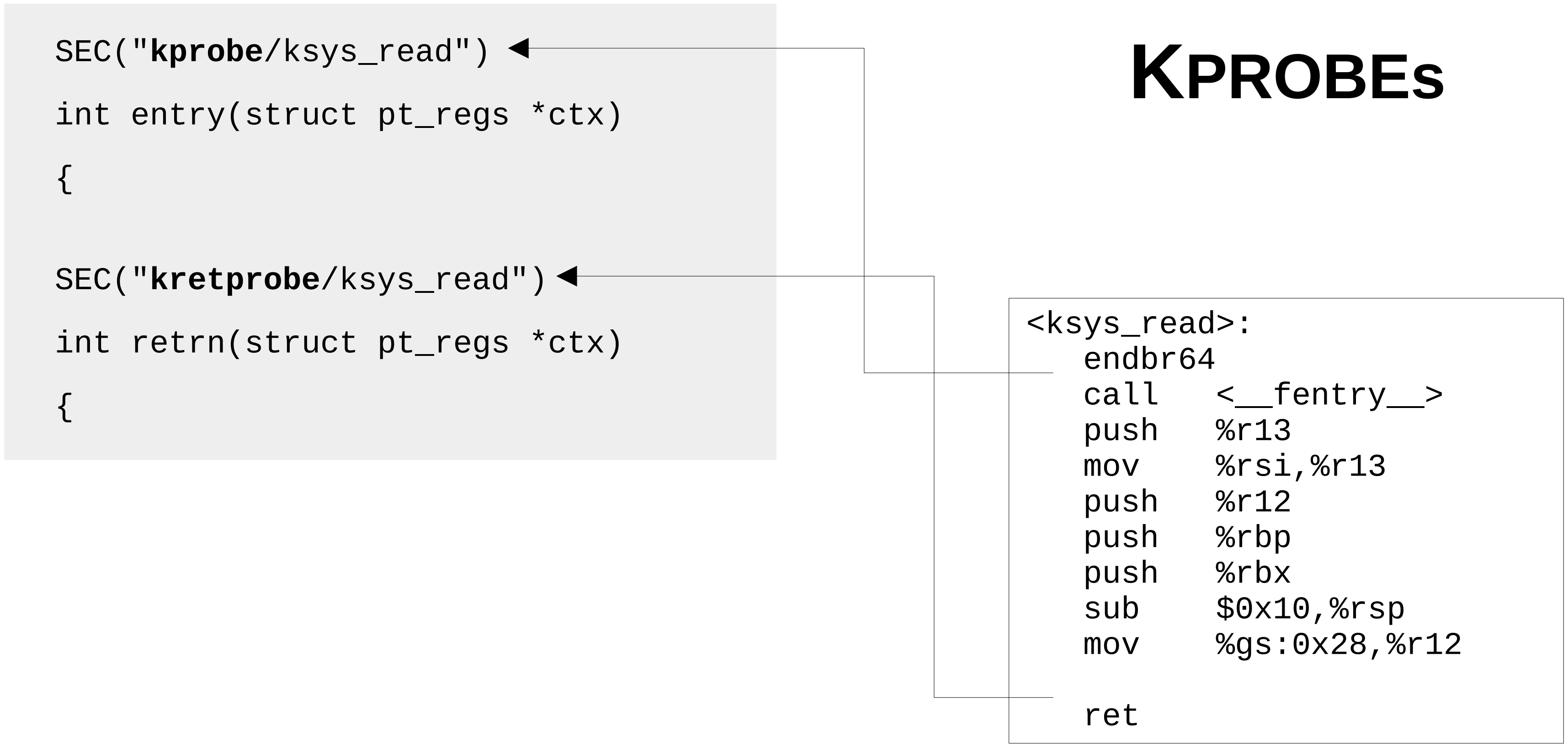
probes update

jiri olsa / isovalent at cisco

SESSION

UPROBES

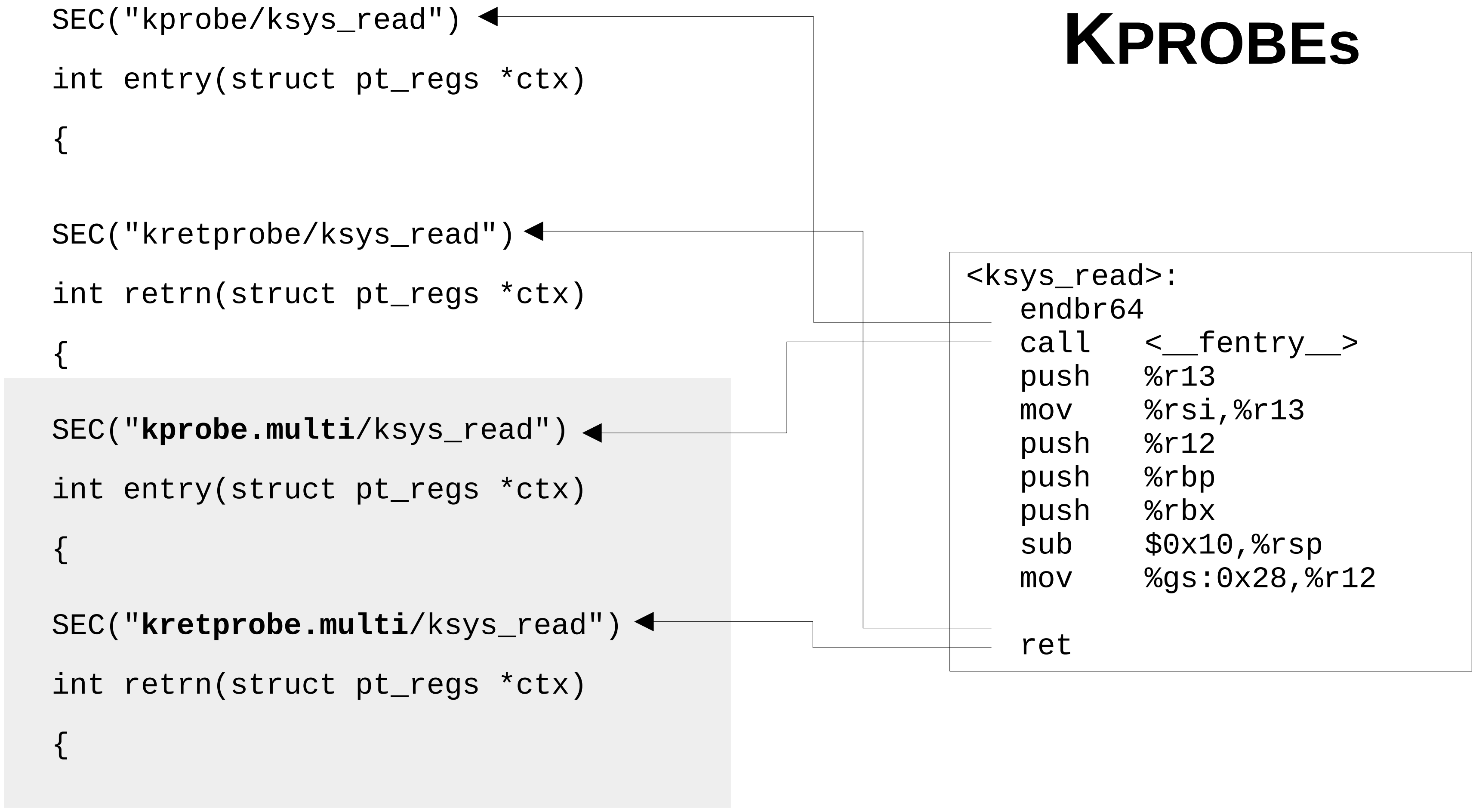
KPROBEs



```
SEC("kprobe/ksys_read")

int entry(struct pt_regs *ctx)

{


SEC("kretprobe/ksys_read")

int retrn(struct pt_regs *ctx)

{
```

```
<ksys_read>:
   endbr64
   call   <__fentry__>
   push   %r13
   mov    %rsi,%r13
   push   %r12
   push   %rbp
   push   %rbx
   sub    $0x10,%rsp
   mov    %gs:0x28,%r12

   ret
```

KPROBEs



```
SEC("kprobe/ksys_read")
int entry(struct pt_regs *ctx)
{
```

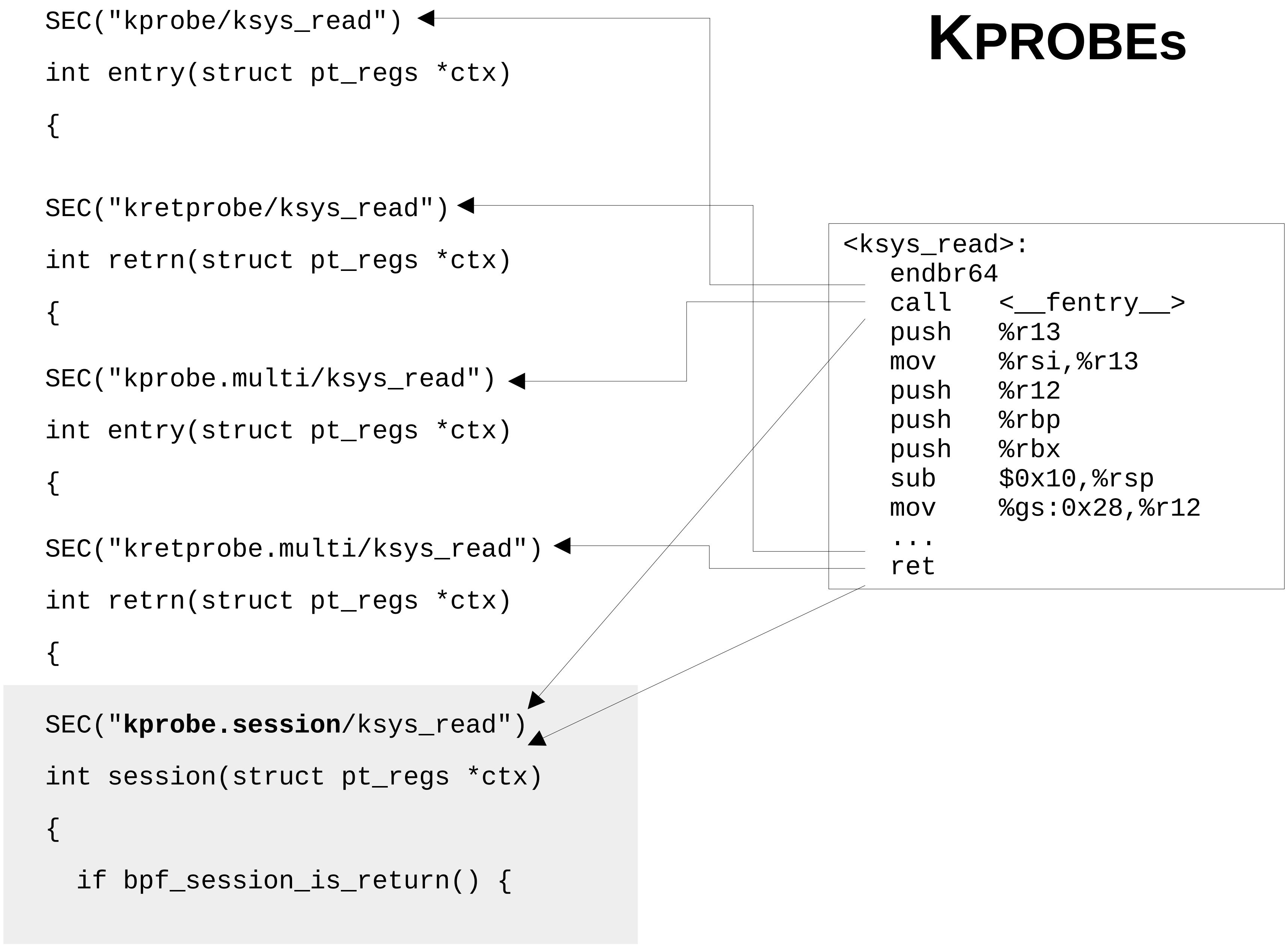
```
SEC("kretprobe/ksys_read")
int retrn(struct pt_regs *ctx)
{
```

SEC("**kprobe.multi**/ksys_read")
`int entry(struct pt_regs *ctx)`
`{`

SEC("**kretprobe.multi**/ksys_read")
`int retrn(struct pt_regs *ctx)`
`{`

```
<ksys_read>:
   endbr64
   call    <__fentry__>
   push    %r13
   mov     %rsi,%r13
   push    %r12
   push    %rbp
   push    %rbx
   sub     $0x10,%rsp
   mov     %gs:0x28,%r12

   ret
```

KPROBEs

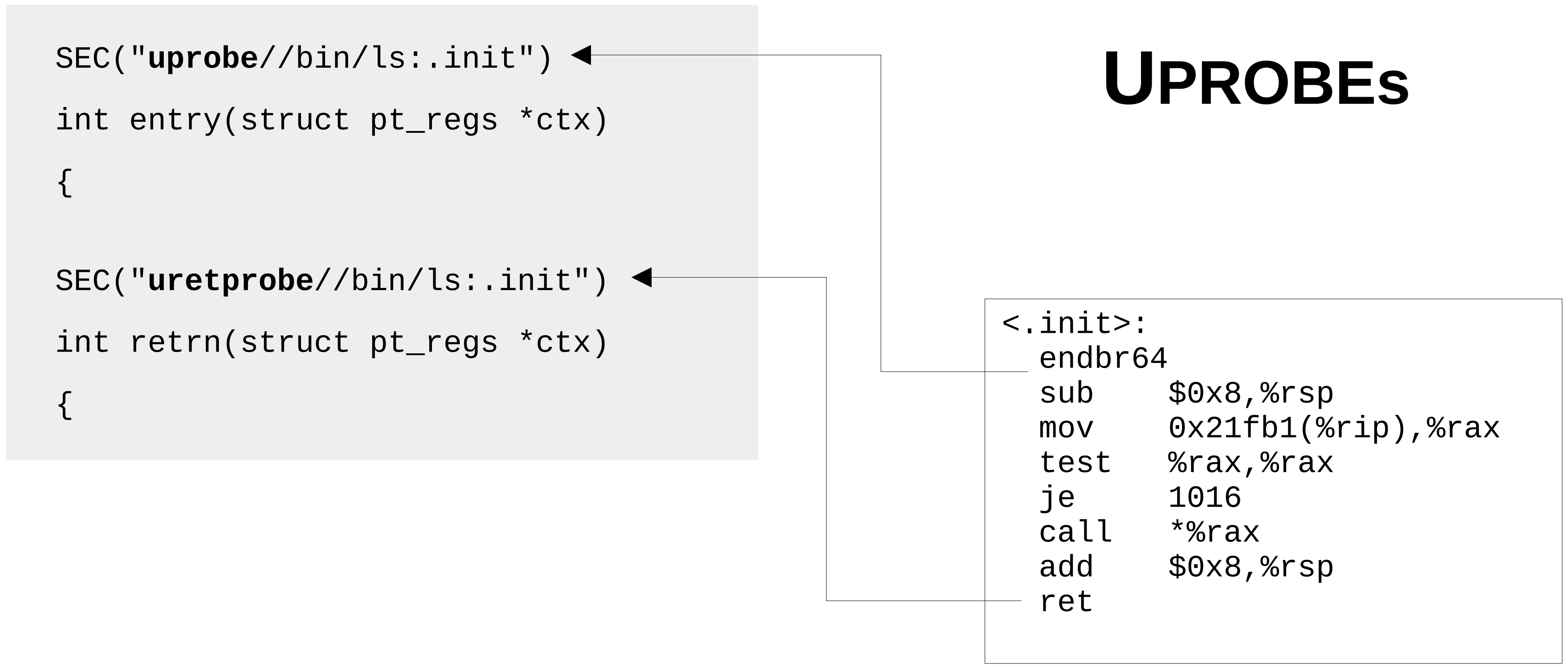


```
SEC("kprobe/ksys_read")
int entry(struct pt_regs *ctx)
{

SEC("kretprobe/ksys_read")
int retrn(struct pt_regs *ctx)
{

SEC("kprobe.multi/ksys_read")
int entry(struct pt_regs *ctx)
{

SEC("kretprobe.multi/ksys_read")
int retrn(struct pt_regs *ctx)
{
```

```
SEC("kprobe.session/ksys_read")
int session(struct pt_regs *ctx)
{
  if bpf_session_is_return() {
```

```
<ksys_read>:
   endbr64
   call    <__fentry__>
   push    %r13
   mov     %rsi,%r13
   push    %r12
   push    %rbp
   push    %rbx
   sub     $0x10,%rsp
   mov     %gs:0x28,%r12
   ...
   ret
```

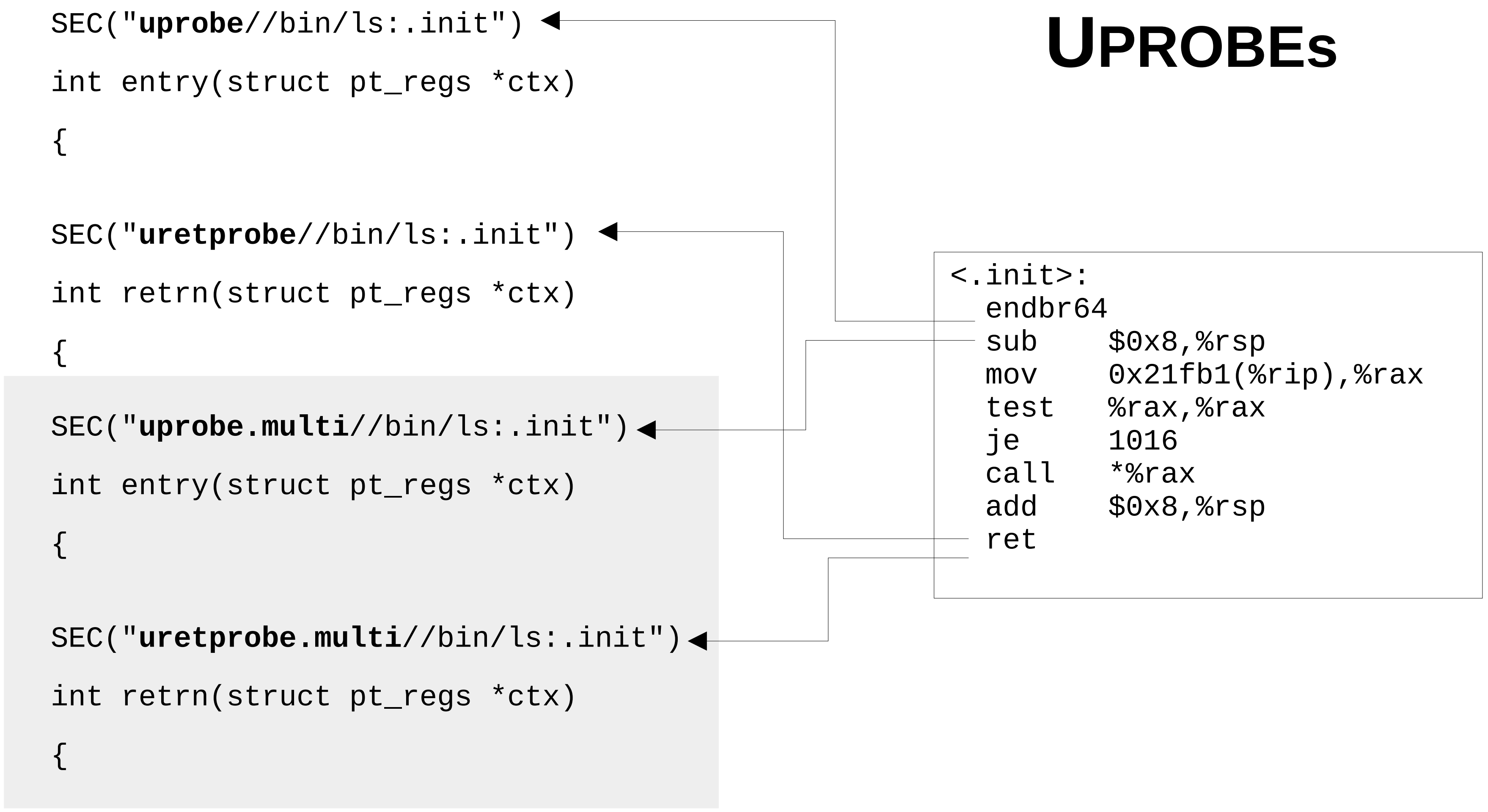
UPROBEs



```
SEC("uprobe//bin/ls:.init")
int entry(struct pt_regs *ctx)
{

SEC("uretprobe//bin/ls:.init")
int retrn(struct pt_regs *ctx)
{
```

```
<.init>:
  endbr64
  sub    $0x8,%rsp
  mov    0x21fb1(%rip),%rax
  test   %rax,%rax
  je     1016
  call   *%rax
  add    $0x8,%rsp
  ret
```

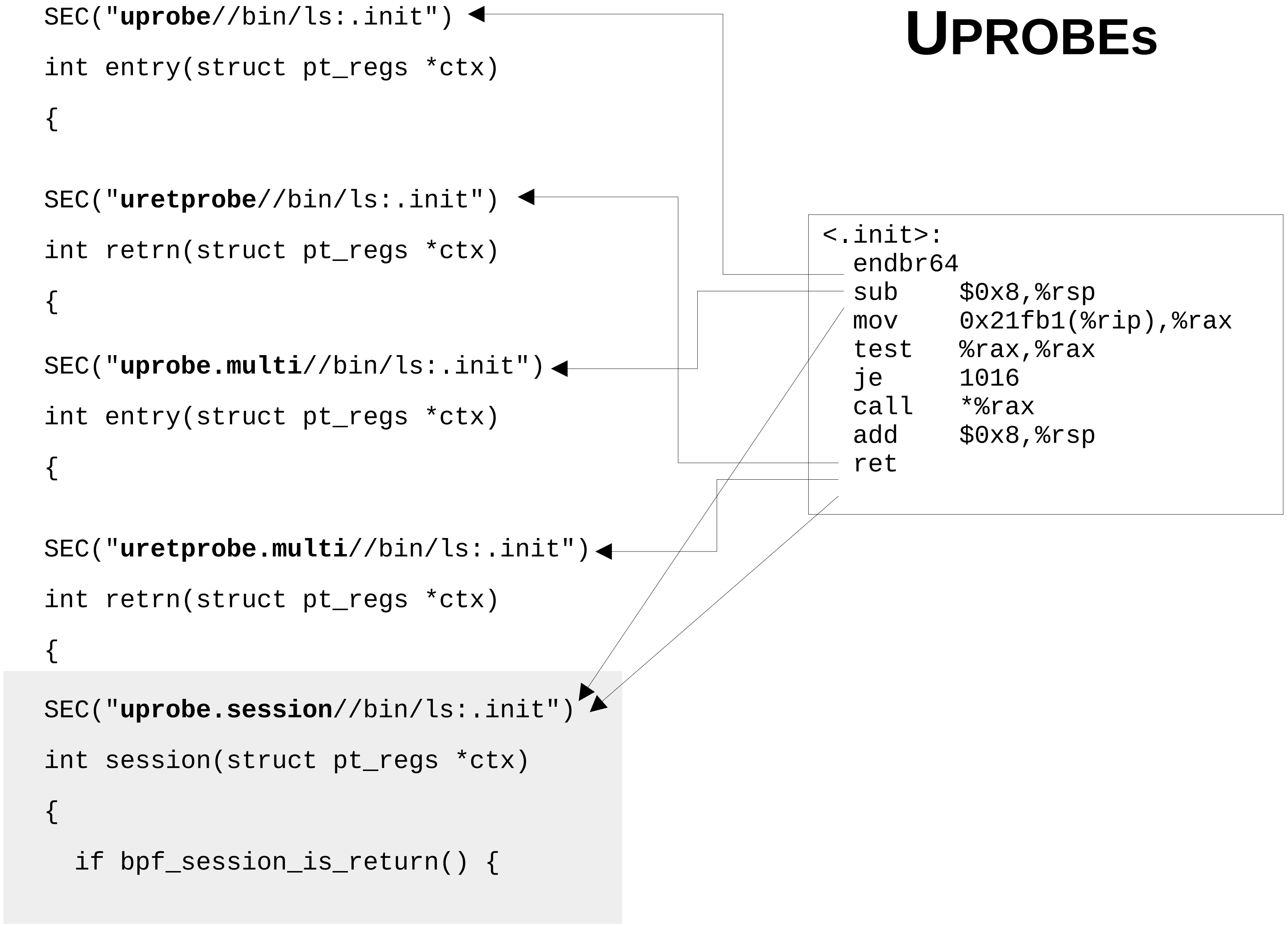
UPROBEs



```
SEC("uprobe//bin/ls:.init")
int entry(struct pt_regs *ctx)
{

SEC("uretprobe//bin/ls:.init")
int retrn(struct pt_regs *ctx)
{

SEC("uprobe.multi//bin/ls:.init")
int entry(struct pt_regs *ctx)
{

SEC("uretprobe.multi//bin/ls:.init")
int retrn(struct pt_regs *ctx)
{
```

```
<.init>:
  endbr64
  sub    $0x8,%rsp
  mov    0x21fb1(%rip),%rax
  test   %rax,%rax
  je     1016
  call   *%rax
  add    $0x8,%rsp
  ret
```

UPROBEs



```
SEC("uprobe//bin/ls:.init")

int entry(struct pt_regs *ctx)

{


SEC("uretprobe//bin/ls:.init")

int retrn(struct pt_regs *ctx)

{

SEC("uprobe.multi//bin/ls:.init")

int entry(struct pt_regs *ctx)

{


SEC("uretprobe.multi//bin/ls:.init")

int retrn(struct pt_regs *ctx)

{

SEC("uprobe.session//bin/ls:.init")

int session(struct pt_regs *ctx)

{

  if bpf_session_is_return() {
```

```
<.init>:
  endbr64
  sub    $0x8,%rsp
  mov    0x21fb1(%rip),%rax
  test   %rax,%rax
  je     1016
  call   *%rax
  add    $0x8,%rsp
  ret
```

SESSION

- on top of kprobe/uprobe_multi links
- one program attached for function entry and return
- conditional program execution on return probe
- session cookie

```
extern bool bpf_session_is_return(void) __ksym;

extern __u64 *bpf_session_cookie(void) __ksym;
```

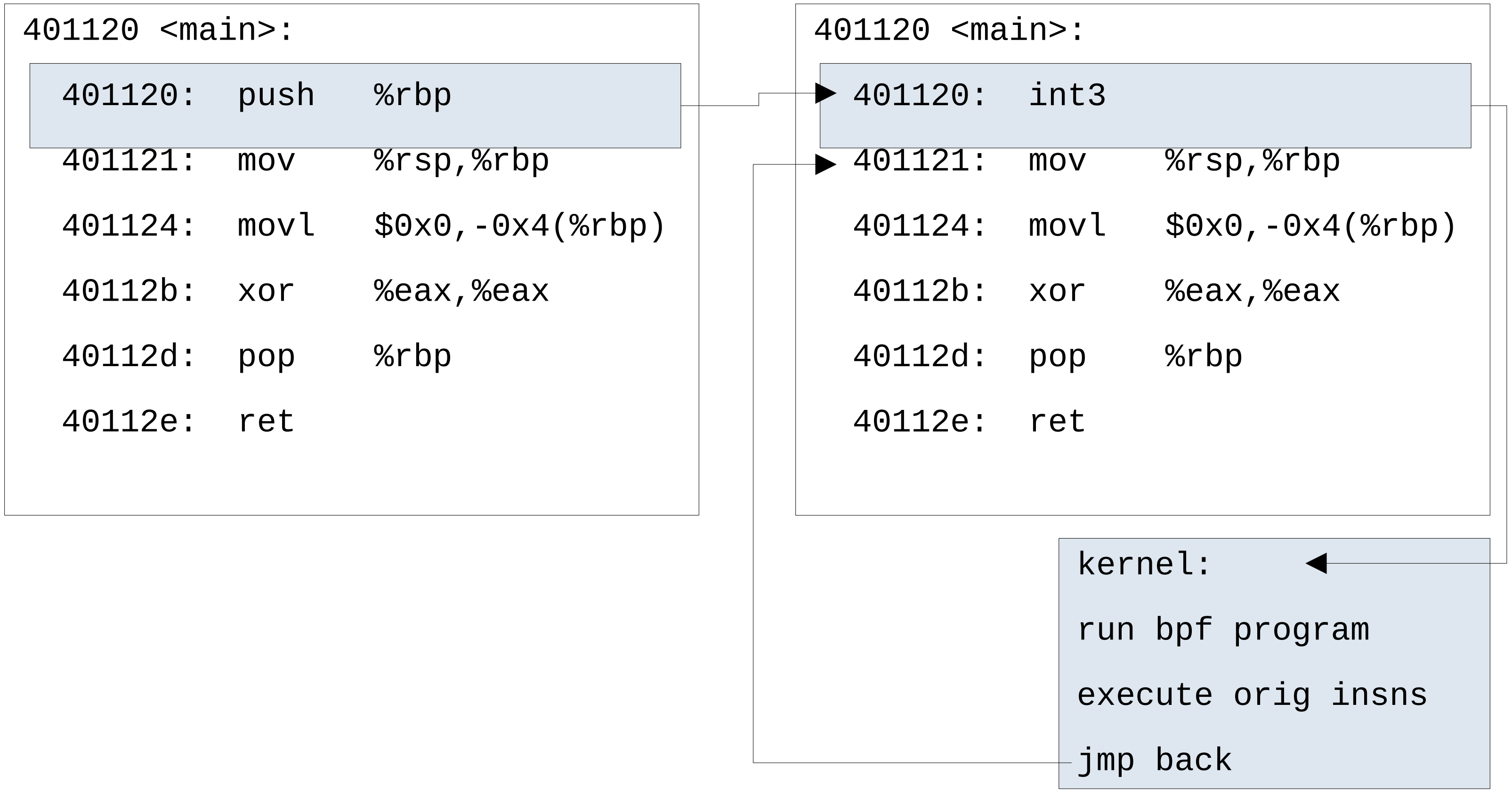
SUPPORT

- libbpf
- tetragon and cilium/ebpf support
- bpftrace

FASTER UPROBEs

- generic uprobes fixes
- x86_64 replace breakpoint with syscall

UPROBE

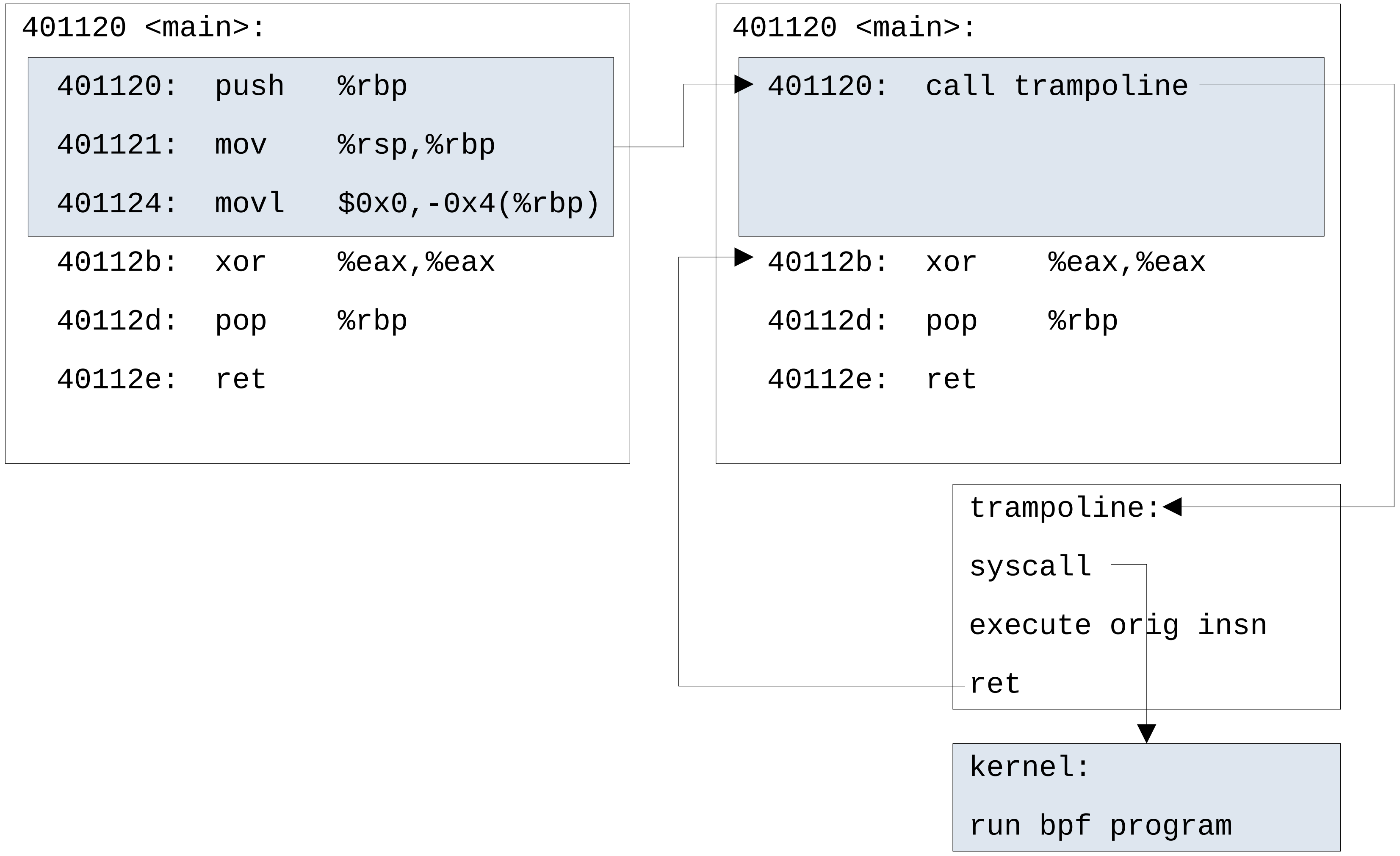


```
401120 <main>:
  401120:  push   %rbp
  401121:  mov    %rsp,%rbp
  401124:  movl   $0x0,-0x4(%rbp)
  40112b:  xor    %eax,%eax
  40112d:  pop    %rbp
  40112e:  ret
```

```
401120 <main>:
  401120:  int3
  401121:  mov    %rsp,%rbp
  401124:  movl   $0x0,-0x4(%rbp)
  40112b:  xor    %eax,%eax
  40112d:  pop    %rbp
  40112e:  ret
```

```
kernel:
run bpf program
execute orig insns
jmp back
```

UPROBE SPEEDUP



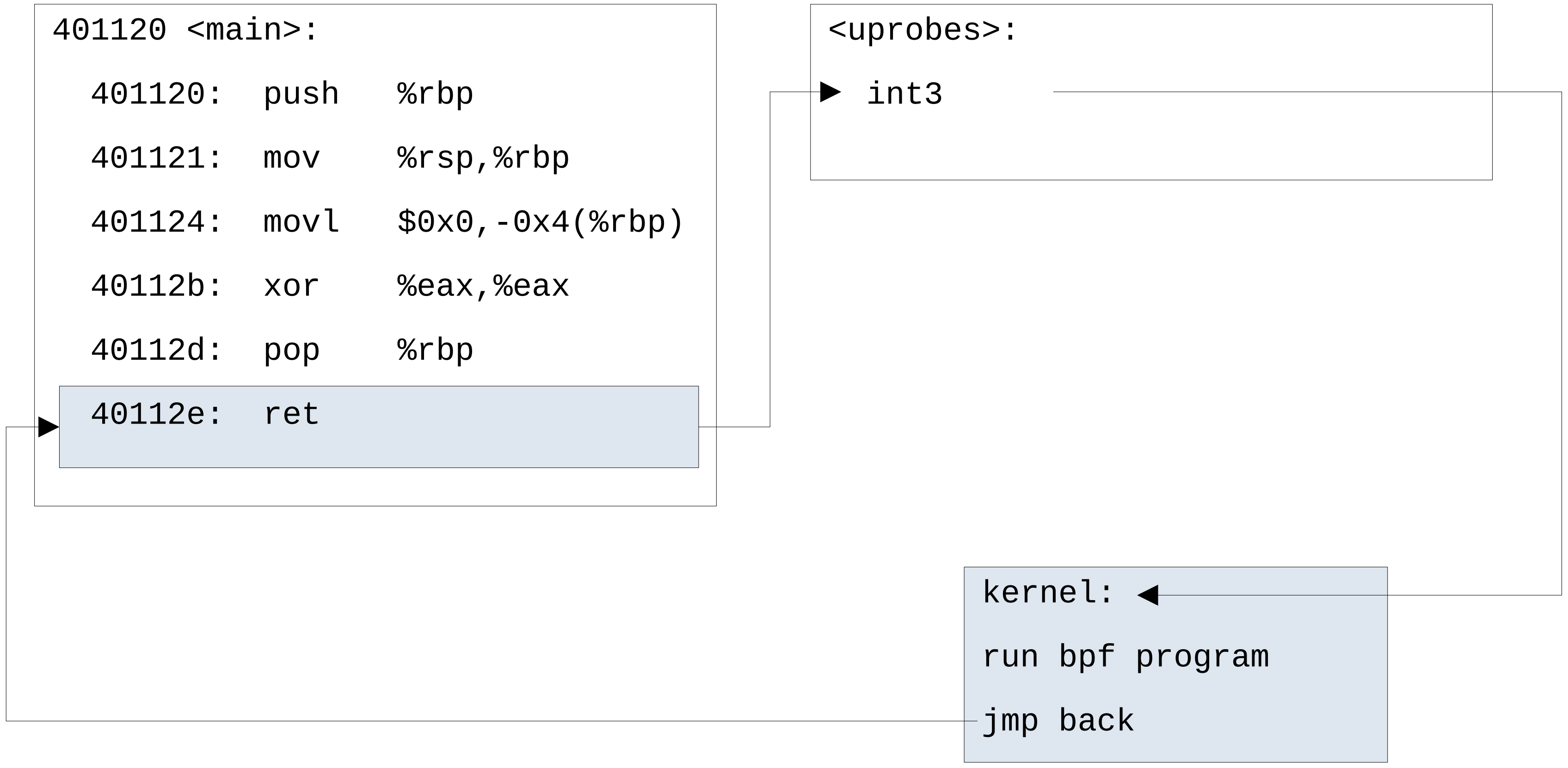
```
401120 <main>:
  401120:  push   %rbp
  401121:  mov    %rsp,%rbp
  401124:  movl   $0x0,-0x4(%rbp)
  40112b:  xor    %eax,%eax
  40112d:  pop    %rbp
  40112e:  ret
```

```
401120 <main>:
  401120:  call trampoline
  40112b:  xor    %eax,%eax
  40112d:  pop    %rbp
  40112e:  ret
```

```
trampoline:
syscall
execute orig insn
ret
```

```
kernel:
run bpf program
```

URETPROBE SPEEDUP

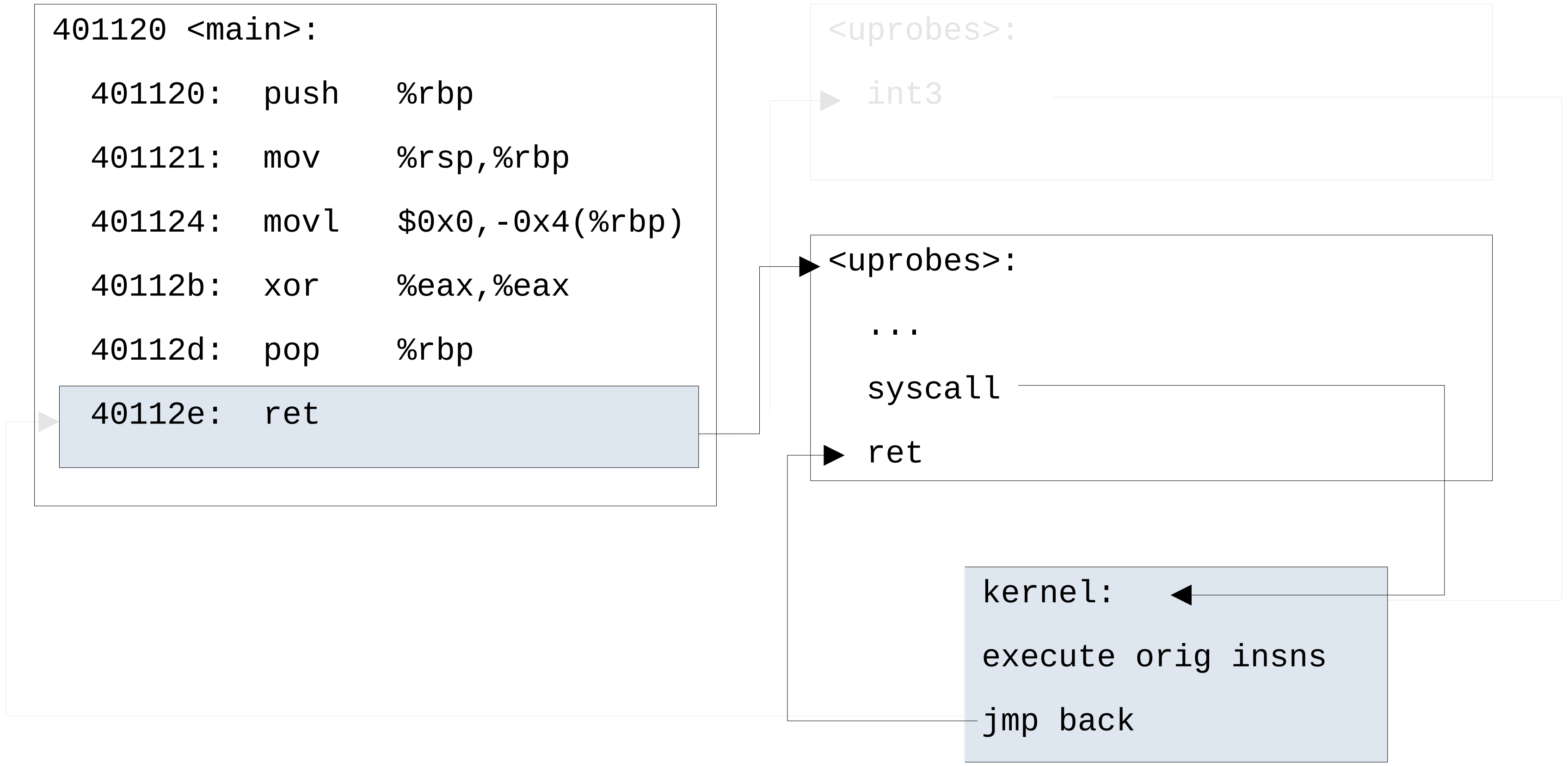


```
401120 <main>:
  401120:  push   %rbp
  401121:  mov    %rsp,%rbp
  401124:  movl   $0x0,-0x4(%rbp)
  40112b:  xor    %eax,%eax
  40112d:  pop    %rbp
  40112e:  ret
```

```
<uprobes>:
  int3
```

```
kernel:
run bpf program
jmp back
```

URETPROBE SPEEDUP



```
401120 <main>:
  401120:  push   %rbp
  401121:  mov    %rsp,%rbp
  401124:  movl   $0x0,-0x4(%rbp)
  40112b:  xor    %eax,%eax
  40112d:  pop    %rbp
  40112e:  ret
```

```
<uprobes>:
  int3
```

```
<uprobes>:
  ...
  syscall
  ret
```

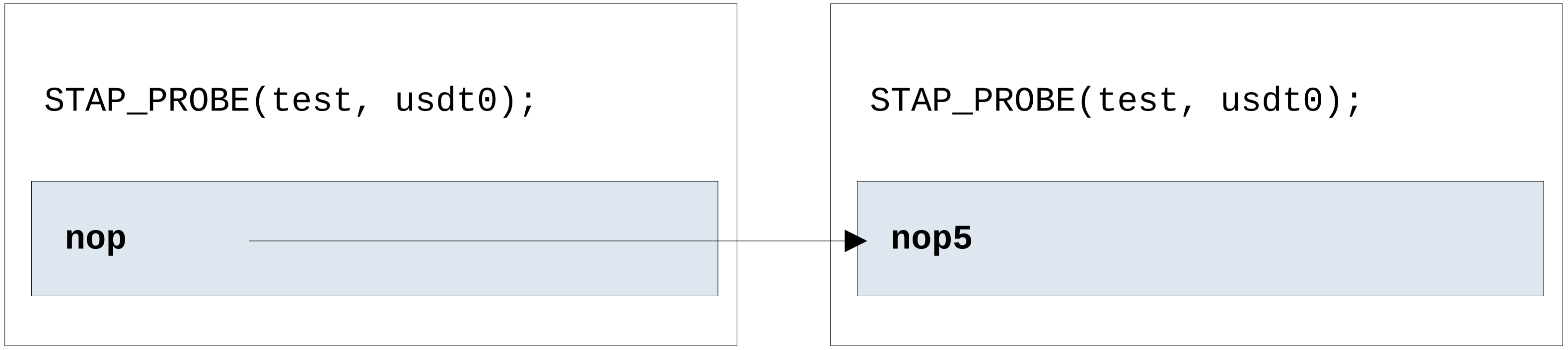
```
kernel:
execute orig insns
jmp back
```

USDT SPEEDUP

```
STAP_PROBE(test, usdt0);
```

```
nop
```

USDT SPEEDUP



```
STAP_PROBE(test, usdt0);
```

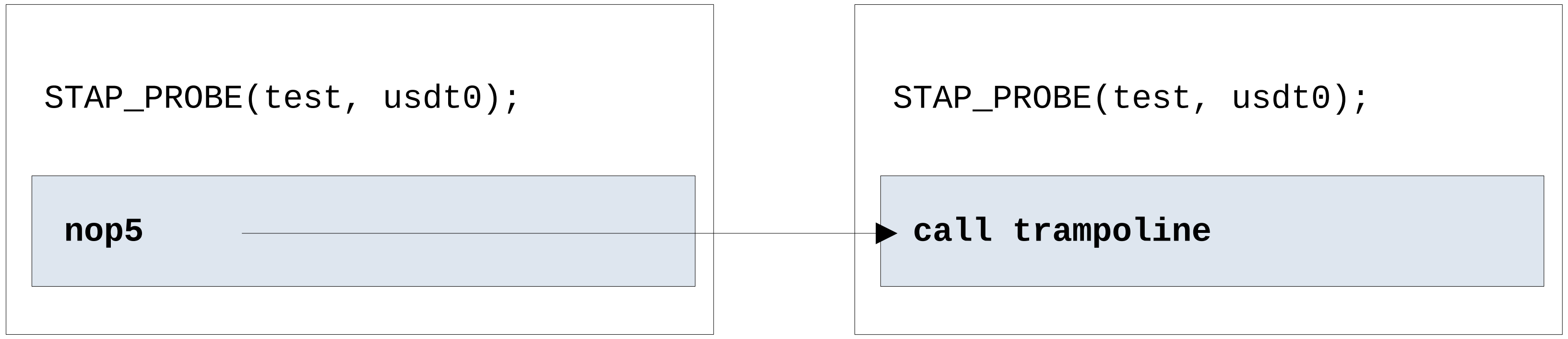
```
STAP_PROBE(test, usdt0);
```

USDT SPEEDUP

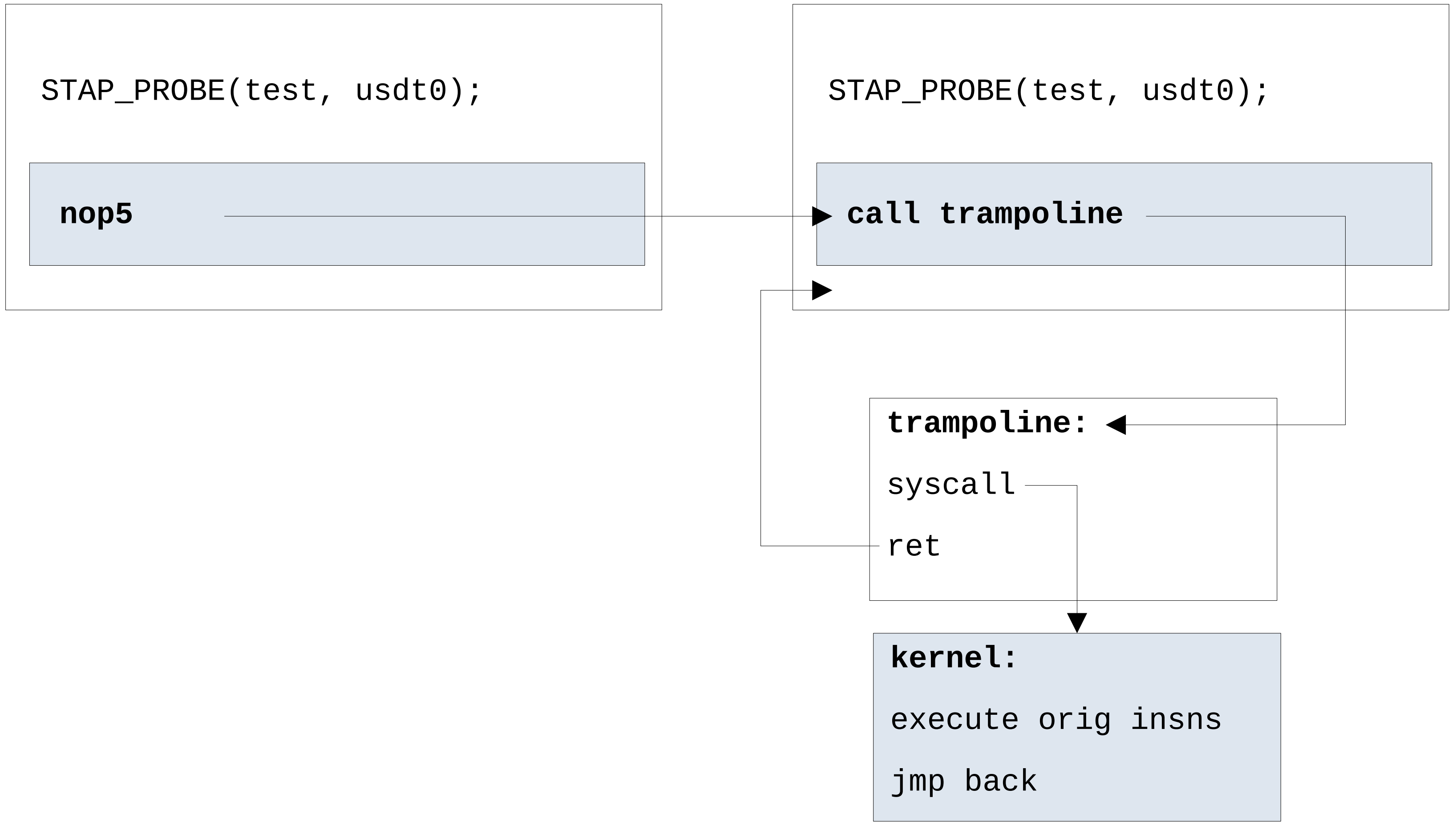
```
STAP_PROBE(test, usdt0);
```

nop5

USDT SPEEDUP



USDT SPEEDUP



PROBLEMs

- 5 byte instruction atomic update
- 5 byte call won’t cover whole address space
- backward compatibility
- seccomp

thanks, questions?