DEEPSEC

BRACE YOURSELVES
EXPLOIT AUTOMATION IS COMING

Andreas Follner

TU Darmstadt / CRISP

TECHNISCHE UNIVERSITÄT DARMSTADT

CRISP
Center for Research in Security and Privacy

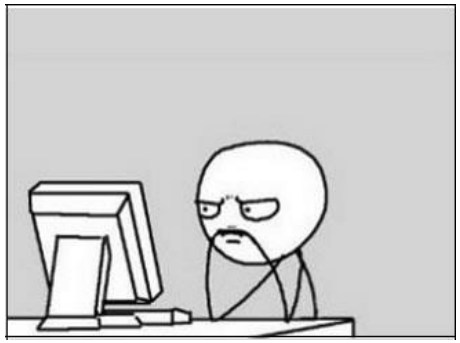
About Me

TECHNISCHE UNIVERSITÄT DARMSTADT

- PhD student at TU Darmstadt (Germany)
  - Center for Research in Security and Privacy (CRISP)

- Originally from Vienna :)
- Interests: applied research in exploitation and mitigation, binary analysis, reverse engineering, i.e.,

  staring at asm



andreas.follner@crisp-da.de

Scope Refinement

TECHNISCHE UNIVERSITÄT DARMSTADT

- **NOT** confused deputy
- **NOT** Java
- **NOT** Android permissions
- **NOT** SQL
- **DEFINITELY NOT** exploit kits

- Good old memory corruption
  - Return-Oriented Programming (ROP)
    - Automated generation of ROP chains

```
eax            0x0      0
ecx            0xbffff8f0       -1073743632
edx            0xbffff52c       -1073744596
ebx            0xb7fd0000       -1208156160
esp
ebp
esi
edi
eip
eflags
cs
ss
ds
es
fs
gs
(gdb) x/
0xbffff8
0xbffff8
0xbffff8
0xbffff8
```

```
0x4162f     pop rax ; add al, 0x5b ; pop rbp ; pop r12
            ; ret ;
0x781c4     pop rdi ; pop r8 ; add rsp, 0x18 ; pop rbx
            ; pop rbp ; pop r12 ; pop r13 ; ret ;
0xe7c1      pop rsi ; add byte ptr [rax], al ; adc
            dword ptr [rax], eax ; sbb byte ptr [rax],
            al ; ret 0x29 ;
0x7800d     pop rcx ; mov eax, ebp ; add rsp, 8 ; pop
            rbx ; pop rbp ; ret ;
0x41b13     pop rdx ; pop rbx ; pop rbp ; pop r12 ; ret
            ;
0x7800c     pop r9 ; mov eax, ebp ; add rsp, 8 ; pop
            rbx ; pop rbp ; ret ;
```

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

- Past
  - Recap basics (stack smashing, data execution prevention, return-oriented programming)
- Present
  - Current exploit development process
  - Mitigations
- Future
  - New attacks, new mitigations

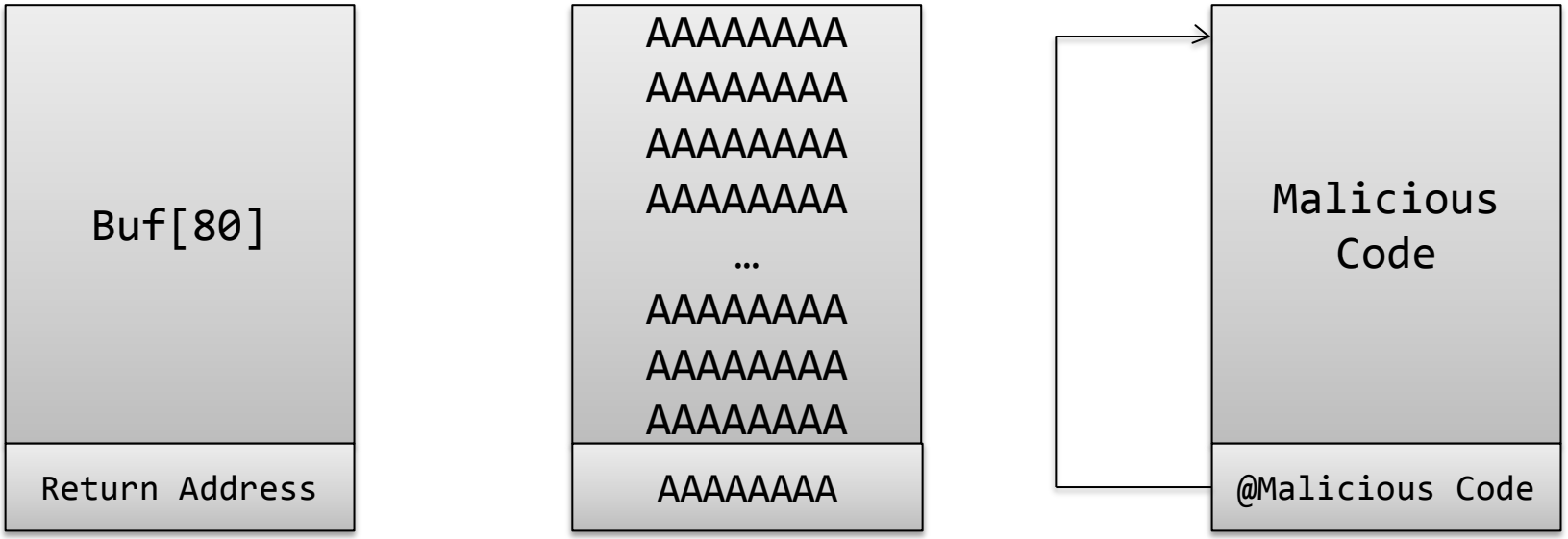
- ROPocop - Dynamic mitigation of code-reuse attacks (Follner et al.) – *Journal of Information Security and Applications (JISA)*
- Analyzing the gadgets - towards a metric to measure gadget quality (Follner et al.) – *International Symposium on Engineering Secure Software and Systems (ESSoS)*
- PSHAPE: Automatically combining gadgets for arbitrary method execution (Follner et al.) – *International Workshop on Security and Trust Management (STM)*

Once upon a time…

Stack Smashing

TECHNISCHE UNIVERSITÄT DARMSTADT

- Aleph1 - Smashing the Stack for Fun and Profit (Phrack 49, Nov. 1996)

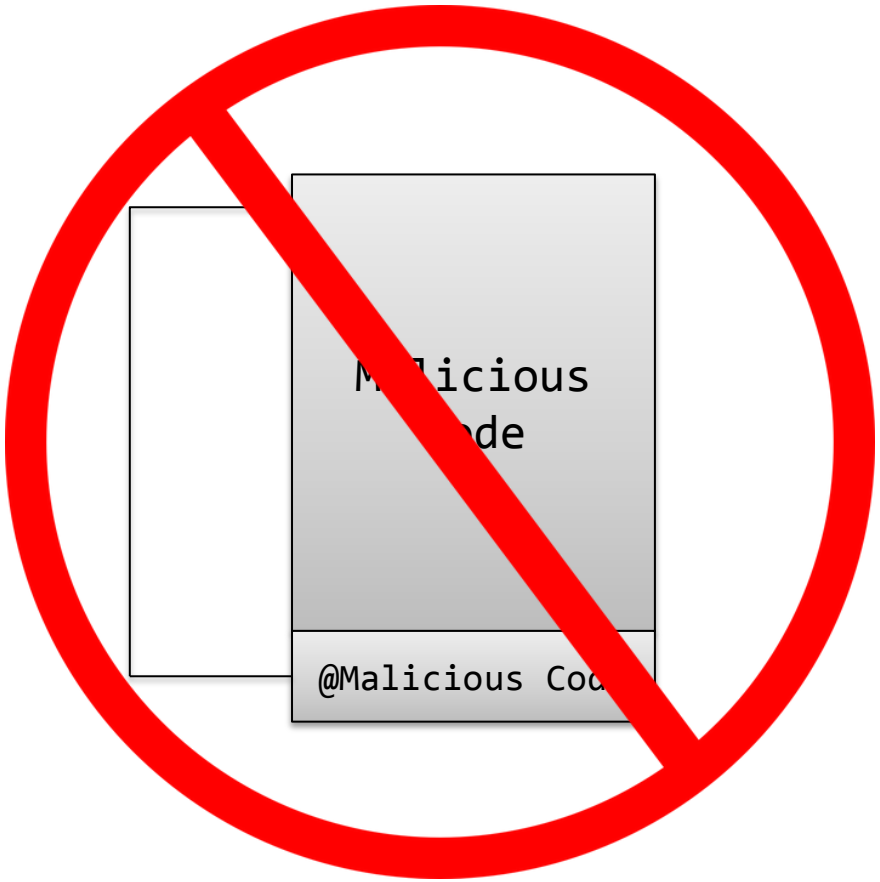


- Other attack vectors, e.g., use-after-free, type confusion

Data Execution Prevention (DEP)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Memory is either writable XOR executable
  - (e.g., heap / stack is writable → not executable)
- Enforced in hardware (NX bit)
- Standard on Linux, Windows, OS X, iOS, Android, …
- Prevents execution of injected code



DEEPSEC

Bypassing DEP

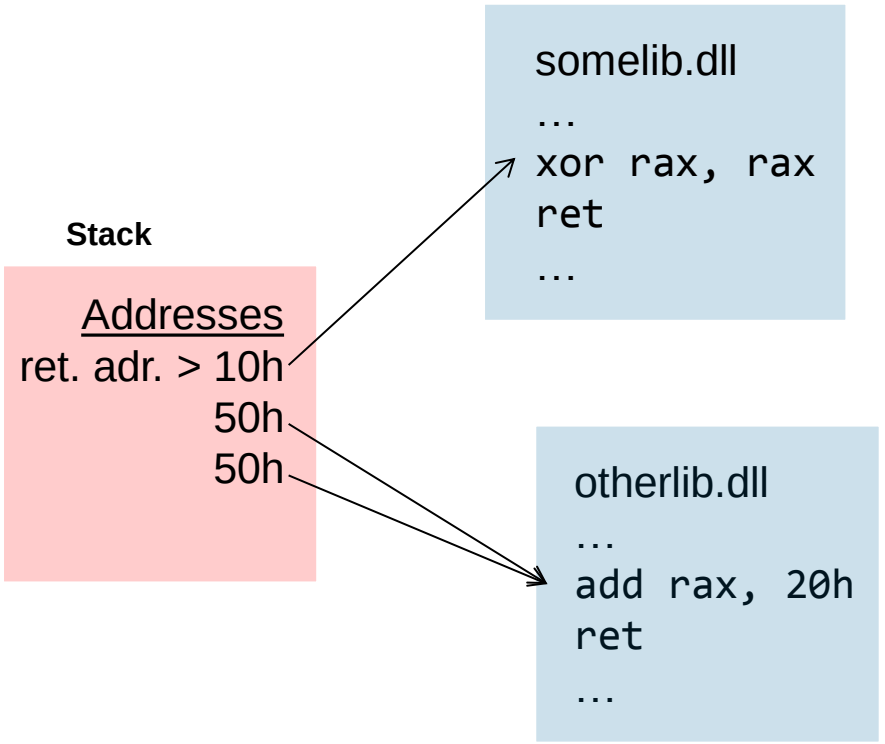
TECHNISCHE UNIVERSITÄT DARMSTADT

- Code ~~injection~~ reuse
- Return-oriented programming (ROP)

Bypassing DEP

- Code ~~injection~~ reuse
- Return-oriented programming (ROP)

- Inject a set of addresses pointing to *gadgets*
- Gadget = (short) sequence of instructions, ends with `ret`
- Gadgets execute consecutively



Bypassing DEP with ROP

TECHNISCHE UNIVERSITÄT DARMSTADT

- Goal: execute arbitrary code
  - Pure ROP exploit
    - Hope all required gadgets are available
  - Inject regular shellcode and use ROP to invoke OS API to make shellcode executable
    - E.g., VirtualProtect, mprotect

```
BOOL WINAPI VirtualProtect(
  _In_  LPVOID lpAddress,
  _In_  SIZE_T dwSize,
  _In_  DWORD  flNewProtect,
  _Out_ PDWORD lpflOldProtect
);
```

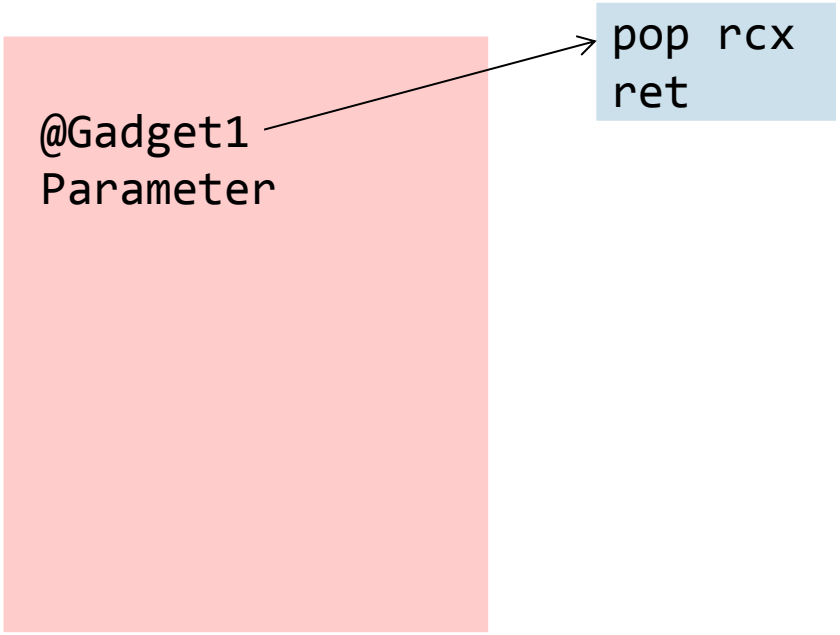
Bypassing DEP with ROP

TECHNISCHE UNIVERSITÄT DARMSTADT

- Invoke function call with parameters we control using ROP

  (x86-64 passes parameters in registers rcx, rdx, r8, r9)

- Mix gadgets and parameters



DEEPSEC

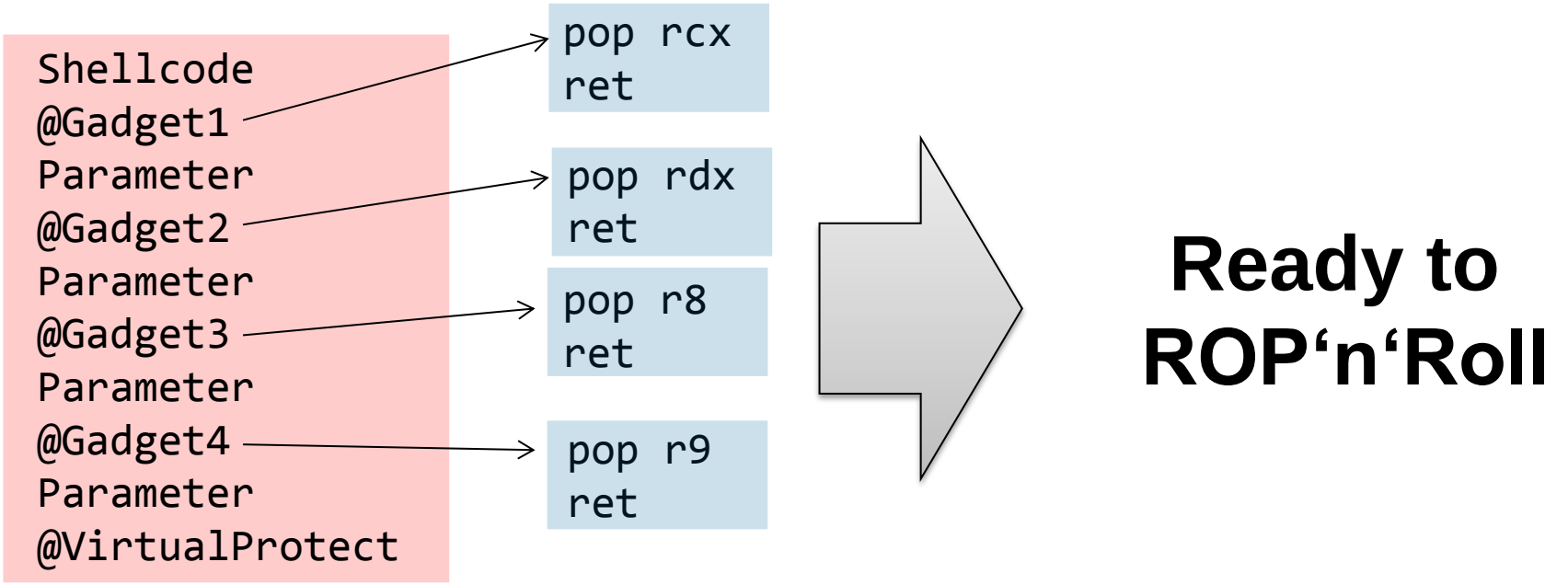
Bypassing DEP with ROP

TECHNISCHE UNIVERSITÄT DARMSTADT

- Invoke function call with parameters we control using ROP

  (x86-64 passes parameters in registers rcx, rdx, r8, r9)

- Mix gadgets and parameters



DEEPSEC

Calling VirtualProtect using ROP

TECHNISCHE UNIVERSITÄT DARMSTADT

rsp	rbp	rax	rbx	rcx	rdx	rdi	rsi	r8	r9	r10	r11	r12	r13	r14	r15	rip

Address	Content
ff0000h	530000h
ff0008h	1
ff0010h	2
ff0018h	ff1018h
ff0020h	520000h
ff0028h	ffd2h
ff0030h	1337h
ff0038h	g07g07h

< ret. Adr.

```
memcpy(smallbuff, inbuff, 1000)
return
```

Buffer Overflow!

DEEPSEC

Calling VirtualProtect using ROP

TECHNISCHE UNIVERSITÄT DARMSTADT

rsp	rbp	rax	rbx	rcx	rdx	rdi	rsi	r8	r9	r10	r11	r12	r13	r14	r15	rip

Address	Content
ff0000h	440000h
ff0008h	ff0100h
ff0010h	450000h
ff0018h	1000h
ff0020h	460000h
ff0028h	40h
ff0030h	470000h
ff0038h	feff00h

< ret. Adr.

```
memcpy(smallbuff, inbuff, 1000)
return
```

Remove some registers…

Calling VirtualProtect using ROP

TECHNISCHE UNIVERSITÄT DARMSTADT

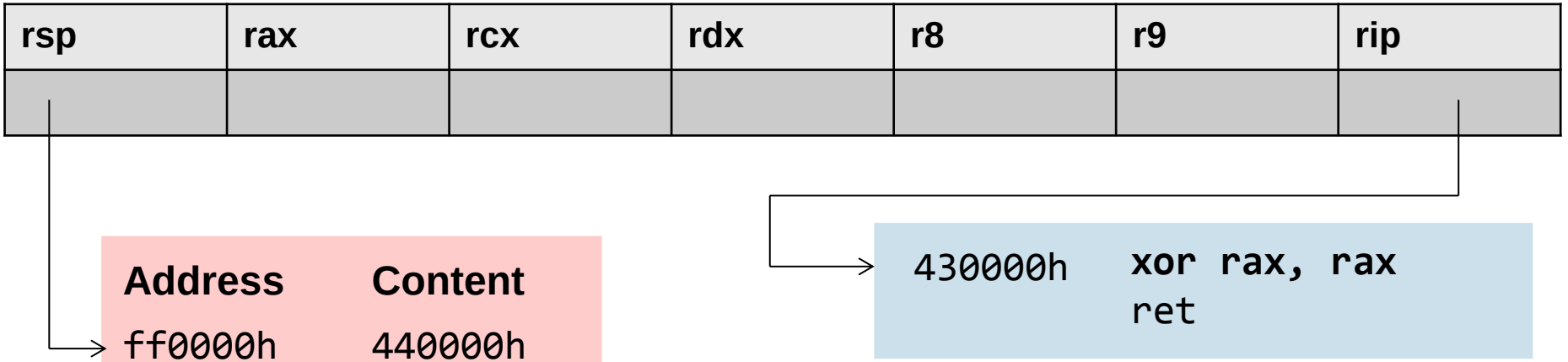
rsp	rax	rcx	rdx	r8	r9	rip

Address	Content
ff0000h	440000h
ff0008h	ff0100h
ff0010h	450000h
ff0018h	1000h
ff0020h	460000h
ff0028h	40h
ff0030h	470000h
ff0038h	feff00h

```
memcpy(smallbuff, inbuff, 1000)
return
```

Switch to ASM…

Calling VirtualProtect using ROP



Address	Content
ff0000h	440000h
ff0008h	ff0100h
ff0010h	450000h
ff0018h	1000h
ff0020h	460000h
ff0028h	40h
ff0030h	470000h
ff0038h	feff00h

```
430000h    xor rax, rax
           ret
```

Calling VirtualProtect using ROP

rsp	rax	rcx	rdx	r8	r9	rip
	0					

Address	Content
ff0000h	440000h
ff0008h	ff0100h
ff0010h	450000h
ff0018h	1000h
ff0020h	460000h
ff0028h	40h
ff0030h	470000h
ff0038h	feff00h

```
430000h   xor rax, rax
          ret
```

Calling VirtualProtect using ROP

TECHNISCHE UNIVERSITÄT DARMSTADT

rsp	rax	rcx	rdx	r8	r9	rip
	0					

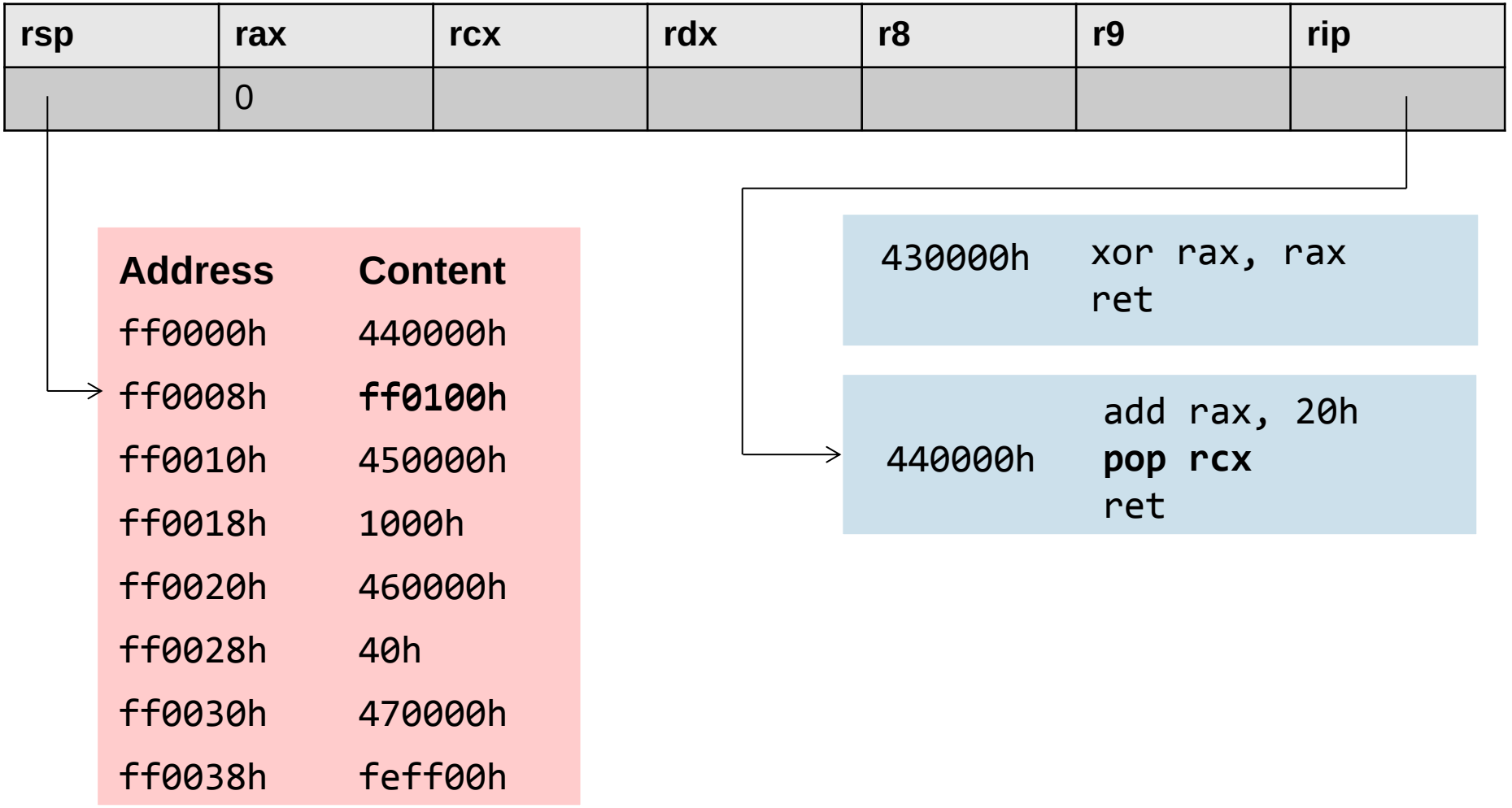
Address	Content
ff0000h	440000h
ff0008h	ff0100h
ff0010h	450000h
ff0018h	1000h
ff0020h	460000h
ff0028h	40h
ff0030h	470000h
ff0038h	feff00h

```
430000h   xor rax, rax
          ret
```

```
          add rax, 20h
440000h   pop rcx
          ret
```


DEEPSEC

Calling VirtualProtect using ROP



Calling VirtualProtect using ROP

TECHNISCHE UNIVERSITÄT DARMSTADT

rsp	rax	rcx	rdx	r8	r9	rip
	0	ff0100h				

Address	Content
ff0000h	440000h
ff0008h	ff0100h
ff0010h	450000h
ff0018h	1000h
ff0020h	460000h
ff0028h	40h
ff0030h	470000h
ff0038h	feff00h

```
430000h   xor rax, rax
          ret
```

```
          add rax, 20h
440000h   pop rcx
          ret
```


DEEPSEC

Calling VirtualProtect using ROP

TECHNISCHE UNIVERSITÄT DARMSTADT

rsp	rax	rcx	rdx	r8	r9	rip
	0	ff0100h				

Address	Content
ff0000h	440000h
ff0008h	ff0100h
ff0010h	450000h
ff0018h	1000h
ff0020h	460000h
ff0028h	40h
ff0030h	470000h
ff0038h	feff00h

```
430000h   xor rax, rax
          ret

440000h   add rax, 20h
          pop rcx
          ret

450000h   imul rax, rcx
          pop rdx
          ret
```


DEEPSEC

Calling VirtualProtect using ROP

rsp	rax	rcx	rdx	r8	r9	rip
	0	ff0100h				

Address	Content
ff0000h	440000h
ff0008h	ff0100h
ff0010h	450000h
ff0018h	**1000h**
ff0020h	460000h
ff0028h	40h
ff0030h	470000h
ff0038h	feff00h

```
430000h   xor rax, rax
          ret

440000h   add rax, 20h
          pop rcx
          ret

450000h   imul rax, rcx
          pop rdx
          ret
```

Calling VirtualProtect using ROP

TECHNISCHE UNIVERSITÄT DARMSTADT

rsp	rax	rcx	rdx	r8	r9	rip
	0	ff0100h	1000h			

Address	Content
ff0000h	440000h
ff0008h	ff0100h
ff0010h	450000h
ff0018h	1000h
ff0020h	460000h
ff0028h	40h
ff0030h	470000h
ff0038h	feff00h

```
430000h   xor rax, rax
          ret

440000h   add rax, 20h
          pop rcx
          ret

450000h   imul rax, rcx
          pop rdx
          ret
```


A FEW
MOMENTS LATER

Calling VirtualProtect using ROP

TECHNISCHE UNIVERSITÄT DARMSTADT

rsp	rax	rcx	rdx	r8	r9	rip
	0	ff0100h	1000h	40h	feff00h	

Address	Content
ff0000h	440000h
ff0008h	ff0100h
ff0010h	450000h
ff0018h	1000h
ff0020h	460000h
ff0028h	40h
ff0030h	470000h
ff0038h	feff00h

```
460000h   pop r8
          ret
```

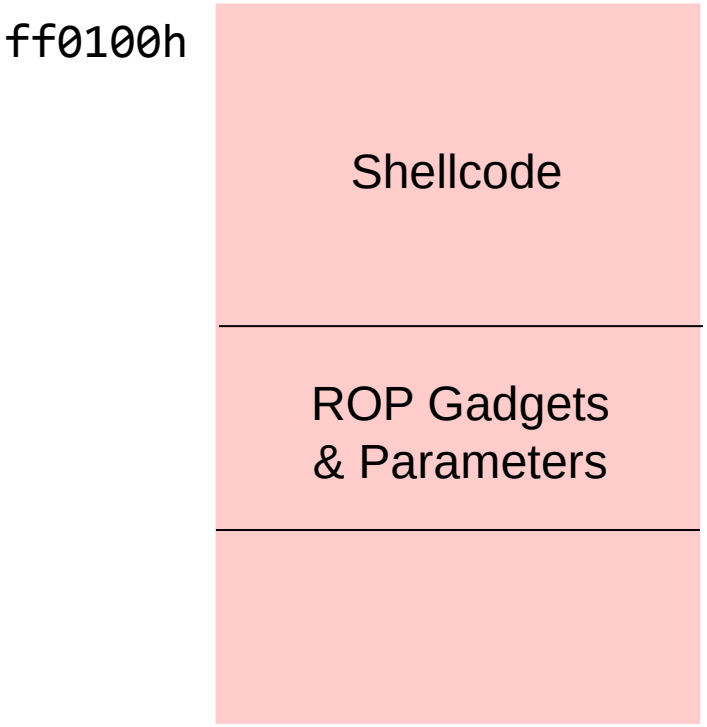
```
          sub rax, rdx
470000h   pop r9
          ret
```

Calling VirtualProtect using ROP

TECHNISCHE UNIVERSITÄT DARMSTADT

rsp	rax	rcx	rdx	r8	r9	rip
	0	ff0100h	1000h	40h	feff00h	

```
BOOL WINAPI VirtualProtect(
  _In_  LPVOID lpAddress,
  _In_  SIZE_T dwSize,
  _In_  DWORD  flNewProtect,
  _Out_ PDWORD lpflOldProtect
);
```



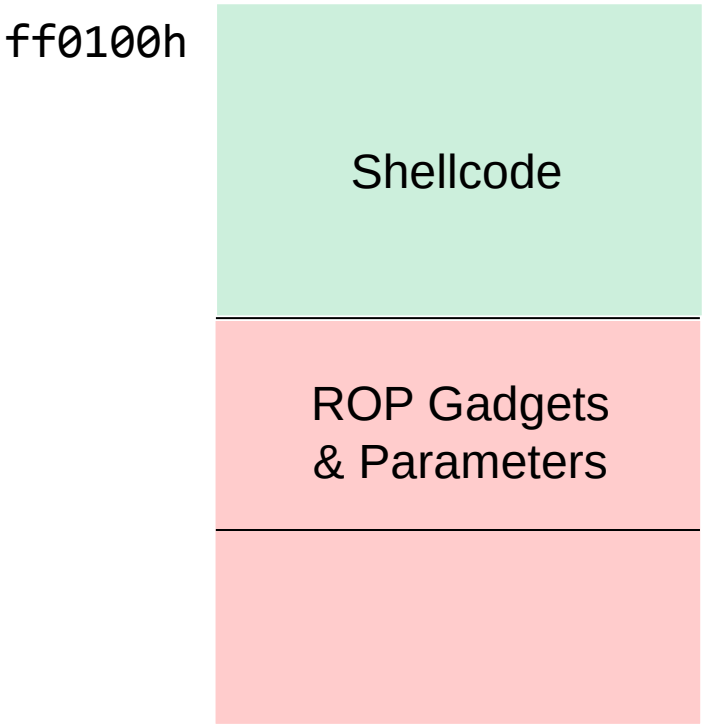
DEEPSEC

Calling VirtualProtect using ROP

TECHNISCHE UNIVERSITÄT DARMSTADT

rsp	rax	rcx	rdx	r8	r9	rip
	0	ff0100h	1000h	40h	feff00h	

```
BOOL WINAPI VirtualProtect(
  _In_  LPVOID lpAddress,
  _In_  SIZE_T dwSize,
  _In_  DWORD  flNewProtect,
  _Out_ PDWORD lpflOldProtect
);
```



DEEPSEC

How we currently develop ROP chains

ROP Chain Development

TECHNISCHE UNIVERSITÄT DARMSTADT

- Discover gadgets using ROPgadget, mona.py, …
- Output:

```
0x000000000041ad18: clc ; cmp edi, edi; jmp rcx
0x000000000041ad17: add al, bh; cmp edi, edi; jmp rcx
0x000000000041ad16: add byte ptr [rax], al; clc ; cmp edi, edi; jmp rcx
0x000000000041ad15: sbb al, 0; add al, bh; cmp edi, edi; jmp rcx
0x000000000041ad14: insb byte ptr es:[rdi], dx; sbb al, 0; add al, bh; cmp edi, edi
0x000000000041ad13: add byte ptr [rsp + rbx], ch; add al, bh; cmp edi, edi; jmp rcx
0x000000000041ad12: add byte ptr [rax], al; insb byte ptr es:[rdi], dx; sbb al, 0;
0x000000000041ad11: add byte ptr [rax], al; add byte ptr [rsp + rbx], ch; add al, bl
0x000000000041add1: cmp al, -1; call qword ptr [rcx]
0x000000000041adcf: add al, ah; cmp al, -1; call qword ptr [rcx]
0x000000000041ae01: cmp al, -1; call qword ptr [rsi]
0x000000000041adff: add al, ah; cmp al, -1; call qword ptr [rsi]
0x000000000041adfb: add byte ptr [rbp + rbx], dl; add al, ah; cmp al, -1; call qwor
0x000000000041adf9: add byte ptr [rax], al; add byte ptr [rbp + rbx], dl; add al, al
0x000000000041ae19: cmp al, -1; call qword ptr [rdx]
0x000000000041ae17: add al, ch; cmp al, -1; call qword ptr [rdx]
0x000000000041ae13: add byte ptr [rbp + rbx], ch; add al, ch; cmp al, -1; call qwor
0x000000000041ae11: add byte ptr [rax], al; add byte ptr [rbp + rbx], ch; add al, ch
0x000000000041b218: mov byte ptr [rbx + 0xffffffffffffffff], bl; jmp qword ptr [rax
0x000000000041b216: add byte ptr [rax], al; mov byte ptr [rbx + 0xffffffffffffffff]
0x000000000041b308: or byte ptr [rbp + 0xffffffffffffffff], bl; jmp qword ptr [rbx]
0x000000000041b306: add byte ptr [rax], al; or byte ptr [rbp + 0xffffffffffffffff],
0x000000000041b607: add al, 0x6b; ret
```

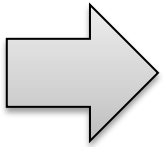

DEEPSEC

ROP Chain Development

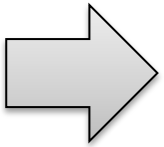
TECHNISCHE UNIVERSITÄT DARMSTADT

- Sift through gadgets, look for patterns like

```
pop rcx # ret
pop rdx # ret
pop r8 # ret
pop r9 # ret
```



- No such luck…



ROP Chain Development

TECHNISCHE UNIVERSITÄT DARMSTADT

- No such luck…
  - Allow additional instructions

```
pop rcx # mov rax, [rsi] # ret
pop rcx # pop rax # pop rbp # pop r10 # ret
pop rcx # add rax, [rbp-8] # ret
pop rcx # push rax # dec rcx # mov al, 1 # ret
```

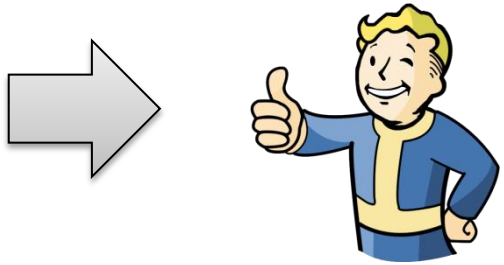
  - Look for semantically equivalent gadgets

```
pop rax # mov rcx, rax # ret
pop rbx # xchg rcx, rbx # ret
mov rax, rsp # sub rcx, 20h # test rcx, rcx # sub rbp, rsi #
mov r10, [rbp] # mov rbx, [rax+68h] # mov rcx, [rax] #
pop r15 # pop r14 # pop r13 # ret
```

ROP Chain Development

TECHNISCHE UNIVERSITÄT DARMSTADT

- Combine gadgets
  - Simple gadgets



  - Complex gadgets, however…

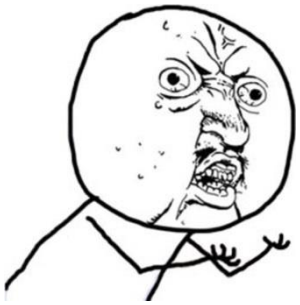
ROP Chain Development

TECHNISCHE UNIVERSITÄT DARMSTADT

- Simple gadgets
  - Too easy, no fun :,(
- Complex gadgets
  - Too tedious, no fun :,(

Creating gadget chains is never fun!

- Why don‘t we automate this whole process?



Automate ROP Chaining

TECHNISCHE UNIVERSITÄT DARMSTADT

- PSHAPE (**P**ractical **S**upport for **H**alf-**A**utomated **P**rogram **E**xploitation)

- Simple gadgets
  - Just find them and put them together

- Complex gadgets
  - Semantic summaries
  - Smart permutations

Semantic Summaries

TECHNISCHE UNIVERSITÄT DARMSTADT

- Need to find gadgets that achieve a certain task
- We do not care *how* that gadgets achieves it
  - E.g., task: ‘‘dereference rsp and load the data in rcx“

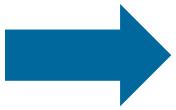
```
pop rcx # ret
pop rcx # pop rax # pop rbp # pop r10 # ret
mov rcx, [rsp] # add rsp, 20h # ret
pop rax # xchg rcx, rax # mov rsi, rdi # ret
pop rax # mov rdx, rax # mov rcx, rdx # ret
```

- Convert ASM to an intermediate representation (IR)
- Propagate read and write statements forward
- Flatten

Semantic Summaries

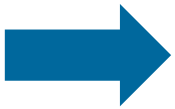
TECHNISCHE UNIVERSITÄT DARMSTADT

```
pop rcx
ret
```



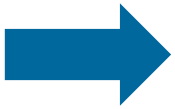
```
rcx = [rsp]
rsp = rsp + 16
```

```
mov rcx, [rsp]
add rsp, 8
ret
```

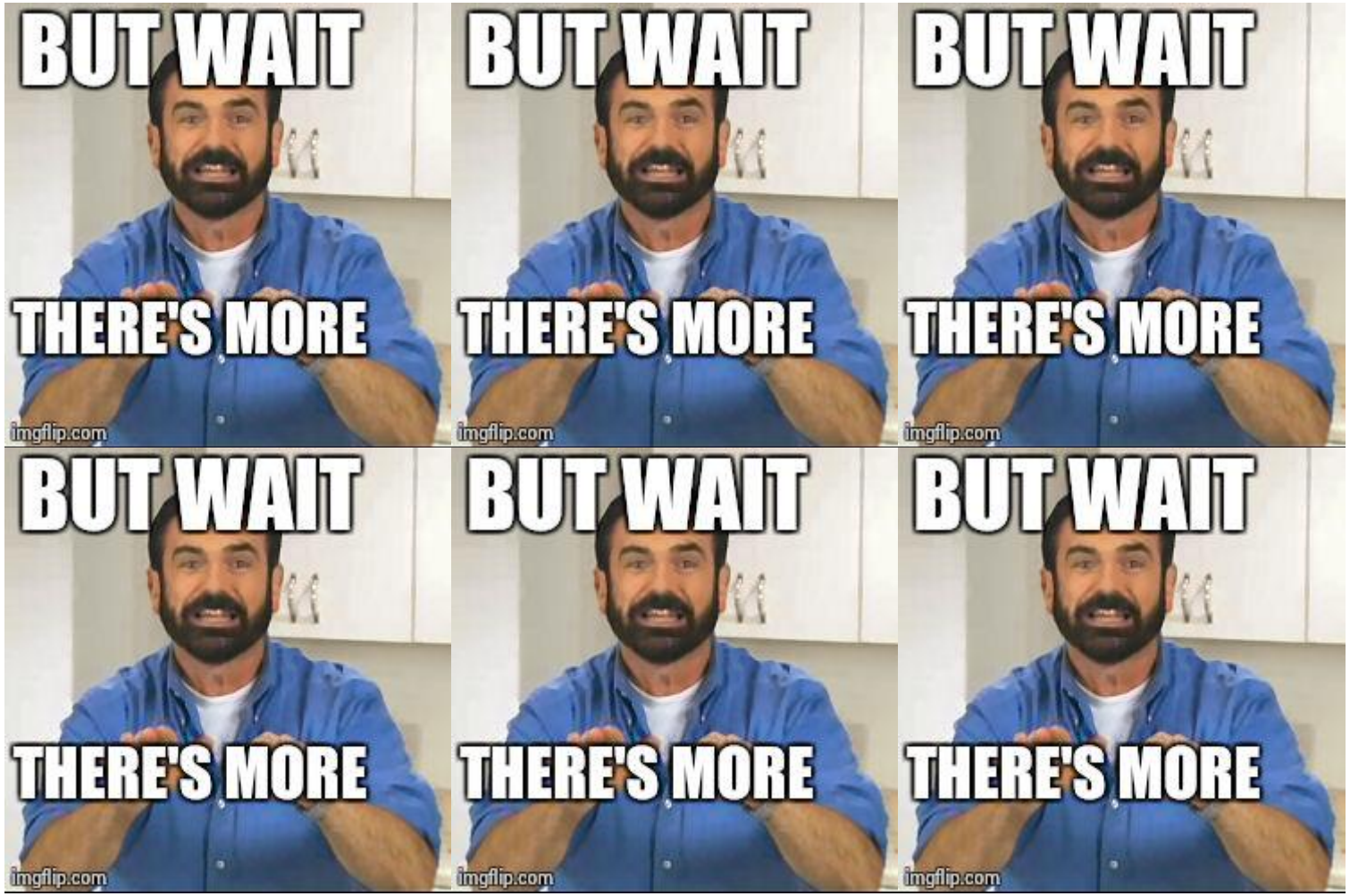


```
rcx = [rsp]
rsp = rsp + 16
```

```
pop rax
xchg rax, rcx
ret
```



```
rax = rcx
rcx = [rsp]
rsp = rsp + 16
```


BUT WAIT
THERE'S MORE
imgflip.com
BUT WAIT
THERE'S MORE
imgflip.com
BUT WAIT
THERE'S MORE
imgflip.com
BUT WAIT
THERE'S MORE
imgflip.com
BUT WAIT
THERE'S MORE
imgflip.com
BUT WAIT
THERE'S MORE
imgflip.com

Semantic Summaries

```
pop rcx
mov rax, [rsi]
ret
```

```
POST: rcx = [rsp]
POST: rax = [rsi]
POST: rsp = rsp + 16
PRE: [rsi]
```

```
mov rax, rsp
mov [rax + 20h], r9
mov [rax + 18h], r8
mov [rax + 10h], rdx
mov [rax + 8], rcx
mov rcx, r9
mov rax, [rcx]
inc rax
mov [rcx + 8], rax
mov rax, [rcx + 4]
inc rax
mov [rcx + 0Ch], rax
ret
```

```
POST: rsp = rsp + 8
POST: rax = [r9 + 4] + 1
POST: rcx = r9

PRE: [r9] <-> [r9 + 0Ch]
PRE: [rsp] <-> [rsp + 20h]
```

Semantic Summaries

TECHNISCHE UNIVERSITÄT DARMSTADT

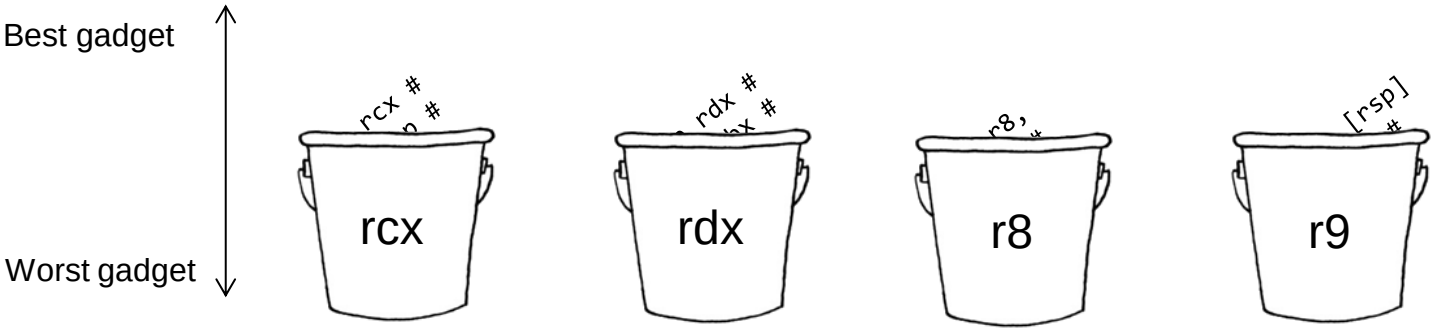
- Summary *what* a gadget does to the system state
- No information about *how* it does it

- Good representation to quickly and easily assess a gadget‘s usefulness
- Helps finding the right gadgets

Smart Permutations

TECHNISCHE UNIVERSITÄT DARMSTADT

- Now we have lots of gadget candidates
- Can‘t just randomly start combining
- Grade gadgets to find the „best“ gadget(s) for each register
- „Best“ is determined mostly by severeness and number of side-effects and preconditions



DEEPSEC

Smart Permutations

TECHNISCHE UNIVERSITÄT DARMSTADT

- Take best gadget for each register, start creating permutations
- Example:

```
pop rcx # ret
pop rdx # mov rcx, 0 # ret
pop r8 # mov rdi, [rbp+ffaah] # xchg rcx, rax # ret
pop r9 # mov r8, rsp # pop r15 # ret
```

- Working order:

```
pop r9 # mov r8, rsp # pop r15 # ret
pop rdx # mov rcx, 0 # ret
pop r8 # mov rdi, [rbp+ffaah] # xchg rcx, rax # ret
pop rcx # ret
```

Smart Permutations

TECHNISCHE UNIVERSITÄT DARMSTADT

- Satisfy preconditions

```
pop r9 # mov r8, rsp # pop r15 # ret
pop rdx # mov rcx, 0 # ret
pop r8 # mov rdi, [rbp+ffaah] # xchg rcx, rax # ret
pop rcx # ret
```

- Prepend, e.g., `pop rbp # ret`

Example of a Chain (nginx)

TECHNISCHE UNIVERSITÄT DARMSTADT

0x412dab	**pop** rax ; **add** rsp, 8 ; **ret** ;
0x45d594	**pop** rbx ; **ret** ;
0x406c20	**pop** rdi ; **ret** ;
0x42892b	**pop** rsi ; **ret** ;
0x425242	**pop** rcx ; **ret** ;
0x444965	**pop** r8 ; **mov qword ptr** [rax], rbx ; **mov** rax, **qword ptr** [rsp + 8] ; **mov qword ptr** [rbx + 0x28], rax ; **mov** rax, **qword ptr** [rsp + 0x18] ; **mov qword ptr** [rbx + 0x18], rax ; **mov edx**, 0 ; **mov** rax, rdx ; **add** rsp, 0x58 ; **pop** rbx ; **pop** rbp ; **pop** r12 ; **pop** r13 ; **pop** r14 ; **pop** r15 ; **ret** ;
0x45a8c4	**pop** rdx ; **ret** ;
0x424219	**mov** r9, **qword ptr** [rsp + 0x28] ; **mov qword ptr** [rbx + 0x48], r9 ; **mov** r10, **qword ptr** [rsp + 0x30] ; **mov qword ptr** [rbx + 0x50], r10 ; **mov** r11, **qword ptr** [rsp + 0x38] ; **mov qword ptr** [rbx + 0x58], r11 ; **add** rsp, 0x48 ; **pop** rbx ; **pop** rbp ; **pop** r12 ; **pop** r13 ; **pop** r14 ; **pop** r15 ; **ret** ;

Related Work

TECHNISCHE UNIVERSITÄT DARMSTADT

- Obviously others had this idea, too
- Plethora of tools which can also create gadget chains…
- … or can they.

Related Work

TECHNISCHE UNIVERSITÄT DARMSTADT

Tool	Search	Auto-chaining	Semantic Summaries	PE/ELF	64-bit
PSHAPE					
nrop					
ROPC					
DEPLib					
Agafi					
mona.py					
ROPgadget					
rp++					
Ropeme					
ropper					
MSFrop					

Related Work

TECHNISCHE UNIVERSITÄT DARMSTADT

Tool	Search	Auto-chaining	Semantic Summaries	PE/ELF	64-bit
PSHAPE	Sem.	Y	Y	Both	Y
nrop	Sem.	N	N	Both	Y
ROPC 	Sem.	Y	N	Both	Y
DEPLib	Sem.	Y	N	PE	N
Agafi	Synt.	Y	N	PE	N
mona.py	Synt.	Y	N	PE	N
ROPgadget 	Synt.	Y	N	Both	Y
rp++	Synt.	N	N	Both	Y
Ropeme	Synt.	N	N	ELF	N
ropper 	Synt.	Y	N	Both	Y
MSFrop	Synt.	N	N	Both	N

ROPC developer: *„ROPC is a POC, and should be treated as such – don't expect it to work on apps from /usr/bin."*

„But what about ASLR?“

TECHNISCHE UNIVERSITÄT DARMSTADT

- Automated bypass not possible (yet)

Attack	Requirements
Info leaks	Info leak vulnerability
Blind ROP	Stack-based overflow, forking application
CAIN	Process inside a VM
Leakless	Non-PIE ELF binary
JIT ROP	Code pointer, attacker-controlled scripting environment

„Okay, how about other Mitigations?“

TECHNISCHE UNIVERSITÄT DARMSTADT

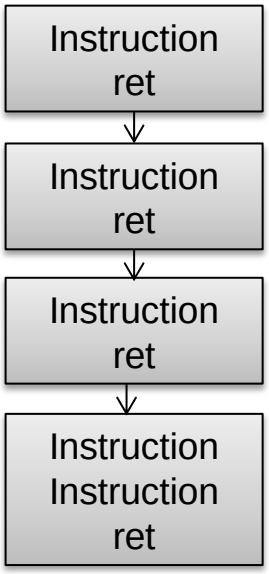
Currently, lots of research on mitigation techniques

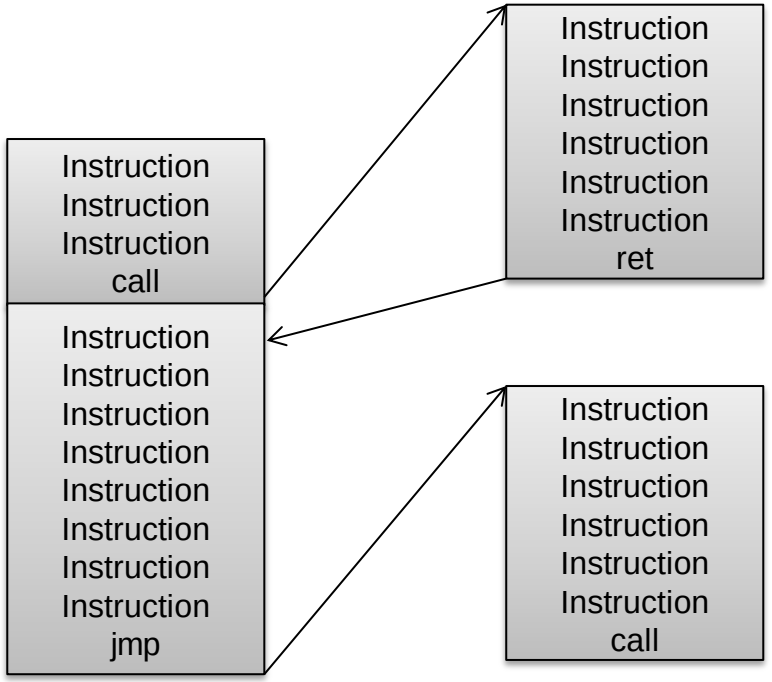
- Shadow stack
- Bounds checking
- Execute-only memory
- Control-flow integrity
- Heuristic-based ROP detectors
- Heap protection
- Use-after-free mitigation
- vtable hardening
- Address space randomization

Heuristic-based ROP detectors

TECHNISCHE UNIVERSITÄT DARMSTADT

- ROP behaviour is very different from „normal“ program behaviour
  - Short basic-blocks
  - Many consecutive indirect branches





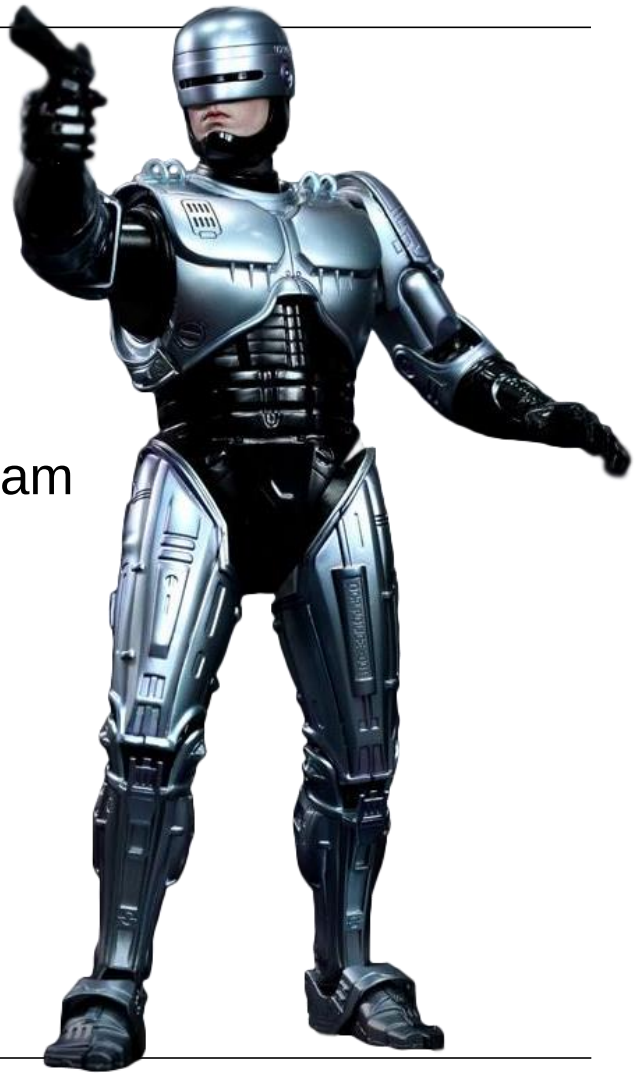
DEEPSEC

ROPocop

TECHNISCHE UNIVERSITÄT DARMSTADT

- Monitors program execution
- Analyses two parameters
  - Length of basic blocks
  - Number of consecutive indirect branches
- Suggests appropriate thresholds for every program
- Implemented with PIN (dynamic binary instrumentation framework by Intel)
- Reliably detects real ROP exploits

- Similar approaches use the same metrics



Bypassing Heuristic Mitigations

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bypass by using only gadgets of a minimum length
  - E.g., use only gadgets of length 5+

```
0x4162f     pop rax ; add al, 0x5b ; pop rbp ; pop r12
            ; ret ;
0x781c4     pop rdi ; pop r8 ; add rsp, 0x18 ; pop rbx
            ; pop rbp ; pop r12 ; pop r13 ; ret ;
0xe7c1      pop rsi ; add byte ptr [rax], al ; adc
            dword ptr [rax], eax ; sbb byte ptr [rax],
            al ; ret 0x29 ;
0x7800d     pop rcx ; mov eax, ebp ; add rsp, 8 ; pop
            rbx ; pop rbp ; ret ;
0x41b13     pop rdx ; pop rbx ; pop rbp ; pop r12 ; ret
            ;
0x7800c     pop r9 ; mov eax, ebp ; add rsp, 8 ; pop
            rbx ; pop rbp ; ret ;
```

Bypassing Heuristic Mitigations

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bypass by using heuristic-breakers
  - Very long (> 25 instructions) gadgets
  - Clobber as few registers as possible

```
nop
nop
add   [rax], al
add   [rax - 0x77], cl
nop
test  al, 0
add   [rax], al
mov   [rax + 0xe8], rdx
mov   [rax + 8], rdx
mov   [rax + 0x10], rdx
mov   [rax + 0x18], rdx
mov   [rax + 0x28], rdx
mov   [rax + 0x30], rdx
mov   [rax + 0x50], rdx
mov   [rax + 0x58], rdx
mov   [rax + 0x60], rdx
mov   [rax + 0x98], rdx
mov   [rax + 0xa0], rdx
…
```

```
…
mov   [rax + 0xf0], rdx
mov   [rax + 0xf8], rdx
mov   [rax + 0x128], rdx
mov   [rax + 0x130], rdx
mov   [rax + 0x120], rdx
mov   [rax + 0x110], rdx
mov   [rax + 0x118], rdx
mov   [rax + 0xd8], rdx
mov   [rax + 0xe0], rdx
mov   [rax + 0xb0], rdx
mov   [rax + 0xb8], 0
mov   [rax + 0x190], rdx
mov   [rax + 0x1c0], rdx
mov   [rax + 0x150], 7
mov   [rax + 0x158], 0x46dd77
add   rsp, 8
ret
```


DEEPSEC

Control-Flow Integrity (CFI)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Compute control-flow graph (CFG) ahead of time
- Enforce CFG at runtime

- Examplary CFI policy
  - `ret` must return to the caller (enforced through a shadow stack)
  - `jmp` must stay within the current function
  - `call` must target a legal function as determined by the CFG

- Large-ish performance impact
- Requires source-code

Control-Flow Integrity (CFI)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Examplary, coarse-grained CFI policy
  - `ret` must return to a call-preceded instruction
  - `jmp` must stay within the current function
  - `call` must target the beginning of a function

- Faster
- Does not rely on source-code
- Less secure
- Used by many implementations

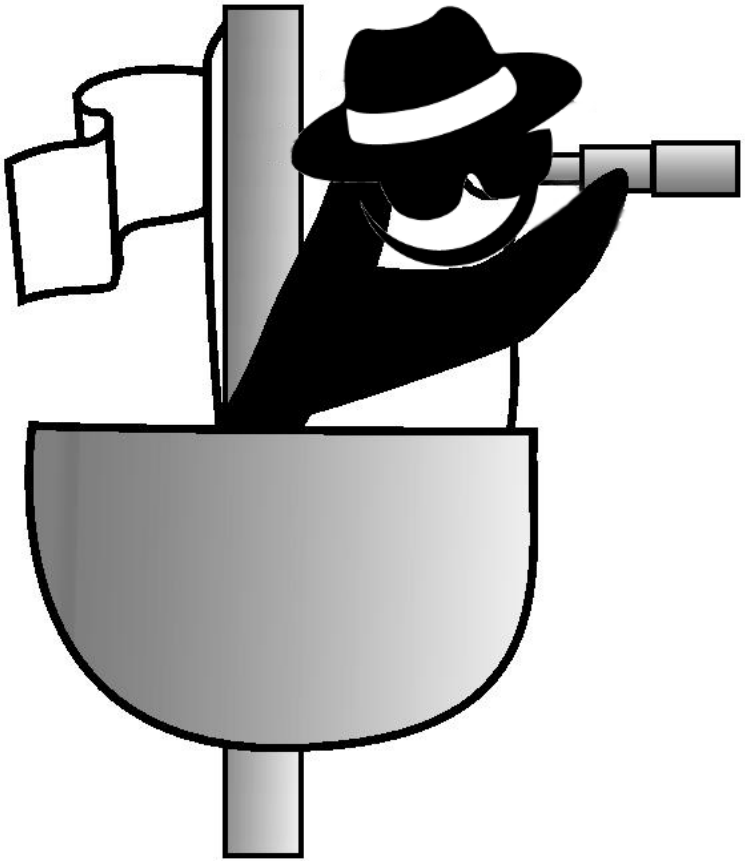
Bypassing CFI

- Use only call-preceded gadgets
- Leaves ~3.3% of gadgets usable
- Still enough gadgets to succeed
- We came across a very interestig gadget…

Bypassing CFI

TECHNISCHE UNIVERSITÄT DARMSTADT

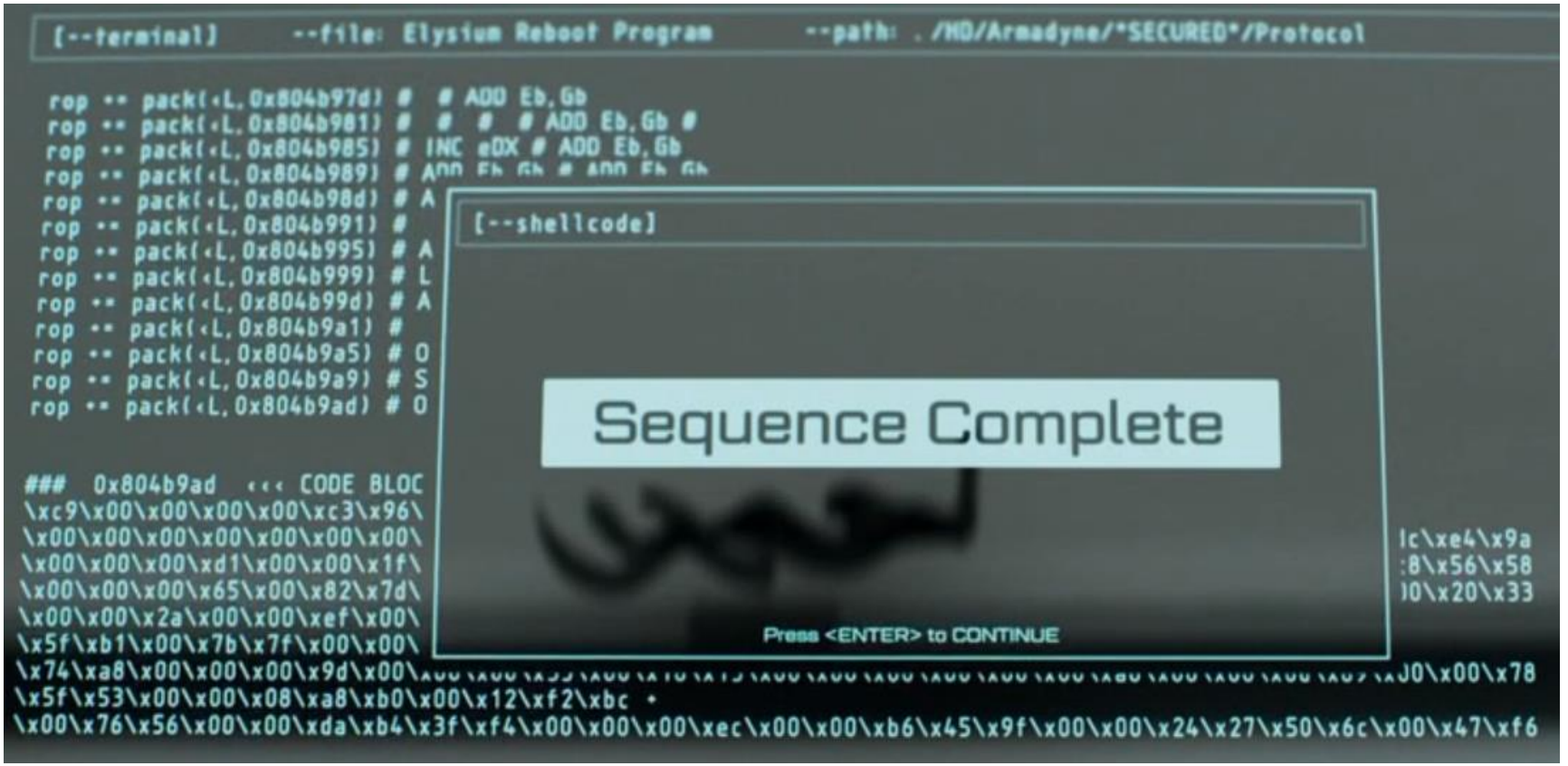
```
mov qword ptr [rbx], r15 ; mov qword ptr [rbx + 8], r14 ; mov qword ptr [rbx
+ 0x10], r13 ; mov qword ptr [rbx + 0x18], r12 ; mov rcx, qword ptr [rsp] ;
mov qword ptr [rbx + 0x20], rcx ; mov rsi, qword ptr [rsp + 8] ; mov qword
ptr [rbx + 0x28], rsi ; mov rdx, qword ptr [rsp + 0x10] ; mov qword ptr [rbx
+ 0x30], rdx ; mov rdi, qword ptr [rsp + 0x18] ; mov qword ptr [rbx + 0x38],
rdi ; mov r8, qword ptr [rsp + 0x20] ; mov qword ptr [rbx + 0x40], r8 ;
mov r9, qword ptr [rsp + 0x28] ; mov qword ptr [rbx + 0x48], r9 ; mov r10,
qword ptr [rsp + 0x30] ; mov qword ptr [rbx + 0x50], r10 ; mov r11, qword ptr
[rsp + 0x38] ; mov qword ptr [rbx + 0x58], r11 ; add rsp, 0x48 ; pop rbx ;
pop rbp ; pop r12 ; pop r13 ; pop r14 ; pop r15 ; ret  ;
```



What‘s ahead?

ROP in the Future

TECHNISCHE UNIVERSITÄT DARMSTADT

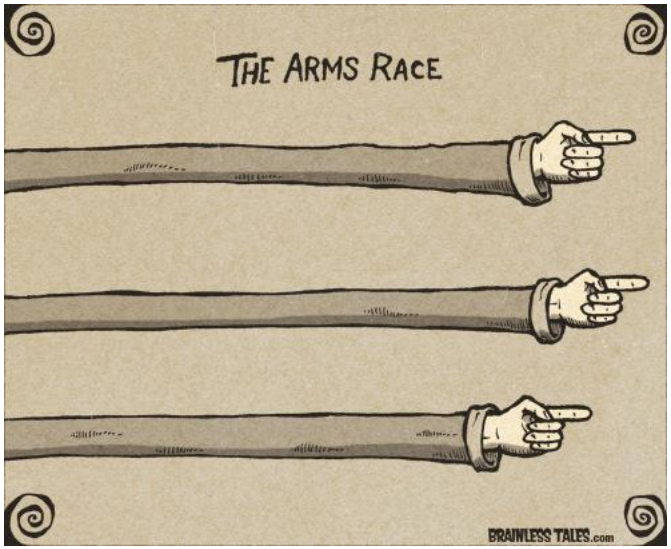


DEEPSEC

Outlook

TECHNISCHE UNIVERSITÄT DARMSTADT

- Mitigations
  - CFI (Intel CET)
  - Fine-grained ASR
  - Execute-only memory
- Attacks
  - Against many of the above
  - ROP Automation?



Thank you!
Q&A