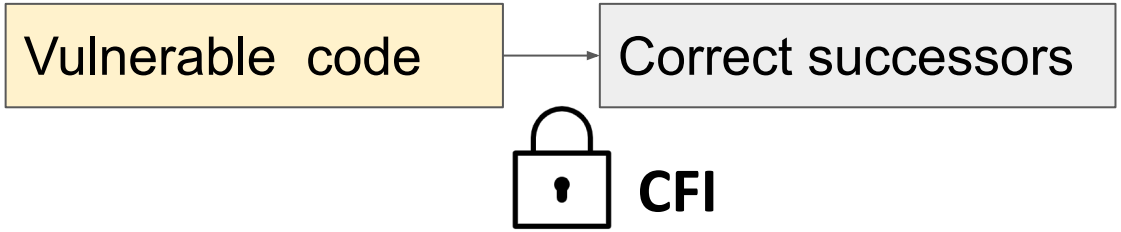
	Compiler Correctness	Security
Compiler-Introduced Double-Fetches	Concurrency bugs Or Benign data race	A weakness of control flow guards

WarpAttack exploits a misalignment between compiler implementations and CFI assumptions

Control Flow Integrity (CFI)

CFI guard control flow

- Inserts run-time checks
- Practical and (reasonably) fine-grained



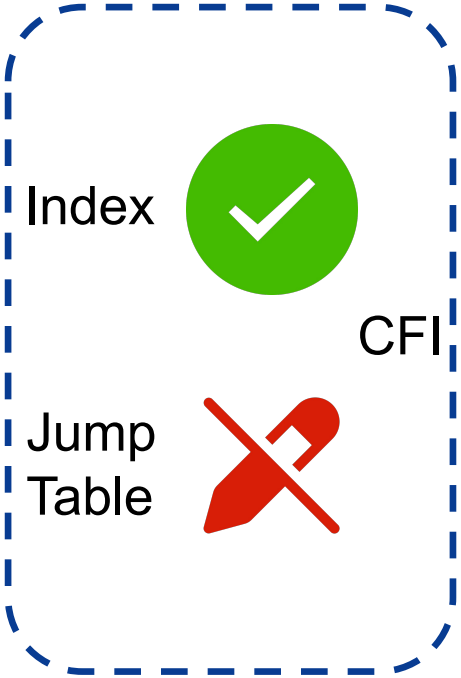
CFI is getting more and more important

Bound-Checked Indirect Jumps

CFI: no need to protect

```
switch(a)
 case 0:
 case 1:
 …
 default:
 …
}
```

```
mov r1, [A]
cmp r1, imm
…
mov r1, [r3+r1*4]
add r1, r3
jmp r1
```

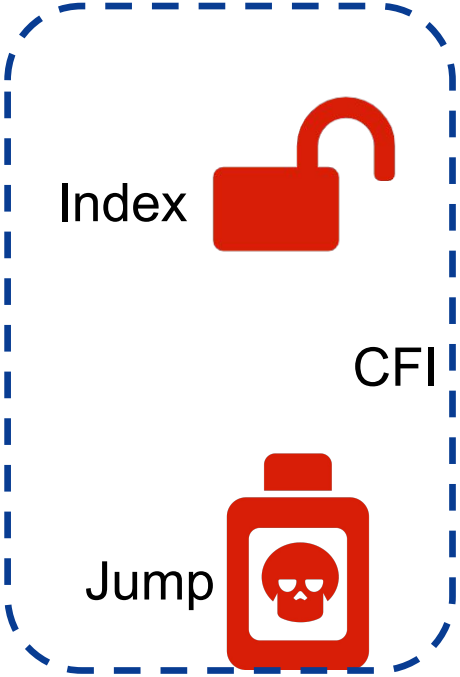


~~Bound-Checked~~ Indirect Jumps

Compilers are not aware of security boundaries

```
mov r1, [A]
cmp r1, imm
…
mov r1, [r3+r1*4]
add r1, r3
jmp r1
```

```
mov r1, [A]
cmp r1, imm
…
mov r2, [A]
mov r2, [r3+r2*4]
add r2, r3
jmp r2
```



WarpAttack: Threat model

Adversarial Capabilities

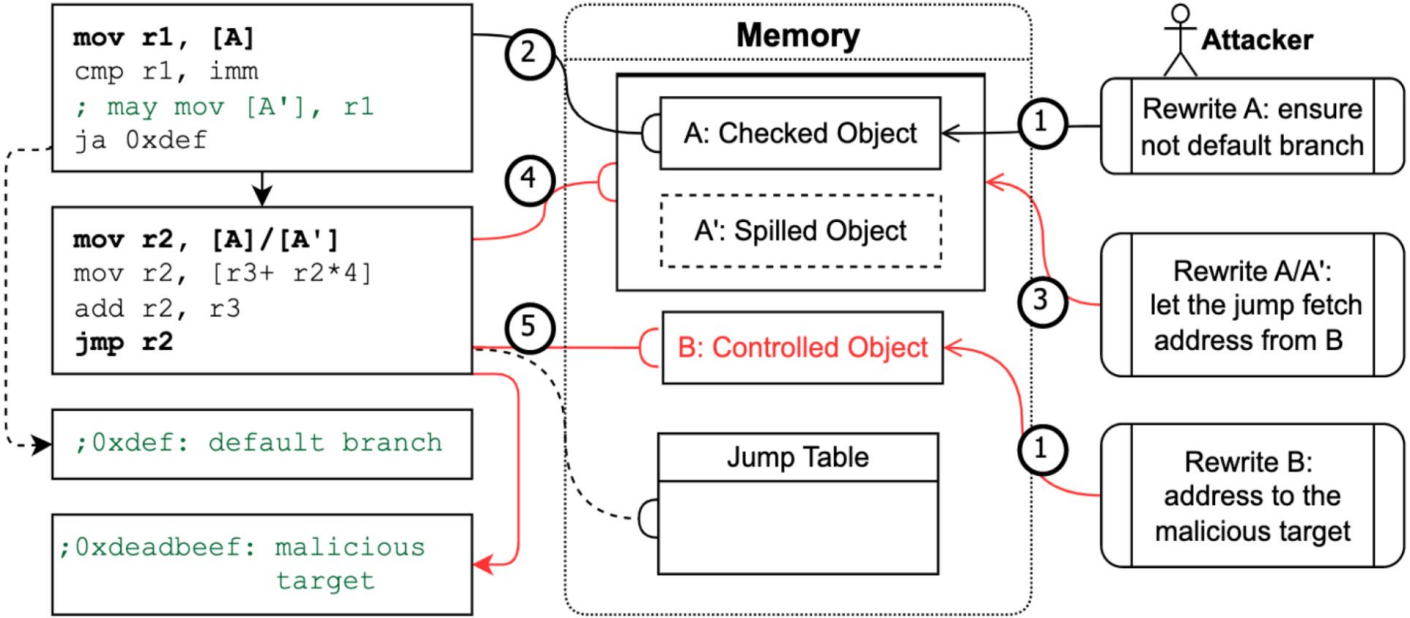
- Arbitrary read-write
- Thread control
- **One triggerable gadget sample**

Defensive Assumptions

- Non-Executable Memory
- Randomization
- Control Flow Protection

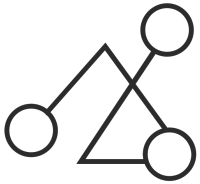
The only requirement beyond CFI's Threat model.

WarpAttack


mov r1, [A]
cmp r1, imm
; may mov [A'], r1
ja 0xdef
mov r2, [A]/[A']
mov r2, [r3+ r2*4]
add r2, r3
jmp r2
;0xdef: default branch
;0xdeadbeef: malicious target
Memory
A: Checked Object
A': Spilled Object
B: Controlled Object
Jump Table
Attacker
Rewrite A: ensure not default branch
Rewrite A/A': let the jump fetch address from B
Rewrite B: address to the malicious target
1
2
3
4
5
1

Challenges and Solutions

Gadget Code Detection:

Challenges	Solutions
Compiler-Introduced? 	
Data Dependency 	 

Challenges and Solutions

Proof-of-Concept Exploit:

Challenges	Solutions
Short time window	
Crashes when wrong	

0.45% success rate in 20 seconds

On Firefox 106.0.1

How WarpAttack affects real world

Vulnerable code in the wild

- All C/C++ programs potentially affected
- 1,600+ victim gadgets in 6 programs

Acknowledgements from




Affected Compilers

- GCC and clang

Other than X86/64

- ARM 32/64*
- RISCV 32/64
- MIPS 32/64

MOSEC2019
Brandon Azad

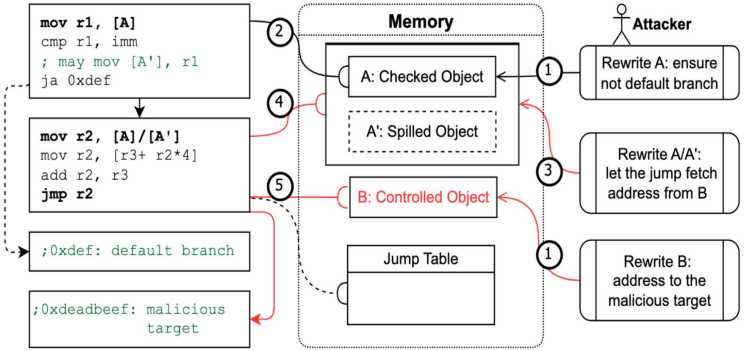
Only X86/64 has two variants,

Others have just stack spilling variant

WarpAttack affects may programs, compilers and architectures

WarpAttack Conclusion

- CFI assumption
- Attack method
- Real world Impact
- Proof-of-Concept



Thanks!

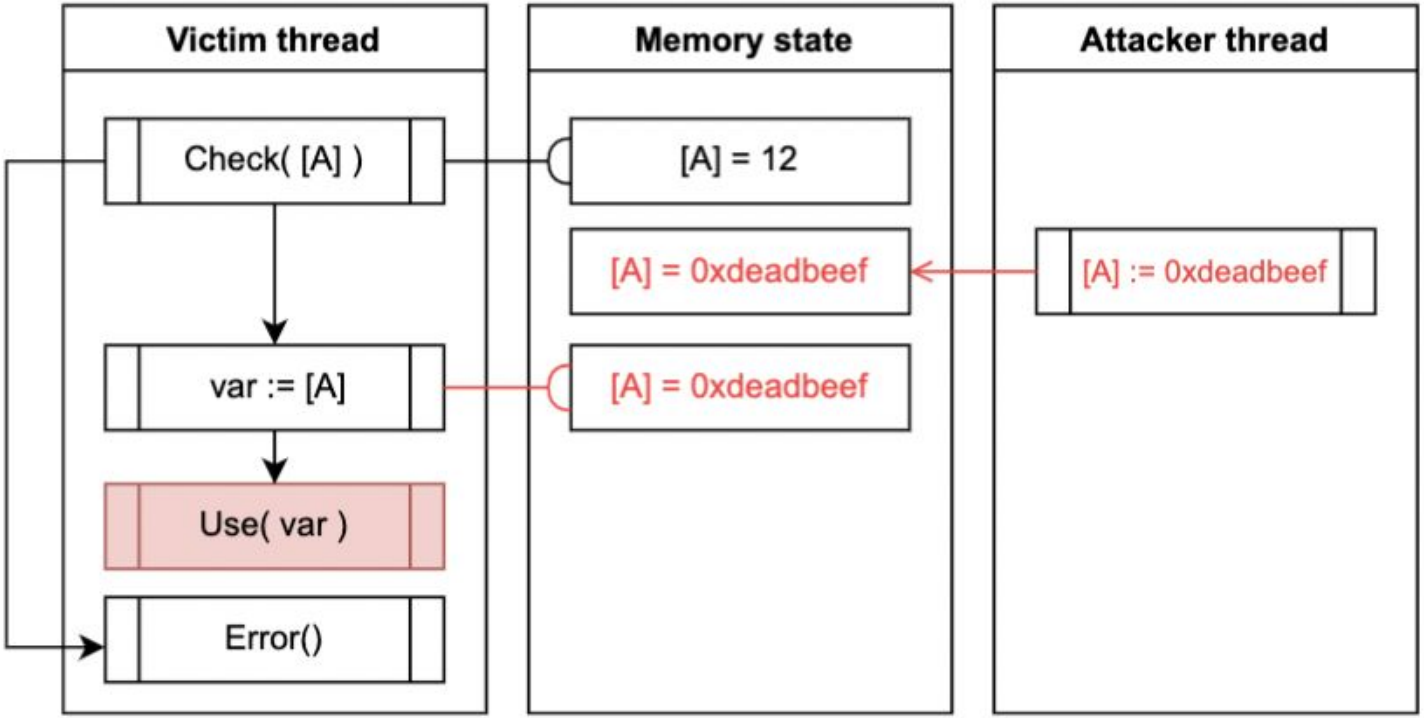
NANJING UNIVERSITY 1902

EPFL

hexhive

- Backup slides

Double Fetch



Mitigations

Avoiding Gadget code generation

- GCC ‘-fno-switch-tables’
- Clang ‘O1’

Protecting Indirect Jump

- CFI checks for switch jump tables

Monitoring for Attack Behavior

- Characteristics like spawning several threads, constantly writing a certain memory site
- Crashes