Linux improvements in memory corruption based protections

Vlatko Košturjak – CTO@Diverto

AGENDA

- **Linux**
- **Overflows**
- **Return Oriented Programming**
- **Intel Control-Flow Enforcement Technology (CET)**
- **Indirect Branch Tracking**
- **Shadow Stack**
- **Take away**
- **Summary**
- **Questions and Answers**

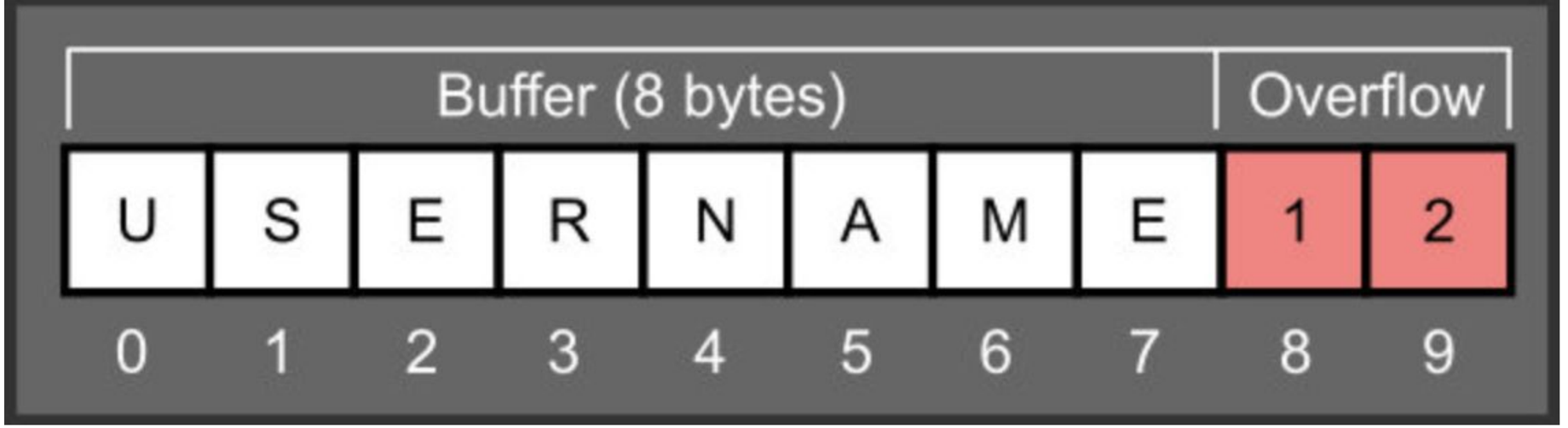
30(40?) minutes

Linux

- **Cloud**
- **Server**
- **Container/Docker/Kubernetes**
- **Phone/Tablet/Readers**
- **Mainframe**
- **Desktop**
- **Embedded/IoT/Pi**
- **Game consoles**
- **...**

Buffer Overflow Intro

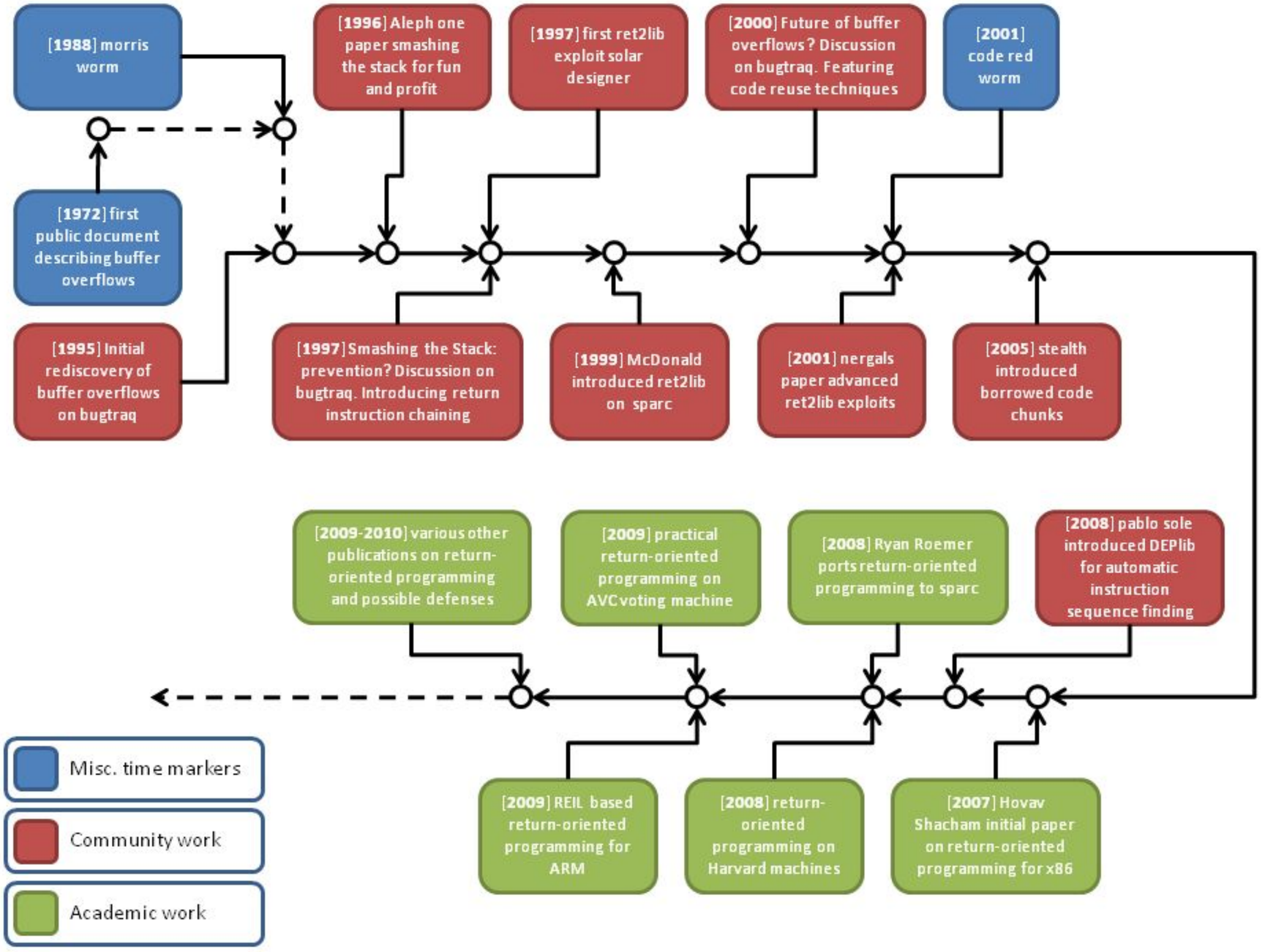
```
void main()
{
    char source[] = "username12"; // username12 to source[]
    char destination[7]; // Destination is 8 bytes
    strcpy(destination, source); // Copy source to destination

    return 0;
}
```



https://www.hackingtutorials.org/exploit-tutorials/buffer-overflow-explained-basics/

Are overflows still relevant?

Roadmap



https://blog.zynamics.com/2010/03/12/a-gentle-introduction-to-return-oriented-programming/

Road to ROP

Morris Worm

- fingerd
- 1988

Smashing Stack

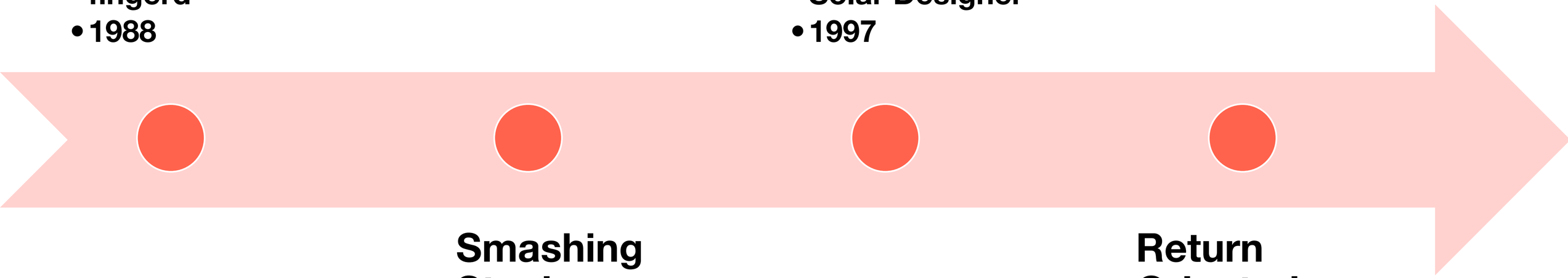
- Aleph One
- 1996

ret2libc

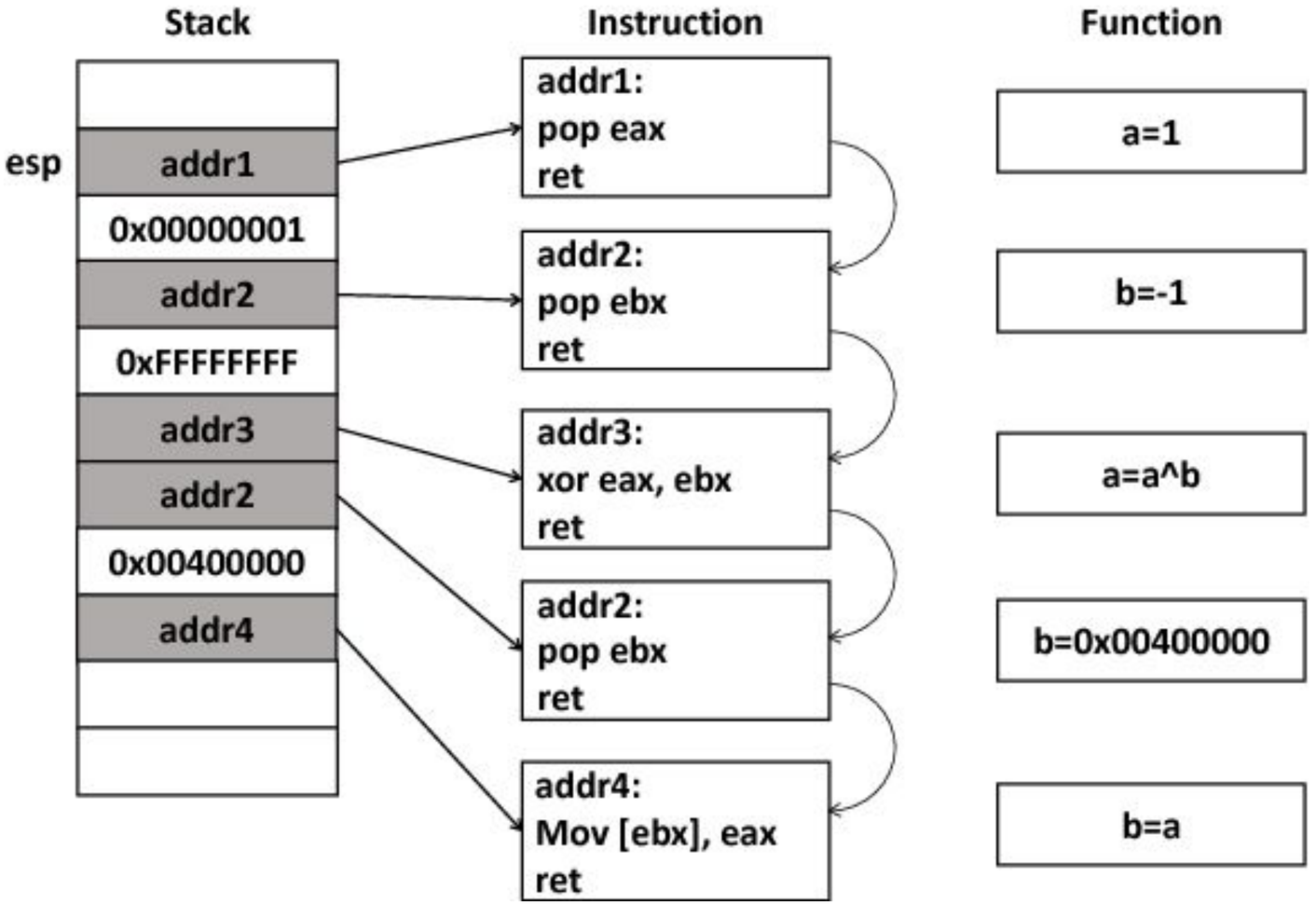
- Solar Designer
- 1997

Return Oriented Programming

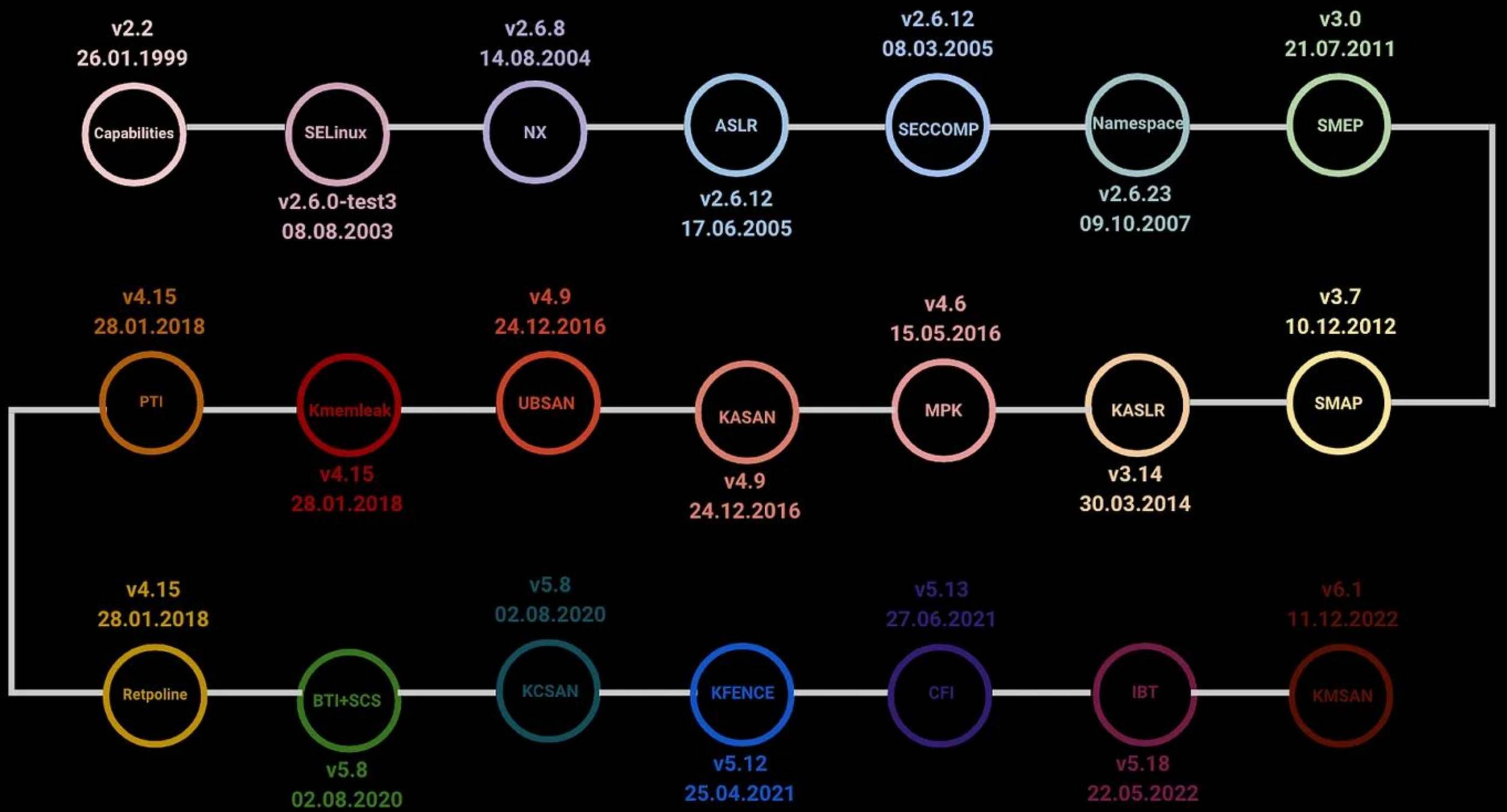
- Hovav Shacham
- 2007



Return Oriented Programming Intro

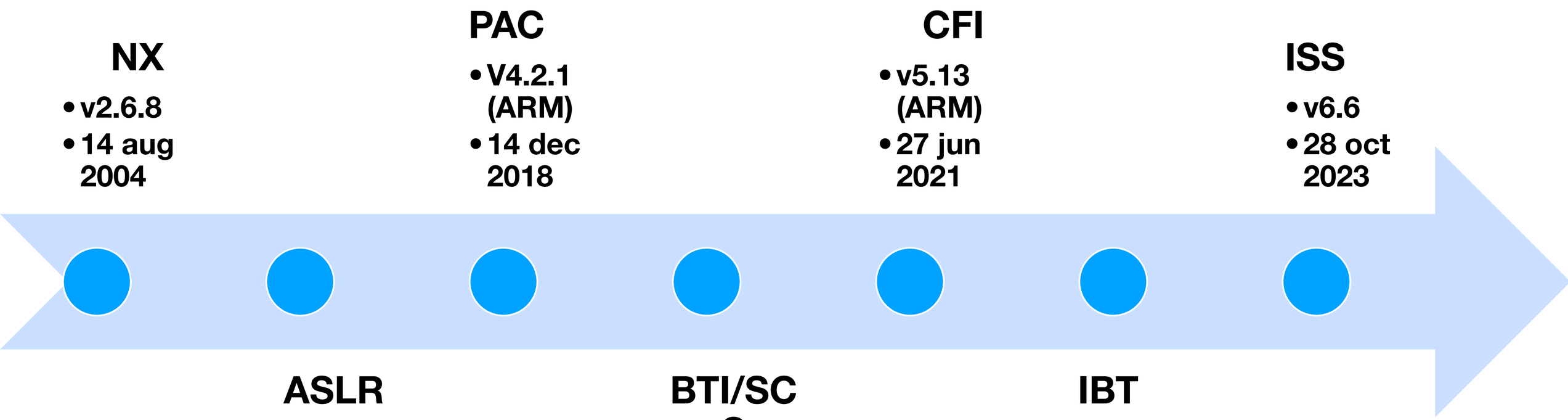


https://www.semanticscholar.org/paper/A-Survey-of-Return-Oriented-Programming-Attack%2C-and-Wang-Xie/7b9a465ce2e417507a7b0e338f4a83434a1fa49e

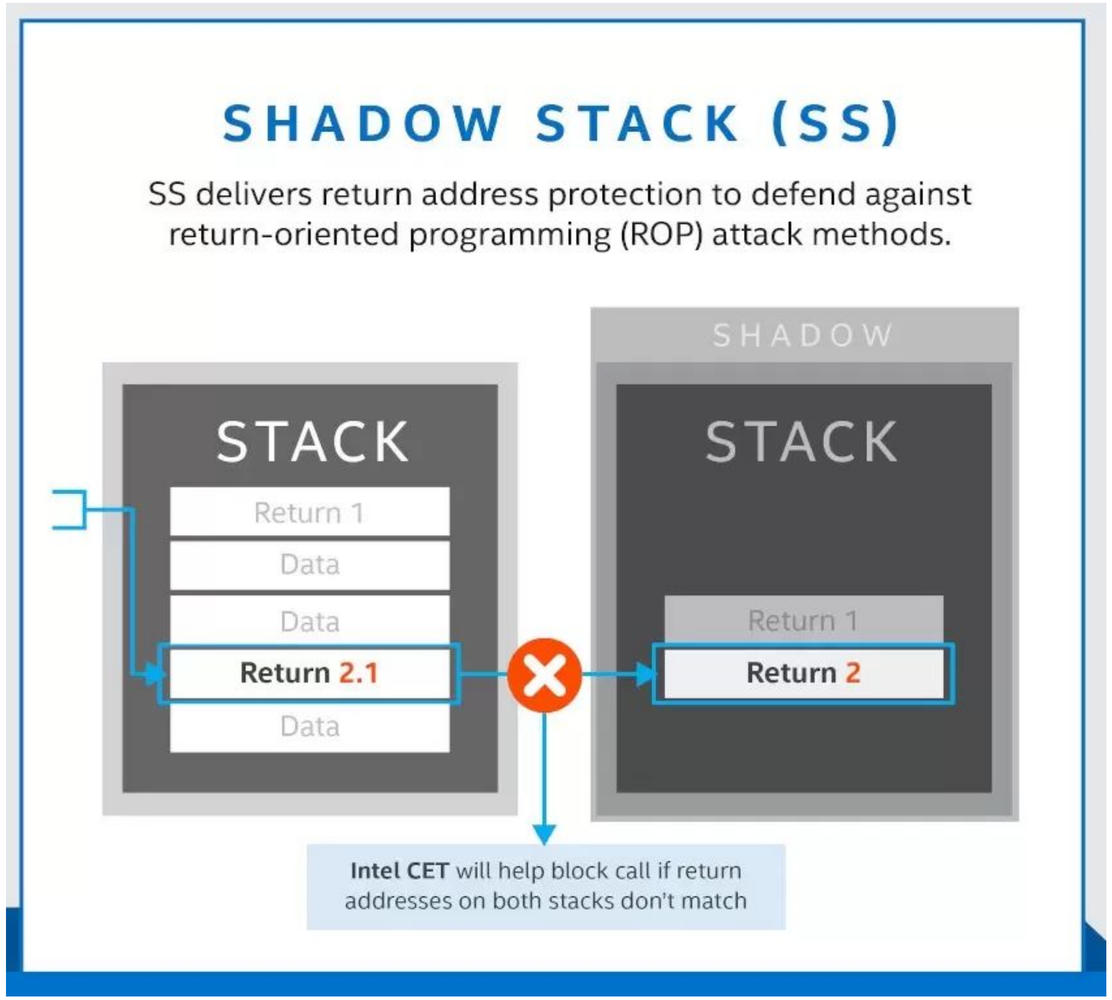


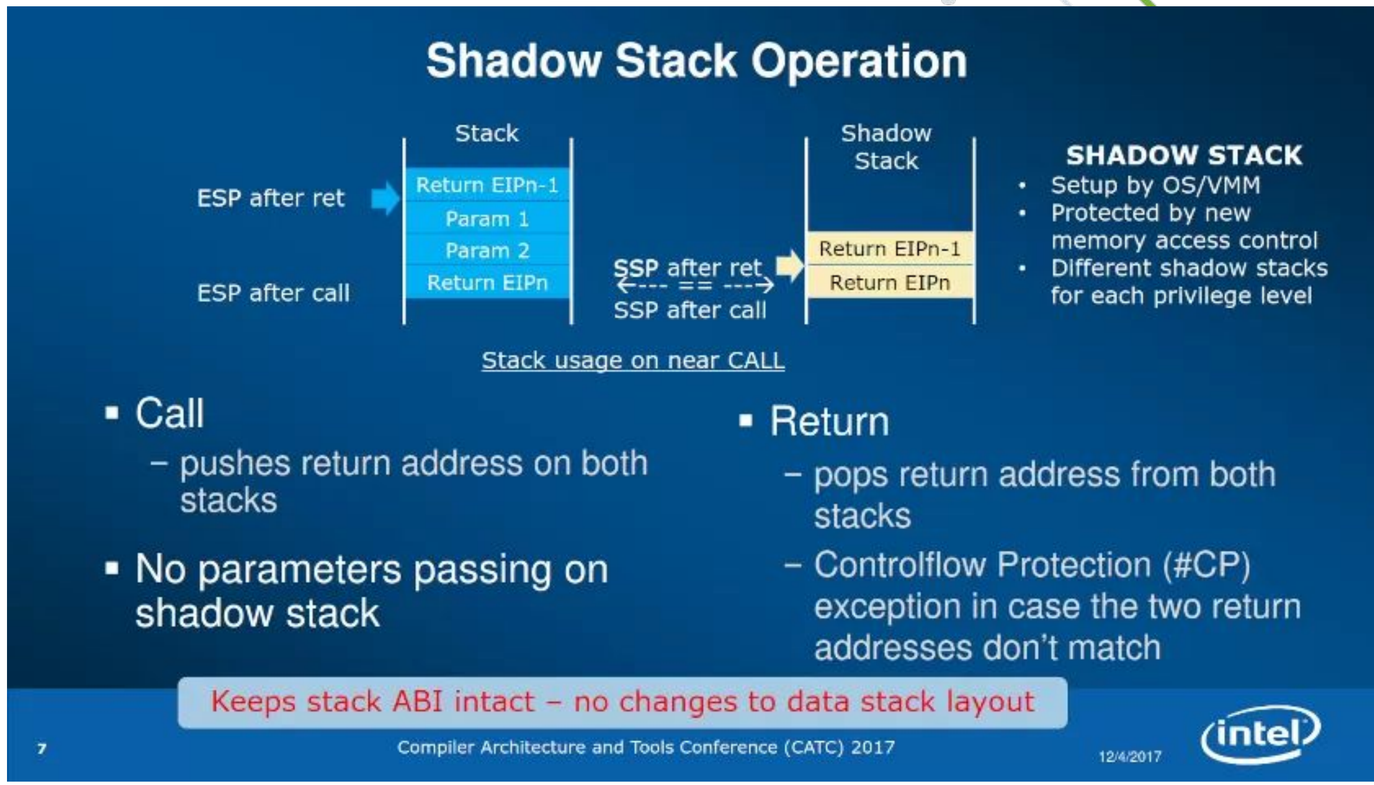
https://medium.com/@ofriouzan/part-3-kernel-level-security-mechanisms-097e8b8ecefa

Relevant Linux Security Mechanisms



(Intel) Shadow Stack





https://www.phoronix.com/news/Linux-6.4-Shadow-Stack-Coming

Shadow stack - clang

```
int foo() {
  return bar() + 1;
}
```

```
push   %rax
callq  bar
add    $0x1,%eax
pop    %rcx
retq
```

-fsanitize=shadow-call-stack

```
mov    (%rsp),%r10
xor    %r11,%r11
addq   $0x8,%gs:(%r11)
mov    %gs:(%r11),%r11
mov    %r10,%gs:(%r11)
push   %rax
callq  bar
add    $0x1,%eax
pop    %rcx
xor    %r11,%r11
mov    %gs:(%r11),%r10
mov    %gs:(%r10),%r10
subq   $0x8,%gs:(%r11)
cmp    %r10,(%rsp)
jne    trap
retq

trap:
ud2
```

https://releases.llvm.org/7.0.1/tools/clang/docs/ShadowCallStack.html

Linux and Intel Shadow Stack

- Hardware
  - CPU with support for Shadow Stack
- Software
  - Kernel
  - Loader/(G)Libc
  - Binary
    - Usually compiler which can produce binary with Shadow Stack support

ShadowCallStack on x86_64 suffered from the same racy security issues as Return Flow Guard and had performance overhead as high as 13% depending on the benchmark.

x86_64 ShadowCallStack was always an experimental feature and never shipped a runtime required to support it, as such there are no expected downstream users.

https://github.com/llvm-mirror/llvm/commit/863ea8c618b1f88ba8c9ec355a07cb3783481642

Linux and Intel Shadow Stack

- Hardware
  - CPU with support for Shadow Stack
- Software
  - Kernel
  - Loader/(G)Libc
  - Binary
    - Usually compiler which can produce binary with Shadow Stack support

Hardware: CPU

- Hardware
  - CPU with support for Shadow Stack
- Intel® Control-Flow Enforcement Technology
  - Tigerlake CPUs and newer
  - CPUs built 2020 and newer
  - Support
    - indirect branch tracking (IBT)
    - shadow stack (SS)

Hardware: CPU – how to check?

```
# apt install cpuid
$ cpuid -1 -i | grep -E 'CET_[SIBT]{2,3}'
      CET_SS: CET shadow stack                       = false
      CET_IBT: CET indirect branch tracking          = false

$ cpuid -1 -i | grep -E 'CET_[SIBT]{2,3}'
      CET_SS: CET shadow stack                       = true
      CET_IBT: CET indirect branch tracking          = true
```

Hardware: CPU – check ?

8.2 Feature Enumeration

CET shadow stacks feature flag - if CPUID.(EAX=7, ECX=0):ECX.CET_SS[bit 7] is 1, the processor supports CET shadow stack features, including the MSR described in section 9.5.

CET indirect branch tracking feature flag - if CPUID.(EAX=7, ECX=0):EDX.CET_IBT[bit 20] is 1, the processor supports CET indirect branch tracking, including the MSR described in section 9.5.

```
#include <cpuid.h>
#include <stdint.h>

int cpu_supports_cet_shadow_stack() {
    uint32_t eax = 0, ebx = 0, ecx = 0, edx = 0;
    __cpuid_count(7, 0, eax, ebx, ecx, edx);
    return (ecx & (1 << 7)) != 0;
}

int cpu_supports_cet_ibt() {
    uint32_t eax = 0, ebx = 0, ecx = 0, edx = 0;
    __cpuid_count(7, 0, eax, ebx, ecx, edx);
    return (edx & (1 << 20)) != 0;
}
```

why not /proc/cpuinfo ?

cpuinfo on same machine

```
processor: 7
vendor_id: GenuineIntel
cpu family  : 6
model    : 140
model name  : 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz
stepping : 1
microcode: 0x6c
cpu MHz     : 2566.733
cache size  : 12288 KB
physical id : 0
siblings : 8
core id     : 3
cpu cores: 4
apicid   : 7
initial apicid : 7
fpu   : yes
fpu_exception  : yes
cpuid level : 27
wp    : yes
flags    : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx
pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl
xtopology nonstop_tsc cpuid aperfmperf tsc_known_freq pni pclmulqdq dtes64
monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1
sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand
lahf_lm abm 3dnowprefetch cpuid_fault epb cat_l2 cdp_l2 ssbd ibrs ibpb
stibp ibrs_enhanced tpr_shadow flexpriority ept vpid ept_ad fsgsbase
tsc_adjust bmi1 avx2 smep bmi2 erms invpcid rdt_a avx512f avx512dq rdseed
adx smap avx512ifma clflushopt clwb intel_pt avx512cd sha_ni avx512bw
avx512vl xsaveopt xsavec xgetbv1 xsaves split_lock_detect dtherm ida arat
pln pts hwp hwp_notify hwp_act_window hwp_epp hwp_pkg_req vnmi avx512vbmi
umip pku ospke avx512_vbmi2 gfni vaes vpclmulqdq avx512_vnni avx512_bitalg
avx512_vpopcntdq rdpid movdiri movdir64b fsrm avx512_vp2intersect md_clear
ibt flush_l1d arch_capabilities
vmx flags: vnmi preemption_timer posted_intr invvpid ept_x_only ept_ad
ept_1gb flexpriority apicv tsc_offset vtpr mtf vapic ept vpid
unrestricted_guest vapic_reg vid ple pml ept_mode_based_exec tsc_scaling
bugs     : spectre_v1 spectre_v2 spec_store_bypass swapgs eibrs_pbrsb gds
```

```
processor: 7
vendor_id: GenuineIntel
cpu family  : 6
model    : 140
model name  : 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz
stepping : 1
microcode: 0x6c
cpu MHz     : 2566.733
cache size  : 12288 KB
physical id : 0
siblings : 8
core id     : 3
cpu cores: 4
apicid   : 7
initial apicid : 7
fpu   : yes
fpu_exception  : yes
cpuid level : 27
wp    : yes
flags    : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe
syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs
bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf
tsc_known_freq pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2
ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe
popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm
3dnowprefetch cpuid_fault epb cat_l2 cdp_l2 ssbd ibrs ibpb stibp
ibrs_enhanced tpr_shadow flexpriority ept vpid ept_ad fsgsbase
tsc_adjust bmi1 avx2 smep bmi2 erms invpcid rdt_a avx512f avx512dq
rdseed adx smap avx512ifma clflushopt clwb intel_pt avx512cd
sha_ni avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves
split_lock_detect user_shstk dtherm ida arat pln pts hwp
hwp_notify hwp_act_window hwp_epp hwp_pkg_req vnmi avx512vbmi umip
pku ospke avx512_vbmi2 gfni vaes vpclmulqdq avx512_vnni
avx512_bitalg avx512_vpopcntdq rdpid movdiri movdir64b fsrm
avx512_vp2intersect md_clear ibt flush_l1d arch_capabilities
vmx flags: vnmi preemption_timer posted_intr invvpid ept_x_only
ept_ad ept_1gb flexpriority apicv tsc_offset vtpr mtf vapic ept
vpid unrestricted_guest vapic_reg vid ple pml ept_mode_based_exec
tsc_scaling
bugs     : spectre_v1 spectre_v2 spec_store_bypass swapgs
eibrs_pbrsb gds
:
```

Software: kernel

- Version
  - 6.6 or higher
- Current Support
  - User space only
- Configuration
  - X86_USER_SHADOW_STACK
  - Prerequisits
    - Binutils v2.29 or
    - LLVM v6
  - /proc/cpuinfo shows CET features (if processor supports it)
    - "user_shstk” in flags means userspace shadow stack

```
processor: 7
vendor_id: GenuineIntel
cpu family  : 6
model    : 140
model name  : 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz
stepping : 1
microcode: 0x6c
cpu MHz     : 2566.733
cache size  : 12288 KB
physical id : 0
siblings : 8
core id     : 3
cpu cores: 4
apicid   : 7
initial apicid : 7
fpu   : yes
fpu_exception  : yes
cpuid level : 27
wp    : yes
flags    : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon
bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf
tsc_known_freq pni pclmulqdq dtes64 monitor ds_cpl
ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4
popcnt tsc_deadline_timer aes xsave avx f  lm abm
3dnowprefetch cpuid_fault epb cat_  ibpb stibp
ibrs_enhanced tpr_shadow flexpriority  pt_ad fsgsbase
tsc_adjust bmi1 avx2 smep bmi2 e  dt_a avx512f avx512dq
rdseed adx smap avx512ifma clflu  wb intel_pt avx512cd
sha_ni avx512bw avx512vl xsaveopt xs  xgetbv1 xsaves
split_lock_detect user_shstk dtherm ida arat pln pts hwp
hwp_notify hwp_act_window hwp_epp hwp_pkg_req vnmi avx512vbmi umip
pku ospke avx512_vbmi2 gfni vaes vpclmulqdq avx512_vnni
avx512_bitalg avx512_vpopcntdq rdpid movdiri movdir64b fsrm
avx512_vp2intersect md_clear ibt flush_l1d arch_capabilities
vmx flags: vnmi preemption_timer posted_intr invvpid ept_x_only
ept_ad ept_1gb flexpriority apicv tsc_offset vtpr mtf vapic ept
vpid unrestricted_guest vapic_reg vid ple pml ept_mode_based_exec
tsc_scaling
bugs     : spectre_v1 spectre_v2 spec_store_bypass swapgs
eibrs_pbrsb gds
:
```

Hardware + Kernel support for shadow stack is working!

Still, we are not there yet!

Software: glibc

- Version
  - 2.39 or higher
    - Released 31th of January, 2024
- Support
  - 64bit only
- Compilation flag
  - --enable-cet
    - Compilation of libc with CET support and protection
    - fails on 32bit build
- Configuration options
- Available at run-time
  - Using standard GLIBC tunables mechanism

Software: glibc – options glibc.cpu.x86_shstk

Value	Description
Off	off always turns off IBT regardless of whether IBT is enabled in the executable and its dependent shared libraries
Permissive	permissive is the same as the default which disables IBT on non-CET executables and shared libraries
On	On always turns on IBT regardless of whether IBT is enabled in the executable and its dependent shared libraries

https://www.gnu.org/software/libc/manual/html_node/Hardware-Capability-Tunables.html

Software: binary

- Need to produce ELF binary with SHSTK flag
  - x86 feature: SHSTK
- Compiler
  - gcc
  - clang
- Compilation flags
  - --mshstk
  - --fcf-protection=full
- Test with
  - $ readelf -n <application> | grep -a SHSTK
  - properties: x86 feature: SHSTK

https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html

```
$./vuln
Hello World
dalkjdlksjalkkkkkkkkkkkkkkkkjkdssssssssssssssssssssssssssssssss
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
sssssssssssssssssssssssssssssssssssssssss
*** stack smashing detected ***: terminated
Aborted
```

Not yet. Hitting other overflow security control. Which one?

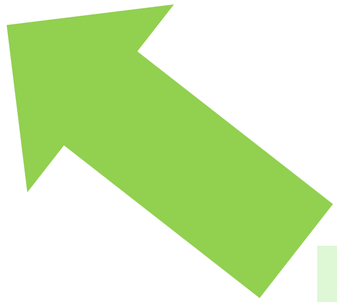
```
# cat /proc/$PID/status  | grep x86_Thread_features
x86_Thread_features:
x86_Thread_features_locked:
```

Not all CET enabled applications and libraries have been properly tested in CET enabled environments. Some CET enabled applications or libraries will crash or misbehave when CET is enabled. Don't set CET active by default so that all applications and libraries will run normally regardless of whether CET is active or not. Shadow stack can be enabled by

```
$ export GLIBC_TUNABLES=glibc.cpu.hwcaps=SHSTK
```

https://sourceware.org/git/?p=glibc.git;a=commit;h=55d63e731253de82e96ed4ddca2e294076cd0bc5

```
$ export GLIBC_TUNABLES=glibc.cpu.hwcaps=SHSTK
$ ./vuln

# cat /proc/$PID/status  | grep x86_Thread_features
x86_Thread_features:    shstk
x86_Thread_features_locked:     shstk wrss
```



YEY! Complete security control of Shadow stack is working!

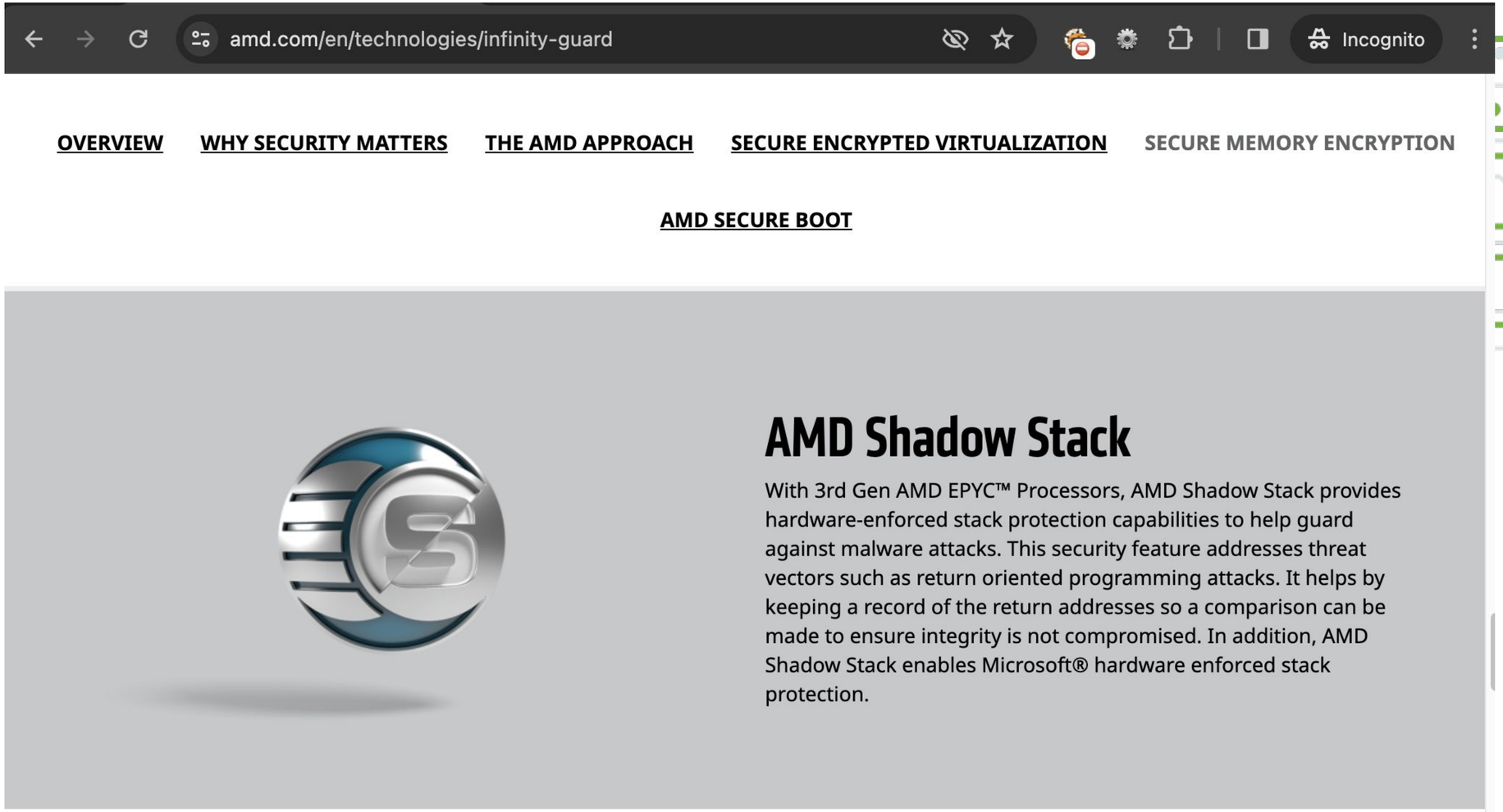
Segmentation fault.

Compatibility

- Legacy applications not getting security for free
  - Work and testing involved
- 64 bit support only
  - 32bit not yet supported
    - Will it ever be?
- Libraries and components
  - All parts should be compiled with shadow stack support
- Virtualization
  - Patches still flying in
  - CI/CD testing limited

https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html

AMD?



„…Thanks, I ran some smoke tests with the updated glibc and it's looking good so far. Additionally, I ran the new kselftest and it passed…” -John Allen (AMD)

https://lore.kernel.org/lkml/Yf2m1ETkcRpk3v+u@dell9853host/

Is this end of the overflows now?

Bypass?

- Find component which is not protected
- Components and complexity
  - Hardware
  - Kernel
  - Glibc
  - Compiler
  - Binary
- Techniques from other OS or architectures
- Implementation details
  - For example: „On exec, shadow stack features are disabled by the kernel. At which point, userspace can choose to re-enable, or lock them.”

Take away

- System administrator
  - If something stops working, where/what to check
  - How to implement yet another overflow security control
- Developer
  - What new compiler flags you should use
  - How you should test your program if it works with ISS controls
- Security Specaliast
  - How to additionaly harden your system
  - How to check if hardening really works
- Security Researcher
  - Fundamentals to get you started

References

- Control-flow Enforcement Technology Specification, Intel, May 2019, Revision 3.0
- Control-flow Enforcement Technology (CET) Shadow Stack
  - https://www.kernel.org/doc/html/next/x86/shstk.html
- Glibc CET branch (before official glibc release)
  - https://gitlab.com/x86-glibc/glibc/-/commits/users/hjl/cet/master

Summary

- Getting shell even if process is vulnerable is getting harder
  - Hard to say impossible
- Shadow stack and Intel/AMD Linux
  - Implementation is now complete
  - Only user space protection
  - Old applications/systems do not get it for free
  - Current limitations: 32bit and virtualization non existant
  - Not yet (by default) in your favourite distribution
- Recommended
  - Fixing at source

Thank you!

www.diverto.hr

kost@diverto.hr

@k0st