30. November 2022

Verwundbare IT-Systeme: Moderne Angriffstechniken zum Ausnutzen von Software Sicherheitslücken

Msc. Oussama Draissi
Prof. Dr.-Ing. Lucas Davi
Sichere Software Systeme
Universität Duisburg-Essen

Team

Prof. Dr.-Ing. Lucas Davi

Msc. O. Draissi

MSc. S. Surminski

MSc. D. Paaßen

MSc. T. Cloosters

MSc. J. Giesen

MSc. C. Niesler

Projekte

NEC

HUAWEI

CASA
CYBER SECURITY IN THE AGE OF LARGE-SCALE ADVERSARIES

CROSSING

erc
European Research Council
Established by the European Commission

Forschungsbereiche

Software Sicherheit

- Zero-Days
- Programmfluss-integrität
- Speicher-randomisierung

Trusted Computing

- Remote Attestation
- Trusted Platform Module (TPM)

Hardwarebasierte Sicherheit

- Intel Software Guard Extensions (SGX)
- ARM TrustZone
- Rowhammer Angriffe

Smart Contract Sicherheit

- Laufzeitvalidierung
- Reverse-Engineering
- Re-Entrancy Angriffe

Cybersicherheit in den Nachrichten

WDR

Wetter Verkehr im WDR

Nachrichten Sport Wissen Verbraucher Kultur Unterhaltung

Cyberangriff auf die Uni Duisburg-Essen legt digitale Dienste lahm

Stand: 28.11.2022, 14:38 Uhr

Das Problem mit Sicherheitslücken in Software

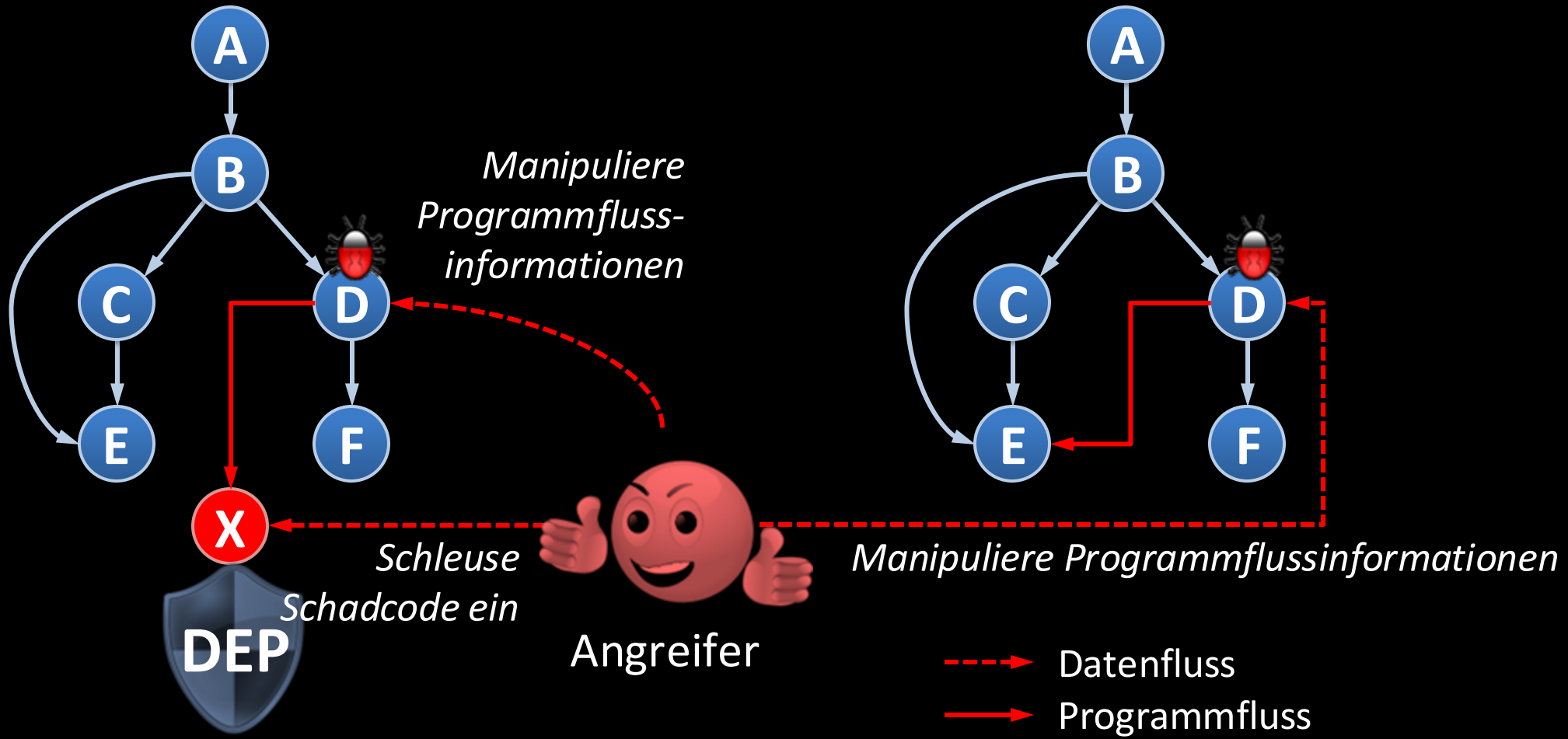
- Software wird immer anspruchsvoller und komplexer
- Die meisten Softwareentwickler sind keine Sicherheitsexperten
- Testen ist teuer

Wahrscheinlichkeit von Software Fehlern (Zero-Days) ist extrem hoch – man bedenke wie oft Software aus Sicherheitsgründen aktualisiert werden muss

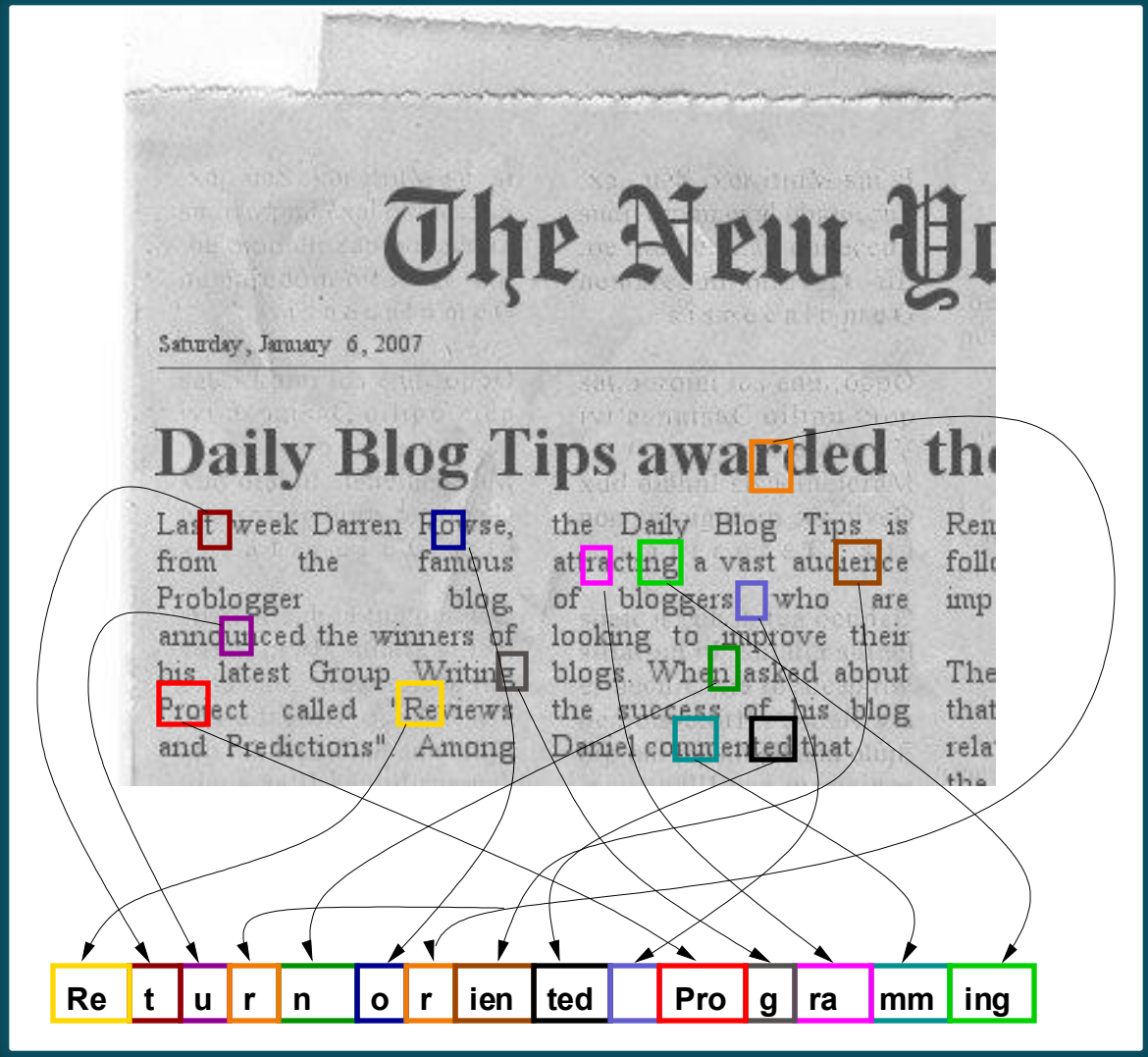
Wie können Software Fehler ausgenutzt werden?

Code-Injection Angriff

Return-Oriented Programming Angriff

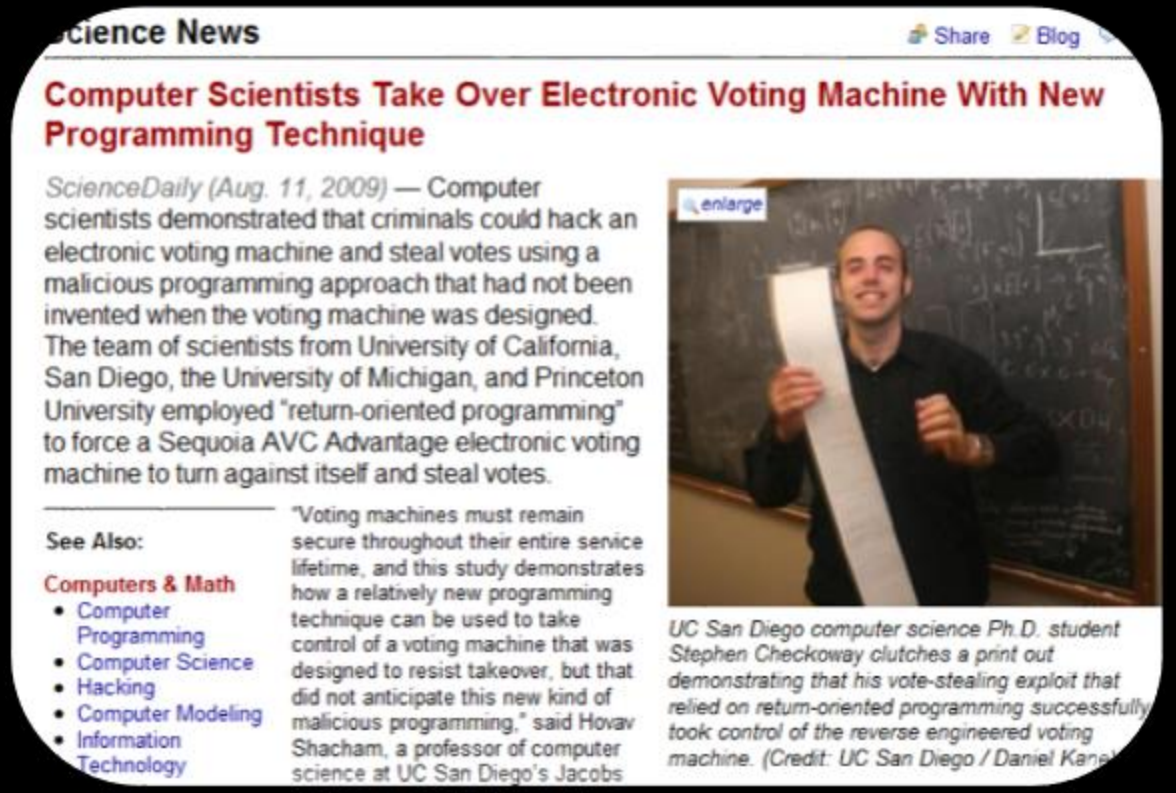


Return-Oriented Programming (ROP)



Ausgewählte ROP Angriffe aus der Praxis

Aus aktuellem Anlass: Wahlmaschinen Hack



cience News

Share Blog

Computer Scientists Take Over Electronic Voting Machine With New Programming Technique

ScienceDaily (Aug. 11, 2009) — Computer scientists demonstrated that criminals could hack an electronic voting machine and steal votes using a malicious programming approach that had not been invented when the voting machine was designed. The team of scientists from University of California, San Diego, the University of Michigan, and Princeton University employed "return-oriented programming" to force a Sequoia AVC Advantage electronic voting machine to turn against itself and steal votes.

"Voting machines must remain secure throughout their entire service lifetime, and this study demonstrates how a relatively new programming technique can be used to take control of a voting machine that was designed to resist takeover, but that did not anticipate this new kind of malicious programming," said Hovav Shacham, a professor of computer science at UC San Diego's Jacobs

enlarge

UC San Diego computer science Ph.D. student Stephen Checkoway clutches a print out demonstrating that his vote-stealing exploit that relied on return-oriented programming successfully took control of the reverse engineered voting machine. (Credit: UC San Diego / Daniel Kane

See Also:

Computers & Math

- Computer Programming
- Computer Science
- Hacking
- Computer Modeling
- Information Technology

Stagefright

Stuxnet

Wie funktioniert das Ausnutzen einer Software Sicherheitslücke?

Kompilieren des Additionsprogramms

Ausgabe: “Gebe 2 Zahlen ein:”
Warte auf Eingabe von Zahl1 und Zahl2
Berechne: Zahl3 = Zahl1+Zahl2
Ausgabe: “Ergebnis:”
Ausgabe: Zahl3
Gehe zum Anfang

Mein Additionsprogramm als Textdatei

Kompilieren

CODE

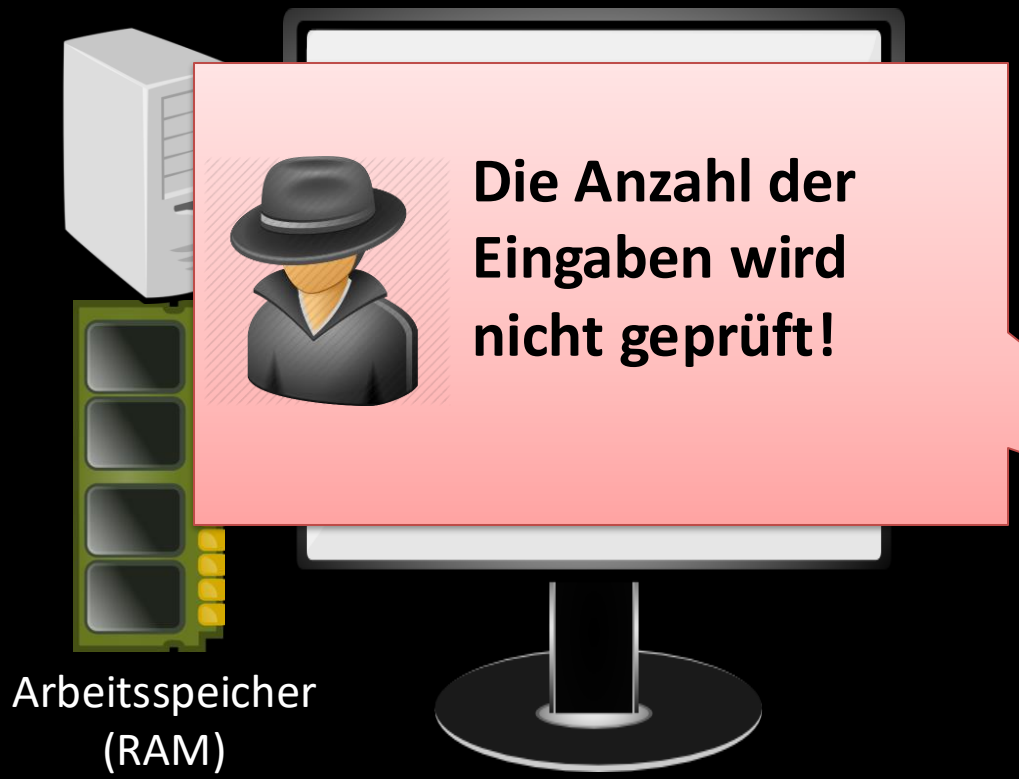
SCHREIB “Gebe 2 Zahlen ein:”
WARTE_AUF_EINGABE
LADE S1
LADE S2
ADD
SPEICHER S3
SCHREIB “Ergebnis:”
SCHREIB S3
SPRING S4

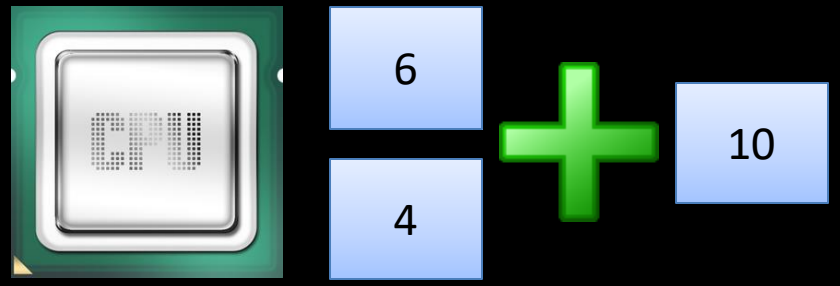
DATEN

S1: Zahl1
S2: Zahl2
S3: Zahl3
S4: Programmanfang

Das Additionsprogramm In Mikrobefehlen

Ausführen des Additionsprogramms

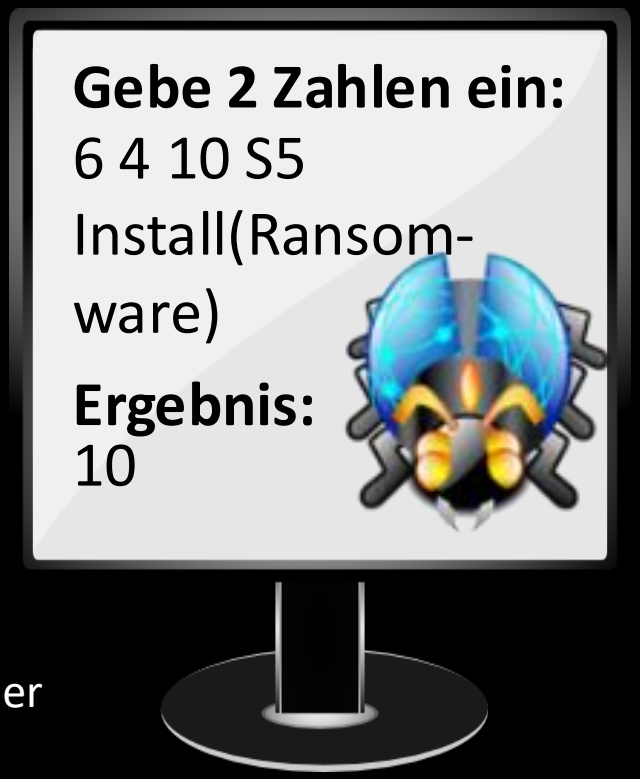




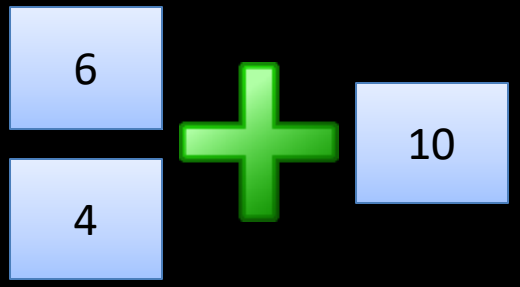
S1:	6
S2:	4
S3:	10
S4:	S6 (Programmstart)
S5:	
	SCHREIB “Gebe 2 Zahlen ein:”
S7:	WARTE_AUF_EINGABE
S8:	LADE [S1]
S9:	LADE [S2]
S10:	ADD
S11:	SPEICHER [S3]
S12:	SCHREIB “Ergebnis:”
S13:	SCHREIB [S3]
S14:	SPRING [S4]

Schadcode Einschleusen

Arbeitsspeicher (RAM)



Prozessor



IP →

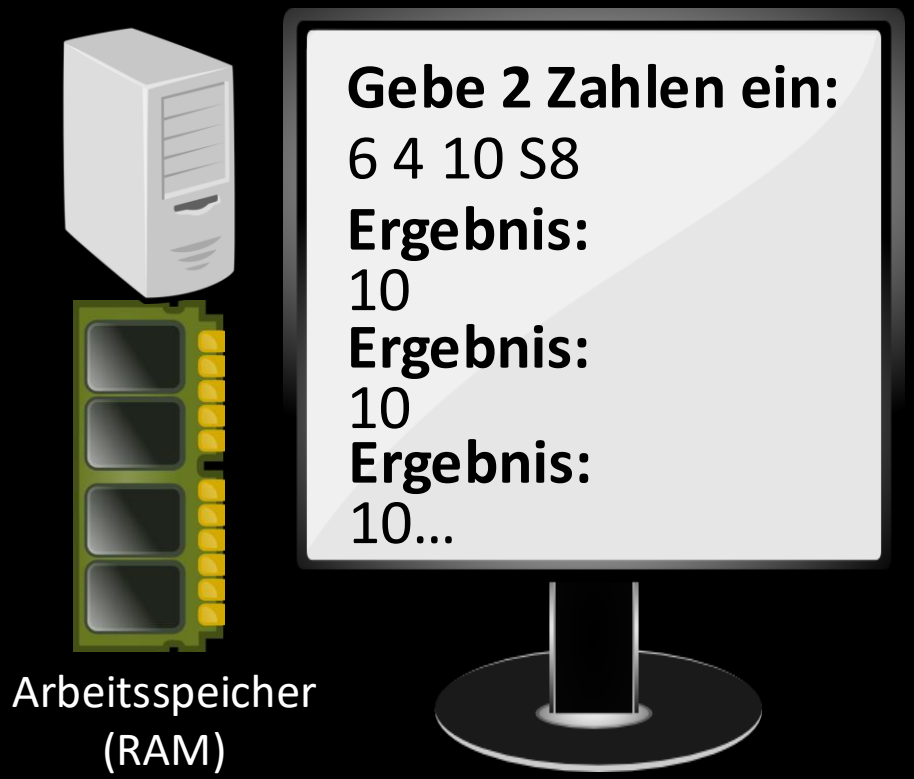
S1:	6
S2:	4
S3:	10
S4:	S5 *(vorher S6)*
S5:	Install(Ransomware)
S6:	SCHREIB “Gebe 2 Zahlen ein:”
S7:	WARTE_AUF_EINGABE
S8:	LADE [S1]
S9:	LADE [S2]
S10:	ADD
S11:	SPEICHER [S3]
S12:	SCHREIB “Ergebnis:”
S13:	SCHREIB [S3]
S14:	SPRING [S4]

Abwehrmethode gegen Code Einschleusen?

Verbiete die Ausführung von Code
im Datenspeicher! (DEP)

Sind wir jetzt sicher gegen Software Angriffe?

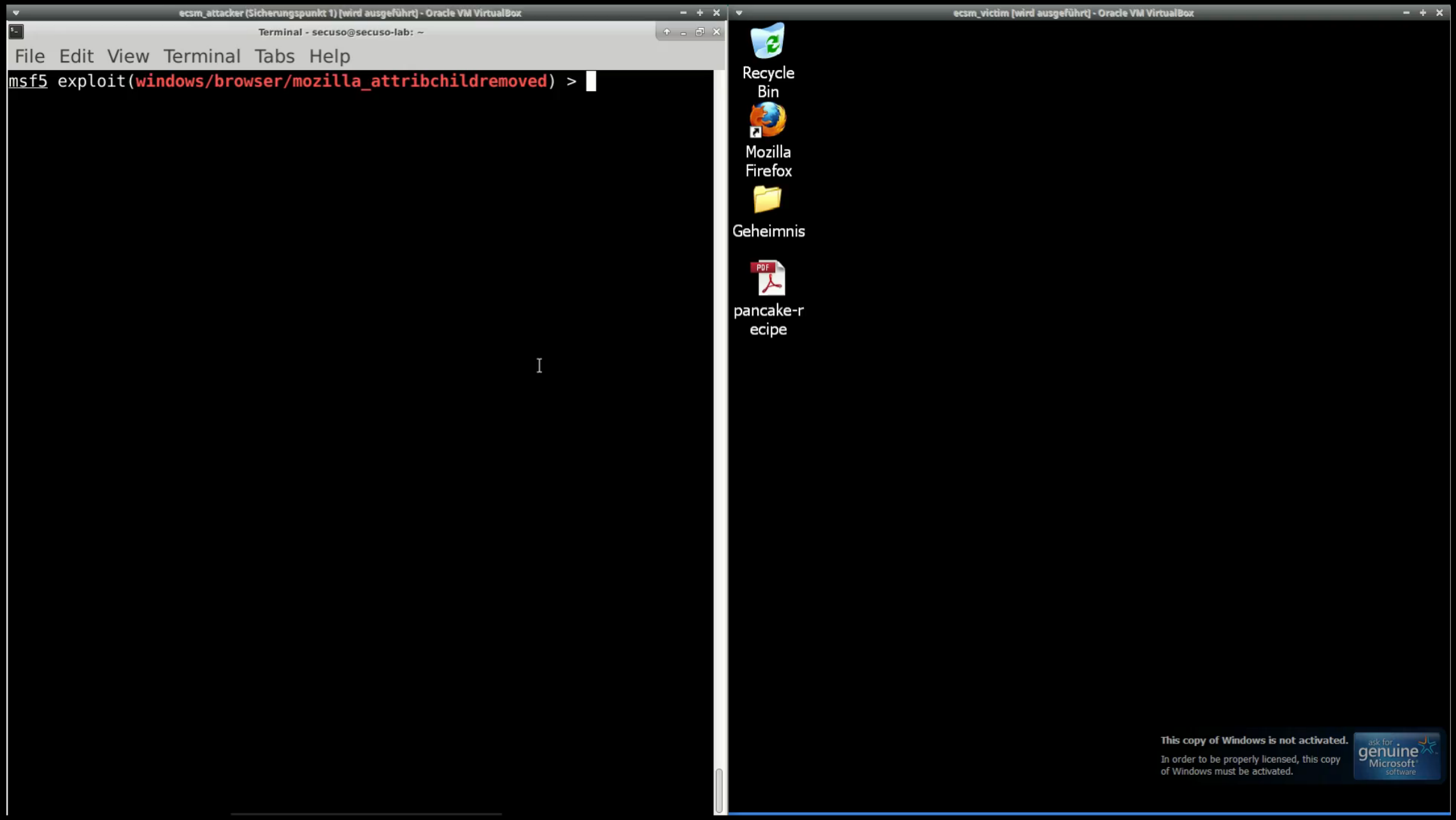
Code Wiederverwendung (ROP)



IP →

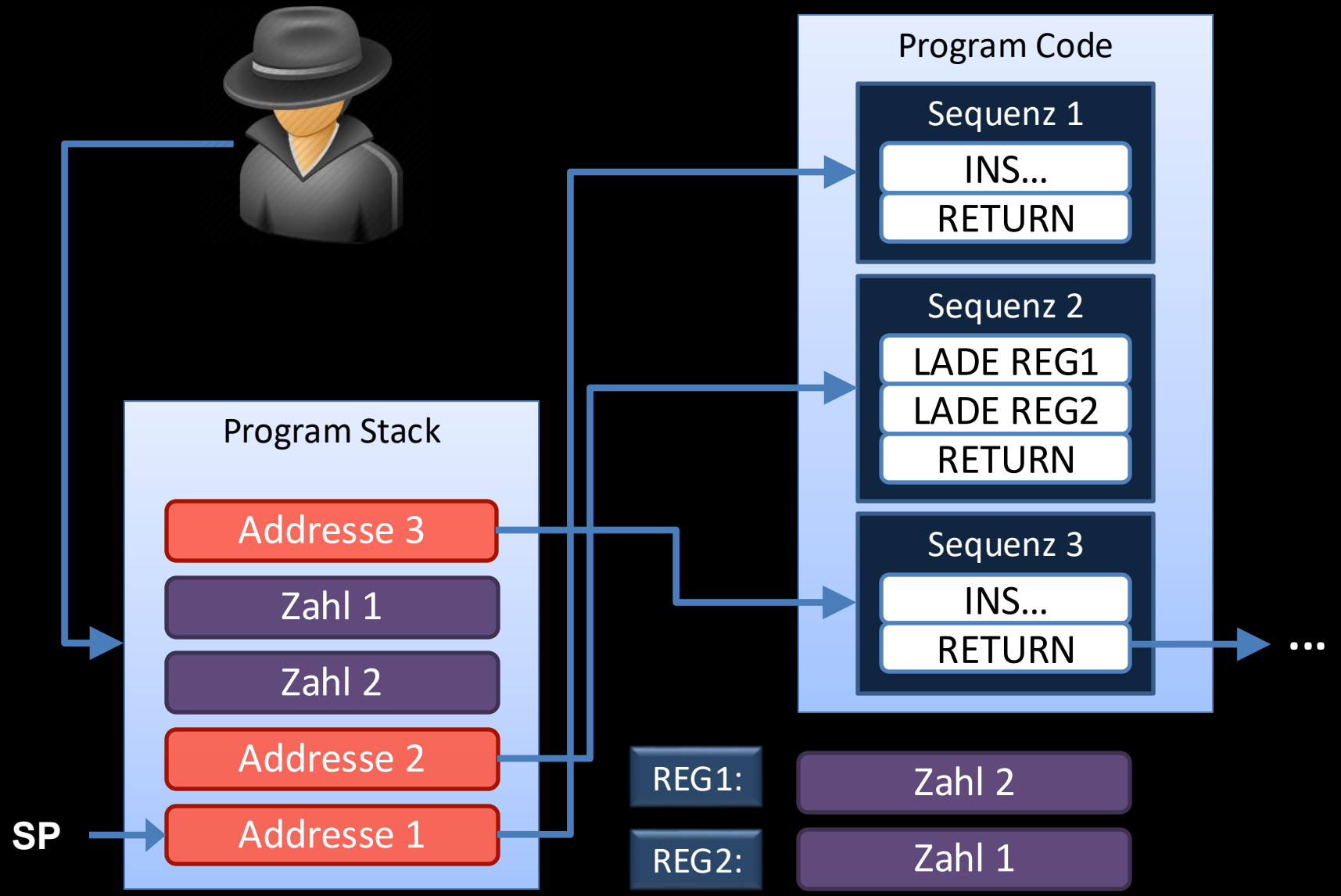
S1:	6
S2:	4
S3:	10
S4:	S8 *(vorher S6)*
S5:	
S6:	SCHREIB “Gebe 2 Zahlen ein:”
S7:	WARTE_AUF_EINGABE
S8:	LADE [S1]
S9:	LADE [S2]
S10:	ADD
S11:	SPEICHER [S3]
S12:	SCHREIB “Ergebnis:”
S13:	SCHREIB [S3]
S14:	SPRING [S4]

Wie mächtig sind ROP Angriffe?

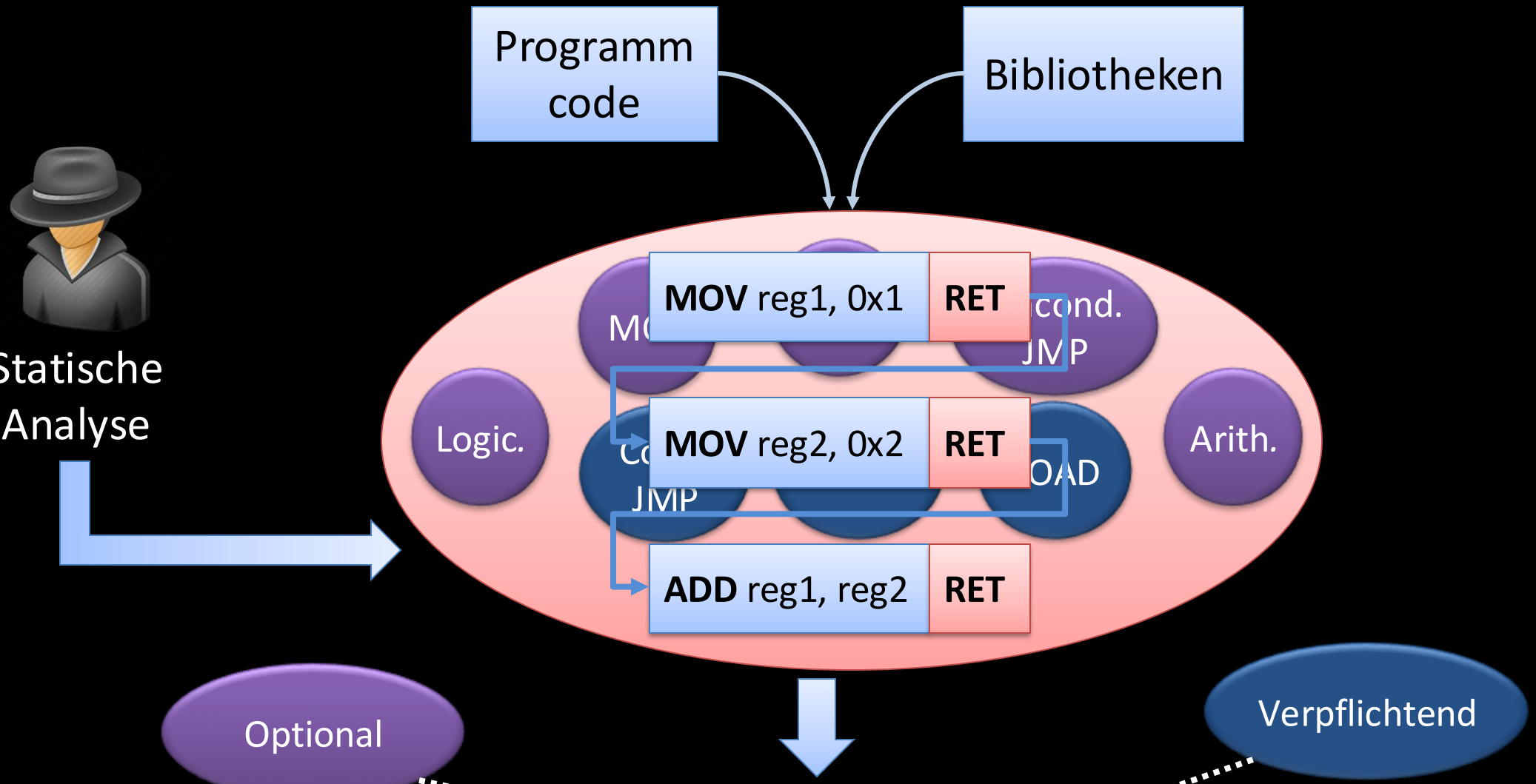

ecsm_attacker (Sicherungspunkt 1) [wird ausgeführt] - Oracle VM VirtualBox
Terminal - secuso@secuso-lab: ~
File Edit View Terminal Tabs Help
msf5 exploit(windows/browser/mozilla_attribchildremoved) >
ecsm_victim [wird ausgeführt] - Oracle VM VirtualBox
Recycle Bin
Mozilla Firefox
Geheimnis
pancake-recipe
This copy of Windows is not activated.
In order to be properly licensed, this copy of Windows must be activated.
ask for genuine Microsoft software

ROP Angriffsdetails

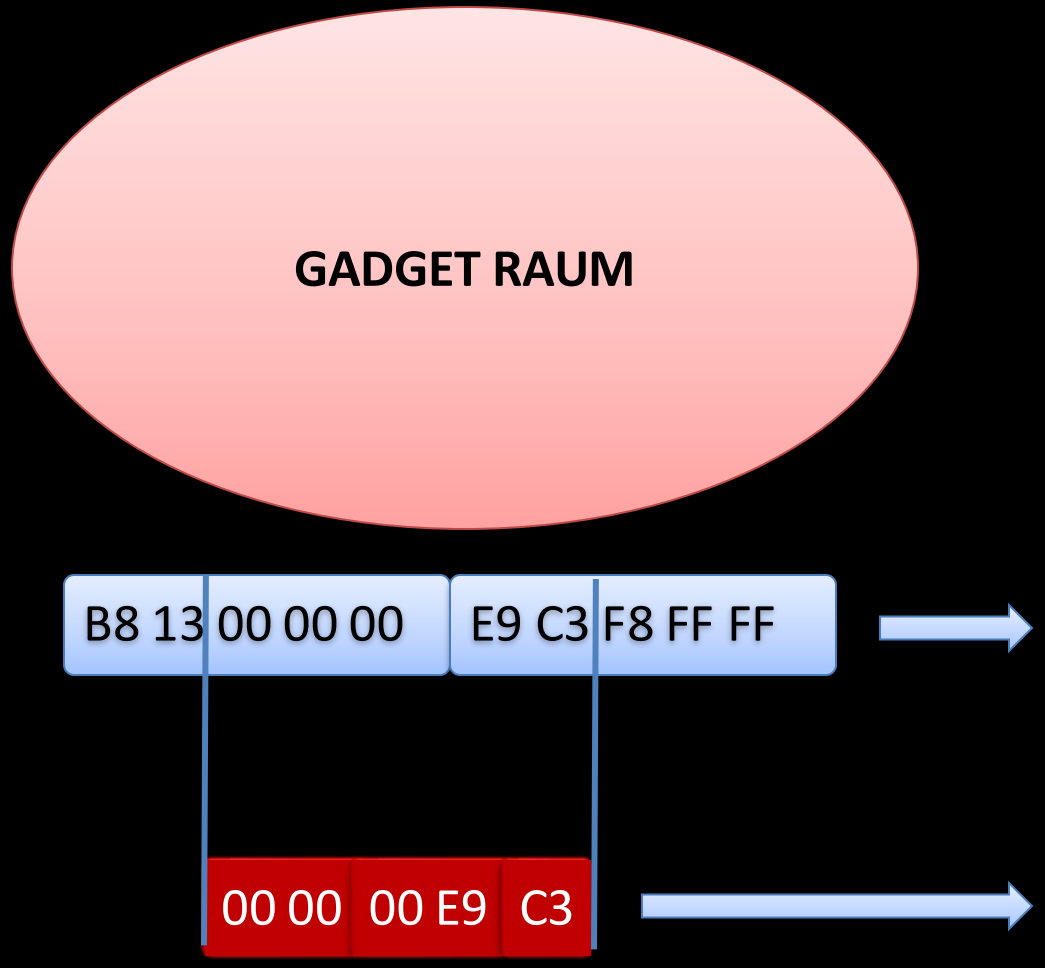
ROP Angriffstechnik



Entscheidend ist die Code Basis



PCs und Laptops sind besonders betroffen



Beabsichtigter Code

```
mov $0x13,%eax
jmp 3aae9
```

Unbeabsichtigter Code

```
add %al,(%eax)
add %ch,%cl
ret
```

Wie können wir uns gegen diese Angriffe schützen?

ethoden

Ran

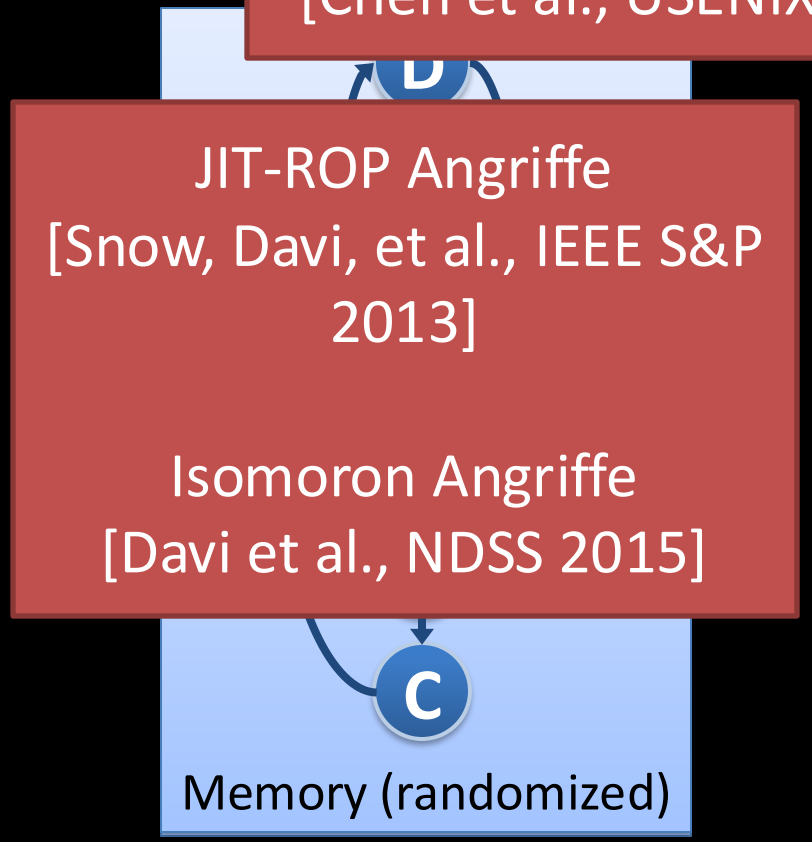
[Cohen 1993 & L

COOP Angriffe
[Schuster et al., IEEE S&P 2015]

COOP++ Angriffe
[Chen et al., USENIX 2021]

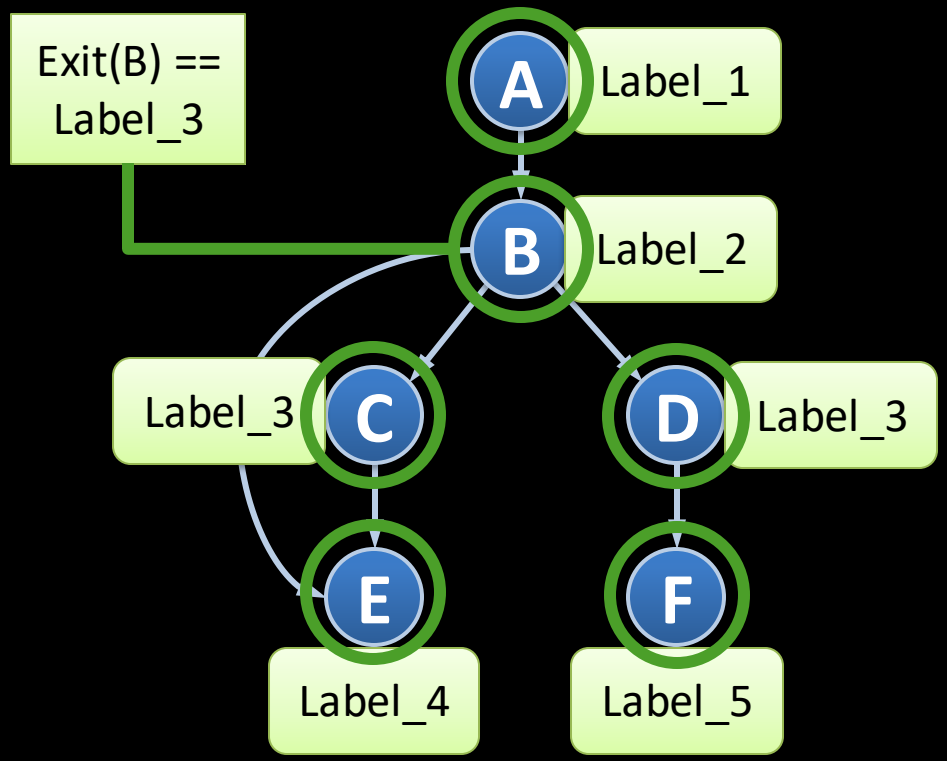
JIT-ROP Angriffe
[Snow, Davi, et al., IEEE S&P 2013]

Isomoron Angriffe
[Davi et al., NDSS 2015]



Programmflussintegrität

[Abadi et al., CCS 2005 & TISSEC 2009]



Moderne Sicherheitstechnologien

Intel CET

- Neue Intel Prozessoren sollen Control-Flow Enforcement Technology (CET) unterstützen
- Hardware-basierter Shadow Stack um Rücksprungadressen zu schützen

Microsoft CFG

- Applikationen für Windows 10 können mittels Microsoft Control-Flow Guard (CFG) geschützt werden
- Sicherheitscheck für Funktionsaufrufe

Google Clang-CFI

- Neues Compiler Feature um Control-Flow Integrity (CFI) für Funktionsaufrufe zu implementieren
- Anwendungsbeispiel ist der Chrome Web Browser

RISC-V Architecture

- Beliebte open-source Architektur
- Entwickler können Prozessoren Designen
- Nutzung in der Praxis
  - Googles: Android RISC-V Branch Sicherheitschip in Telefonen (OpenTitan)
  - Apple: ersetzt Embedded-Kerne (WiFi, Thunderbold, NAND, ...)
  - Internet der Dinge: z.B. RISC-V ESP32

Ist es wirklich sicherer als x86?

RISC-V vs ROP

Klassisches ROP
(Eine Instruktion für das return)

	X86	X86_64	ARM32	RISC-V	ARM64
Schreibbarer Programm Counter	×	×	●	×	×
Stack-return Instruktion	●	●	●	×	×
Max. Argumente in Registern	0	4–6	4	8	8
Spezielle Funktionsaufrufregister	●	●	●	×	×
Instruktionsgröße	1	1	4/2	4/2	4

Unintendierte Gadget

Spezielle Gadgets für Funktionsaufrufe nötig

Ist RISC-V die Lösung für ROP-Angriffe?

October 26, 2022: Limassol, Cyprus

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

TECHNISCHE UNIVERSITÄT DARMSTADT

RAiD 2022

RiscyROP: Automated Return-Oriented Programming Attacks on RISC-V and ARM64

Tobias Cloosters, David Paaßen, Jianqiang Wang, Oussama Draissi, Patrick Jauernig, Emmanuel Stapf, Lucas Davi, Ahmad Sadeghi

25th International Symposium on Research in Attacks, Intrusions and Defenses

Evaluation: CVE-2013-2028 Exploit (nginx)

Malicious
3-argument call
to *mprotect*

x86_64

```
; avoid side-effect (next xor)
pop rax  ; some known address
ret

; third argument (0x7)
pop rdx
xor [rax-0x77], cl
ret

; second argument (0x1000)
pop rsi
ret

; first argument (shellcode dst)
p rdi
t
```

RISC-V

```
c.ldsp      a0,0x8(sp)
c.j         0x41c6f4  ; constant jump
c.ldsp      ra,0x48(sp)
c.ldsp      s0,0x40(sp)
c.ldsp      s1,0x38(sp)
c.ldsp      s2,0x30(sp)
c.ldsp      s3,0x28(sp)
c.ldsp      s4,0x20(sp)
c.ldsp      s5,0x18(sp)
c.addi16sp  sp,0x50
c.jr        ra        ; return

c.addi4spn  s1,sp,0x0
c.ld        a1,0x8(s1)
c.ld        a2,0x10(s1)
c.ld        a3,0x18(s1)
c.ld        a4,0x20(s1)
c.ld        a5,0x28(s1)
sd          a6,0x0(a0)
c.sd        a1,0x8(a0)
c.sd        a2,0x10(a0)
c.sd        a3,0x18(a0)
c.sd        a4,0x20(a0)
c.sd        a5,0x28(a0)
sd          a0,0x2a8(s0)
c.ldsp      ra,0x18(sp)
c.ldsp      s0,0x10(sp)
c.ldsp      s1,0x8(sp)
c.mv        a0,s2
c.ldsp      s2,0x0(sp)
c.addi16sp  sp,0x20
c.jr        ra        ; return
```

ARM64

```
ldr   x0, [sp, #0x40]
b     #0x2e548  ; constant jump
adrp  x1, #0x122000
ldr   x1, [x1, #0xa40]
ldr   x2, [sp, #0x48]
ldr   x3, [x1]
subs  x2, x2, x3
mov   x3, #0
b.ne  #0x2e5bc  ; conditional jump
ldp   x21, x22, [sp, #0x20]
ldp   x23, x24, [sp, #0x30]
ldp   fp, lr, [sp], #0x50
br    lr        ; return

ldr   x4, [sp, #0x60]
cbnz  x0, #0xafa90
ldp   x21, x22, [sp, #0x20]
mov   x20, #0x1f4
ldp   x23, x24, [sp, #0x30]
ldp   x25, x26, [sp, #0x40]
ldp   x27, x28, [sp, #0x50]
adrp  x0, #0x122000
ldr   x0, [x0, #0xa40]
ldr   x1, [sp, #0x1d8]
ldr   x2, [x0]
subs  x1, x1, x2
mov   x2, #0
b.ne  #0xaff34  ; conditional jump
mov   x0, x20
ldp   x19, x20, [sp, #0x10]
ldp   fp, lr, [sp], #0x1e0
br    lr        ; return

mov   x2, x22
mov   x1, x19
mov   x0, x26
blr   x4        ; indirect call
```

Wenige Instruktionen
Ausschließlich unintendierte Instruktionen

RiscyROP zeigt:

RISC-V macht ROP zwar anspruchsvoller, **aber ist nicht** in der Lage es zu verhindern!

Lange und komplexe Gadgets mit vielen Nebeneffekten

Was kann der Nutzer tun?

- Begrenzte Möglichkeiten
- Up-to-date Software und auch Hardware
- Skript-Ausführung eingrenzen
- Verdächtige Links nur in einer virtuellen Maschine öffnen

Zusammenfassung

- Return-Oriented Programming (ROP) Angriffe werden genutzt, um Sicherheitslücken in Software auszunutzen
- Schlechte Nachricht
  - Der Nutzer kann sich nur begrenzt vor ROP Angriffen schützen, da diese Angriffe Zero-Day Lücken ausnutzen
- Gute Nachricht
  - Akademische und industrielle Forschungslösungen zur Verteidigung von ROP Angriffen werden immer effektiver und effizienter