Software Security 1

Flipped Classroom

> 13.11.2024

Agenda

- More ROP Techniques:
  - SROP
  - Stack Pivoting
  - One Gadget
- Defeating the Shadowstack:
  - JOP & COP

Sigreturn-Oriented Programming (SROP)

Sigreturn-Oriented Programming (SROP)

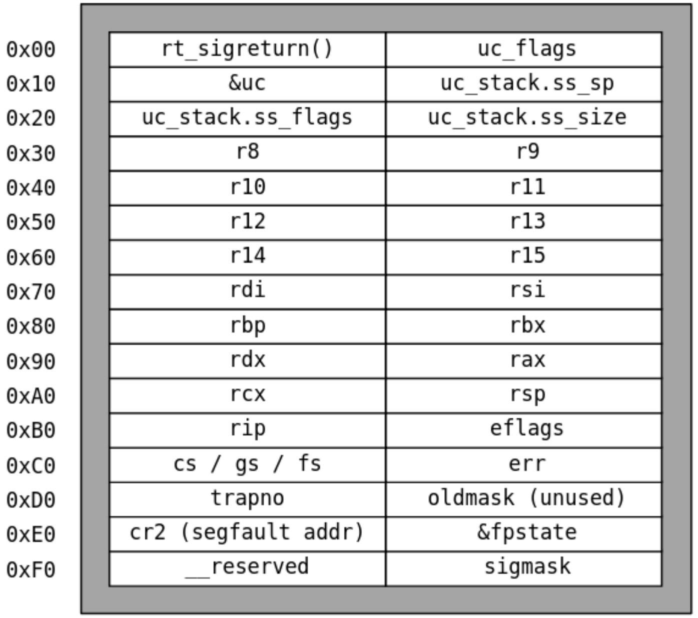
- Unix-based systems have mechanisms for Inter-process communication (IPC) called **signals** that can interrupt program execution. These signals are handled by signal handlers.
- To keep track of the state before interruption, the system stores all current state or context (registers, flags, etc.) **on the stack** and later retrieves the data back to the original containers from the snapshot before the signal was handled.

Sigreturn-Oriented Programming (SROP)

- The `syscall` `rt_sigreturn` or just Sigreturn, is responsible for restoring this snapshot back to the actual data.
- What if a malicious agent that controls the stack and RIP uses this syscall to manipulate the system context state?
- This is what is called Sigreturn-Oriented Programming.

Sigreturn-Oriented Programming (SROP)

- Sigreturn-Oriented Programming is a complementary technique for code-reuse attacks like ROP.
- SROP is useful when you don't have many gadgets to control the registers you want.



0x00	rt_sigreturn()	uc_flags
0x10	&uc	uc_stack.ss_sp
0x20	uc_stack.ss_flags	uc_stack.ss_size
0x30	r8	r9
0x40	r10	r11
0x50	r12	r13
0x60	r14	r15
0x70	rdi	rsi
0x80	rbp	rbx
0x90	rdx	rax
0xA0	rcx	rsp
0xB0	rip	eflags
0xC0	cs / gs / fs	err
0xD0	trapno	oldmask (unused)
0xE0	cr2 (segfault addr)	&fpstate
0xF0	__reserved	sigmask

Stack Pivoting

Stack Pivoting

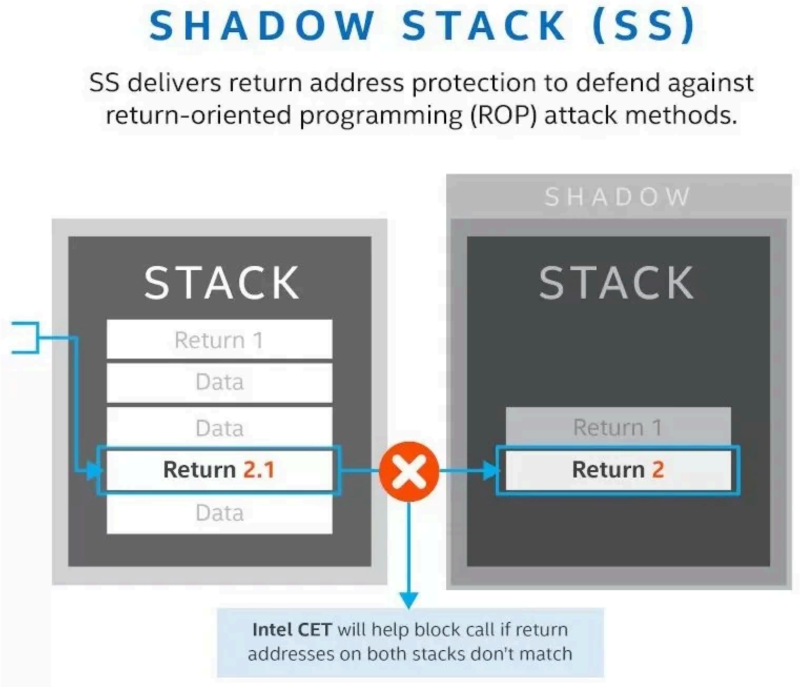
- Sometimes you have just a little space that you can write with your stack overflow.
- In some circumstances, you may manipulate the RSP to "increase" your stack space.
- Some stack pivoting gadgets include `leave; ret`, `pop rsp; ret`, and `xchg, rsp; ret`.

One Gadget

One Gadget

- Sometimes you just need one gadget to do the job. Luckily, glibc may include a gadget that will execute execve("/bin/sh") when you return to just one gadget.
- They are not easy to find by hand. Tools like one_gadget can help.
- You need to respect certain constraints to trigger a one gadget. For some versions of glibc, these are not so easy to fulfill.

Defeating Shadowstack



SHADOW STACK (SS)

SS delivers return address protection to defend against return-oriented programming (ROP) attack methods.

Intel CET will help block call if return addresses on both stacks don't match

Jump-Oriented Programming & Call-Oriented Programming

- With the Shadowstack, now `ret` instructions are lava, don't touch them.
- You can use gadgets that do indirect branching like `jmp` and `call` to avoid `ret`. These code reuse techniques are just like `ROP`, and like all of them, they are more examples of weird machines.