~~Secure applications on top of L4~~

FOSDEM'14

Sartakov A. Vasily

2014

Security Gap

Red FOSDEM ‘14

Sartakov A. Vasily
ksys labs
2014

Syuzhet

- Intro
- Myth about uKernel: Security vs Performance
- Attacks on stack
- W xor X memory support in L4Re
- Conclusion

About me

- Sartakov A. Vasily
- Ksys labs – Small RnD company
  - Mobile and network Hardware-software systems
  - Not only uKernels
  - Open Source and Research projects
    - Evaluate, apply, implement
    - Industry point of view

About us

- Joined to community 3 years ago
- Fiasco.OC + L4Re
- Genode
- Respect Open Source – we publish too.

What has changed since..?

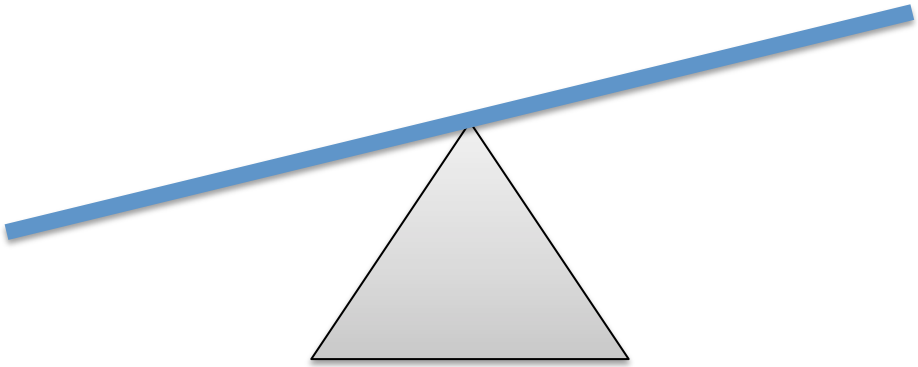
- Transformation from university to commercial projects
- New step of maturity

Syuzhet

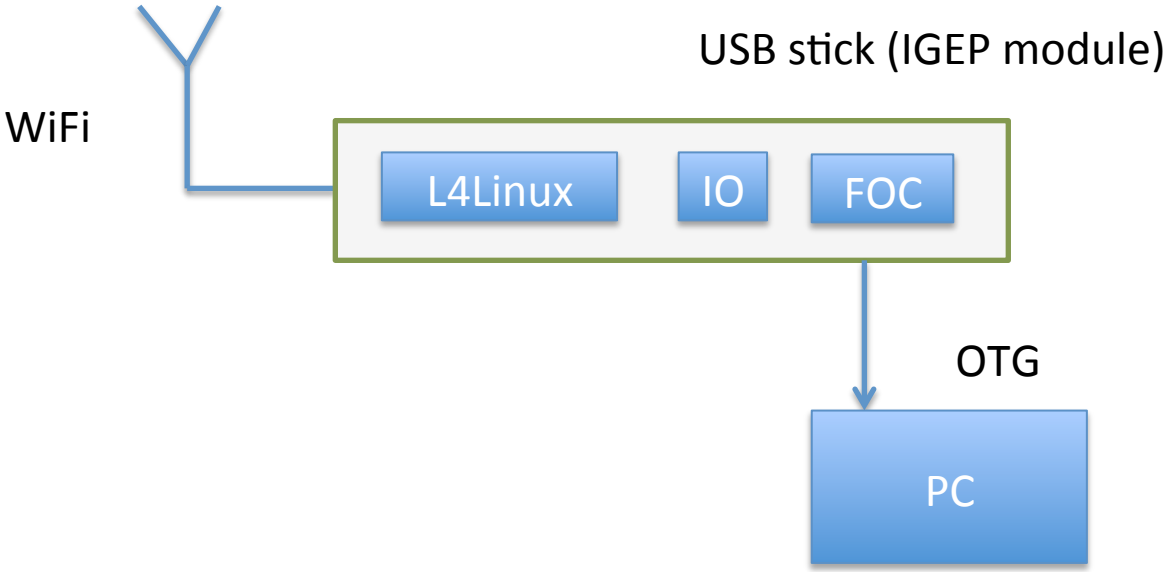
- Intro
- Myth about uKernel: Security vs Performance
- Attacks on stack
- W xor X memory support in L4Re
- Conclusion

Myth

- Dialectical pairs: performance vs security
- Security is a most strong part…(??)



L4Linux, USB-OTG, WiFi, Omap3

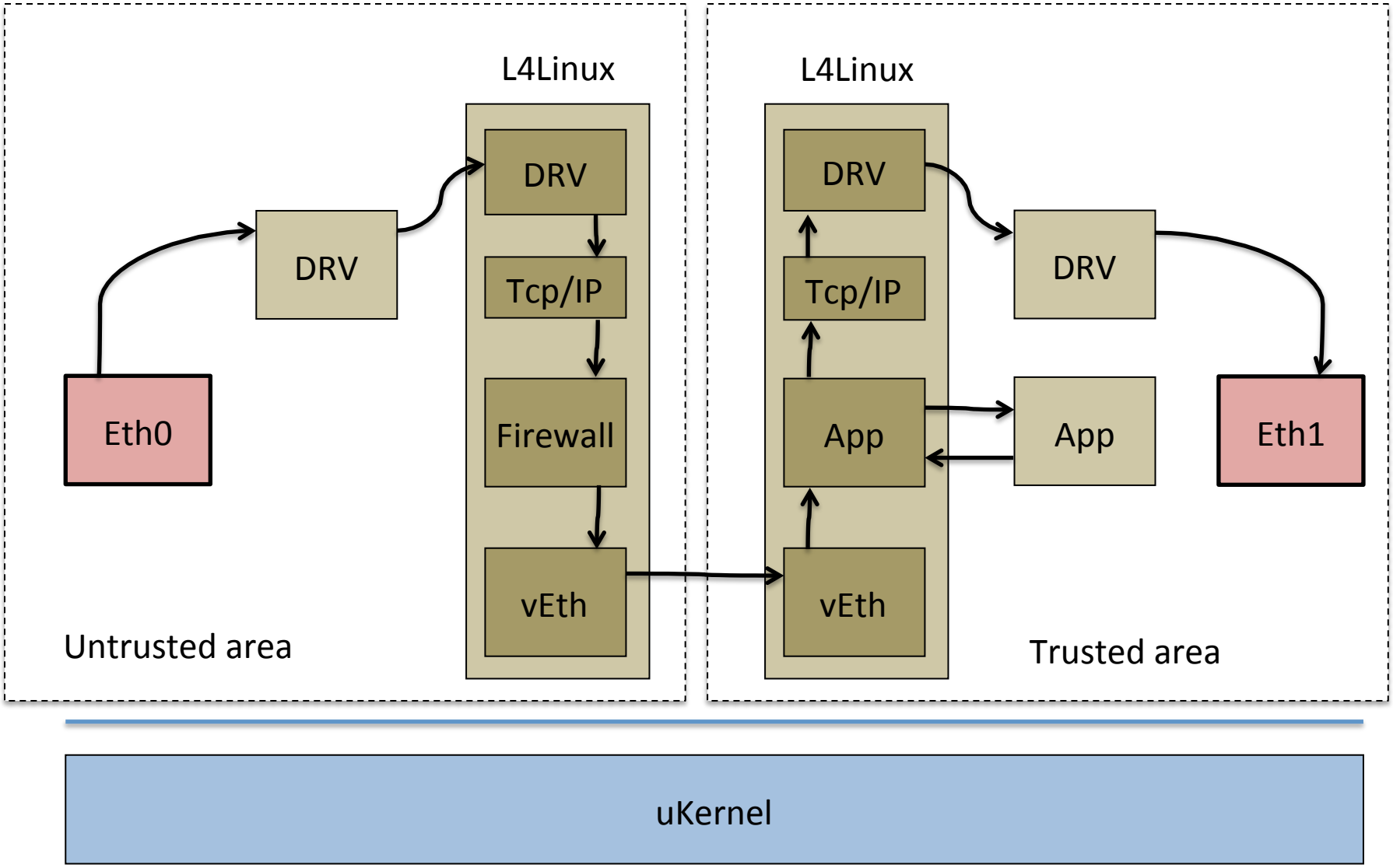


Linux vs uKernel:

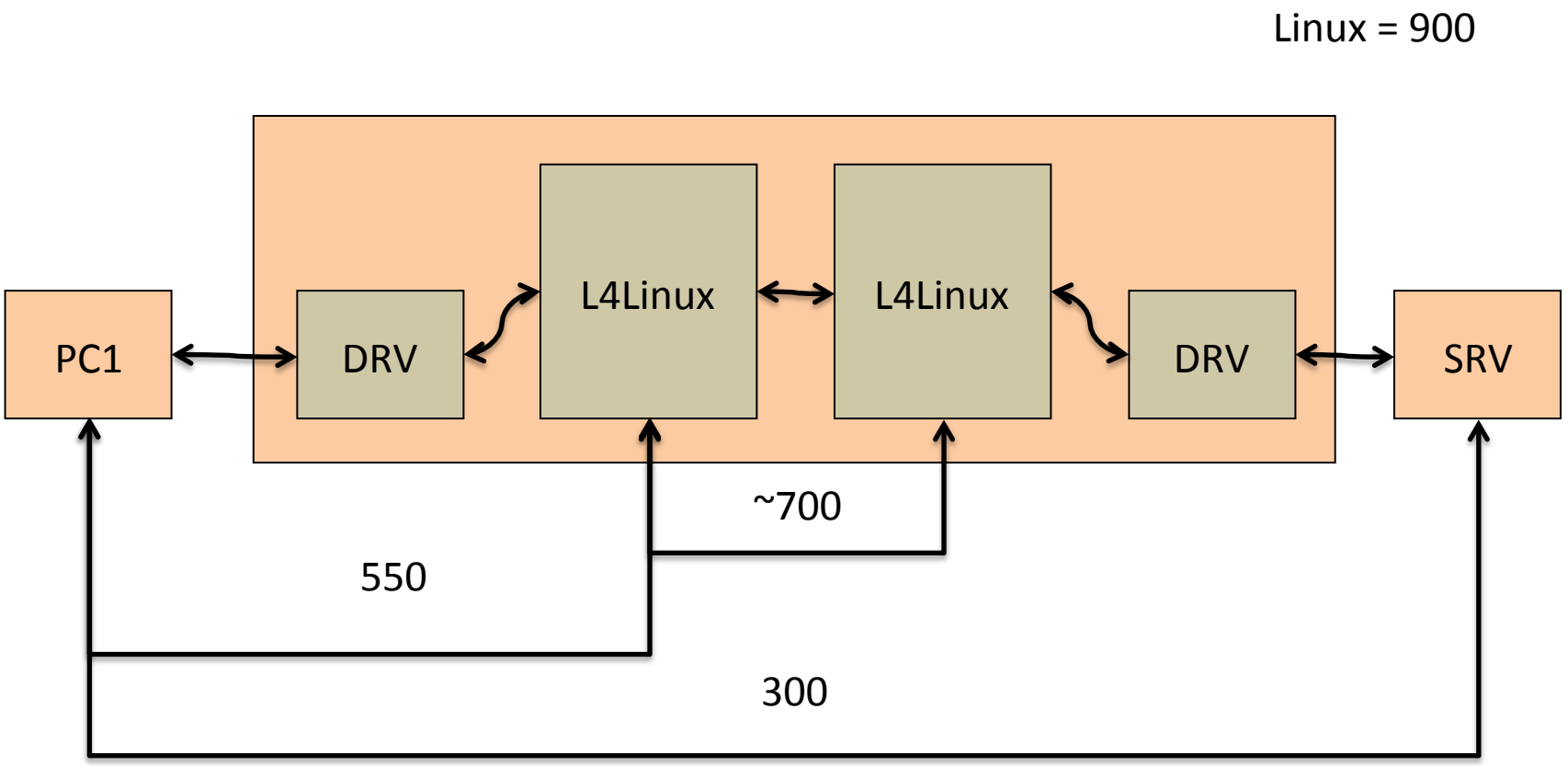
L: 5.5 Mbits/sec

l4Linux: 7.35 Mbits/sec

Example2: Network service on top x86

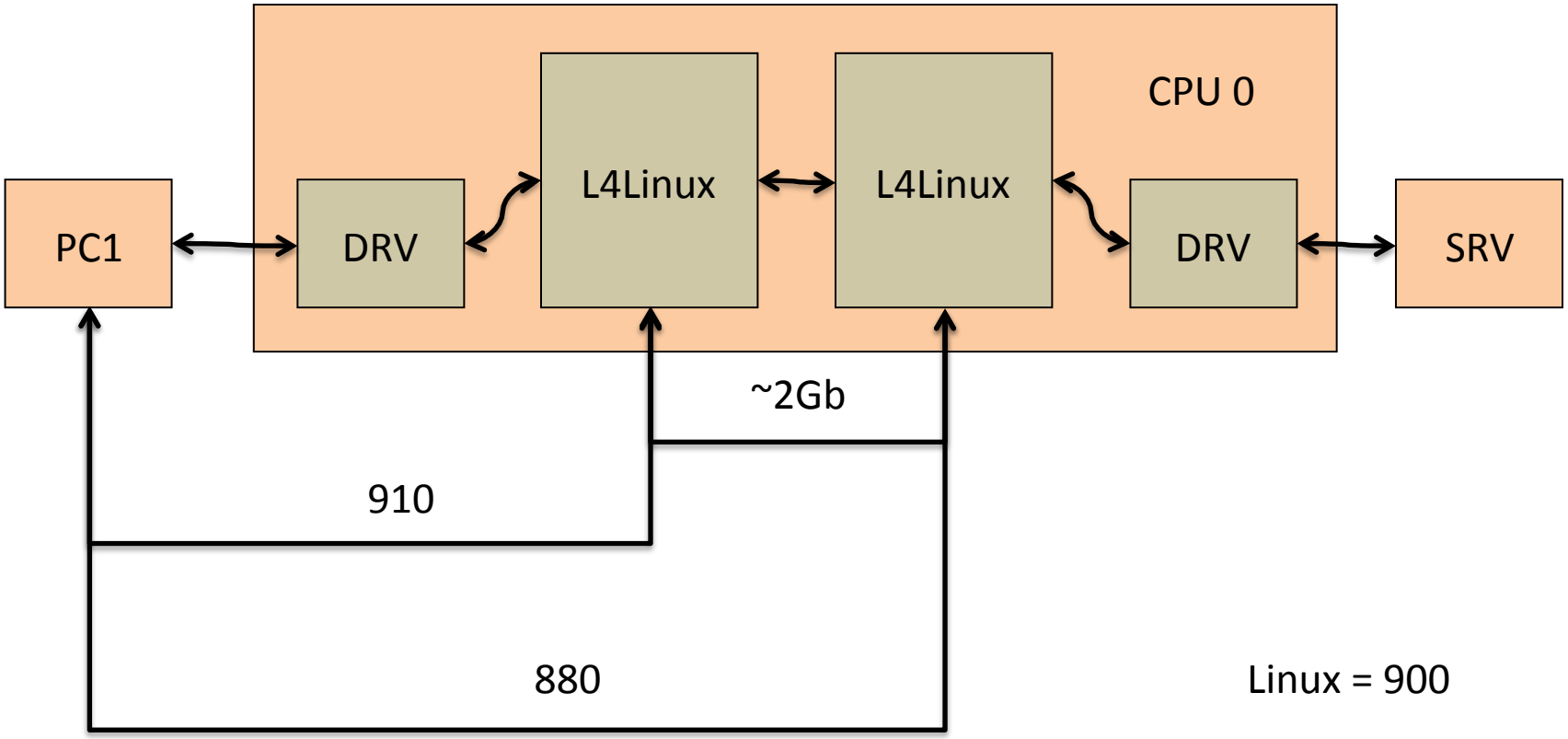


Example2: Performance tests (Genode)

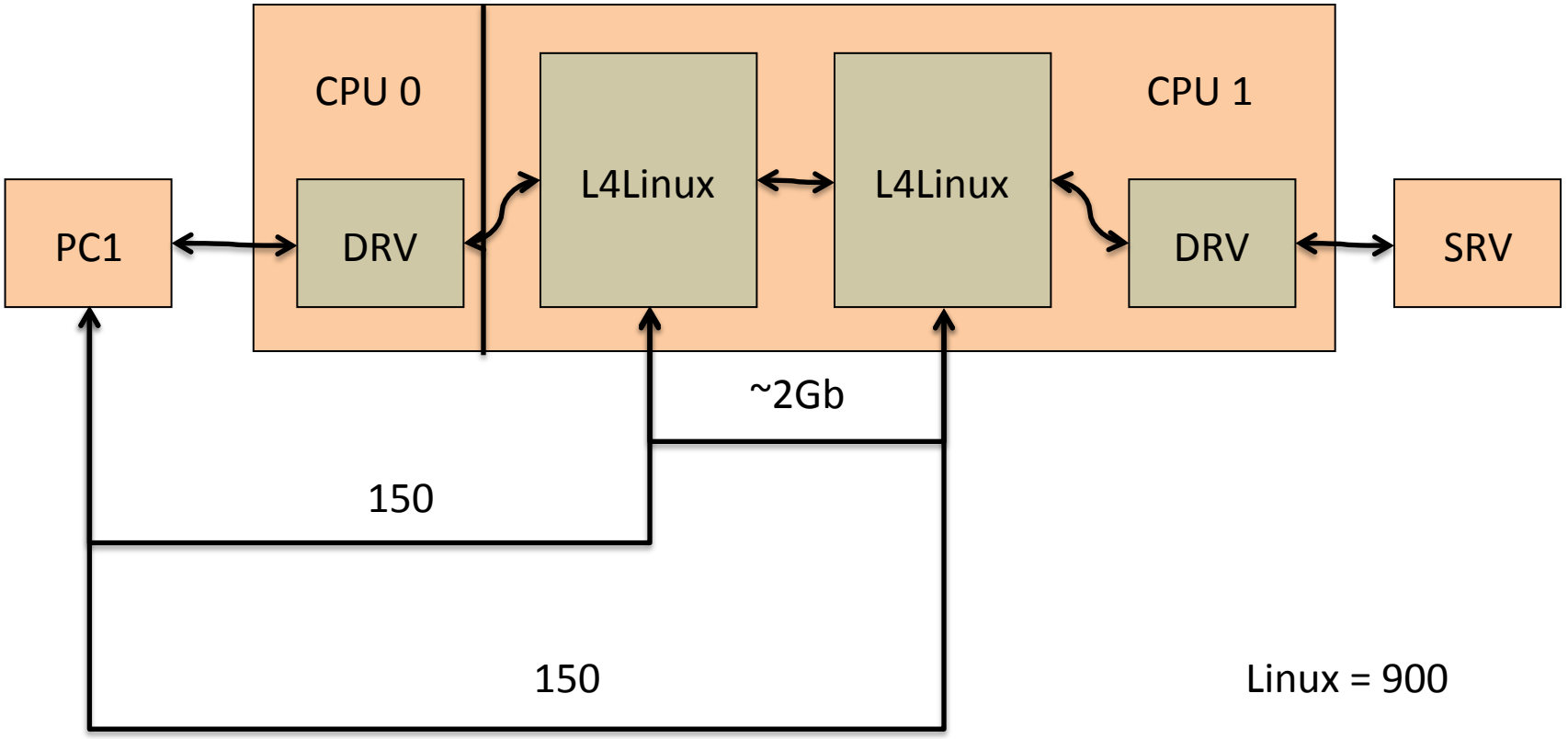


Disclamer: This is a old experiment result, since that time Genodes rework event mechanism and add some new software

Example2: Performance tests (L4Re)



Example3: Performance tests (L4Re) (SMP)

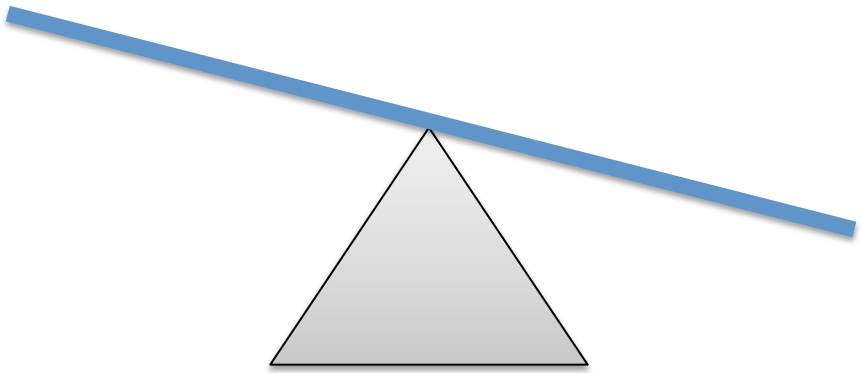


Conclusion

- Performance is not an unsolvable problem
- Performance is not an out-of-box feature

Come back again to Myth

- uKernels are secure
  - Small size of TCB (less errors, verification)
  - Stable api/abi
  - Drivers in userspace
  - Isolation/separation of components
- This is true, but…



Stack protection

- Linux: Exec shield, since 2003
- Linux: PaX, since 2000
- Windows (sic!) : DEP, since XP SP2 (!!!)

- Genode – “canaries” is disabled in toolchains (StackGuard)
- L4Re – “canaries” is disabled at compilation time by gcc flag

Why I am care

- Third part software (Linux, BSD)
- A lot of wrappers
- -> potentially vulnerable points
- -> malicious software and intrusion

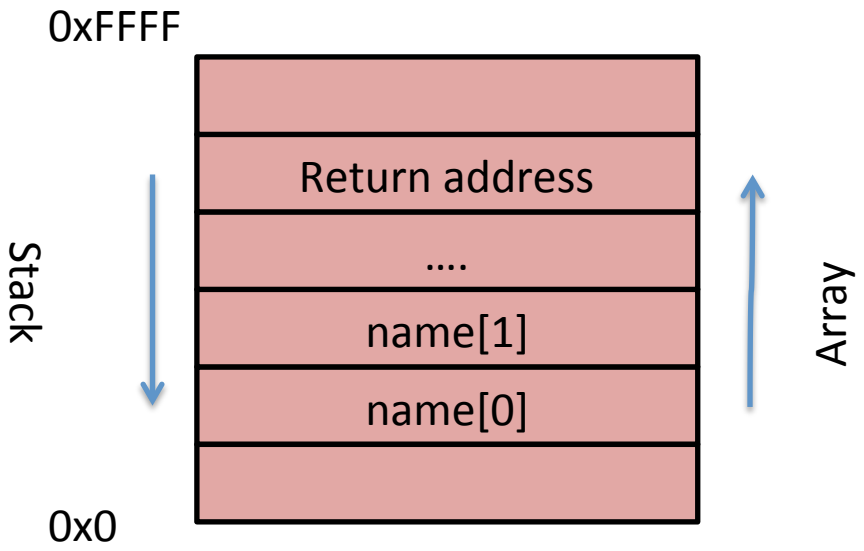
Syuzhet

- Intro
- Myth about uKernel: Security vs Performance
- Attacks on stack
- W xor X memory support in L4Re
- Conclusion

Smashing the stack for fun and profit ©

- Von Neumann architecture:
  - Data and instructions are in the same place
  - There is not difference
  - Type of memory defined by operations on it
  - Data can be used as instructions

Smashing the stack




```
int namelen (void) {
        char name[21];
        gets(name);
        return strlen(name);
}
```

- Best case – Segfault
- Worst case - malicious execution

Intrusion

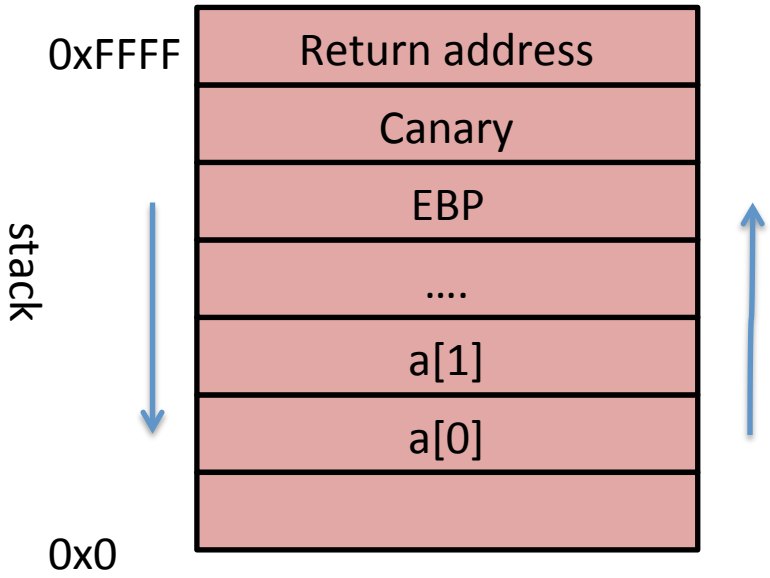
- Payload
- Execution flow

- Memory that devoted for data becomes set of instrictions

Counteraction

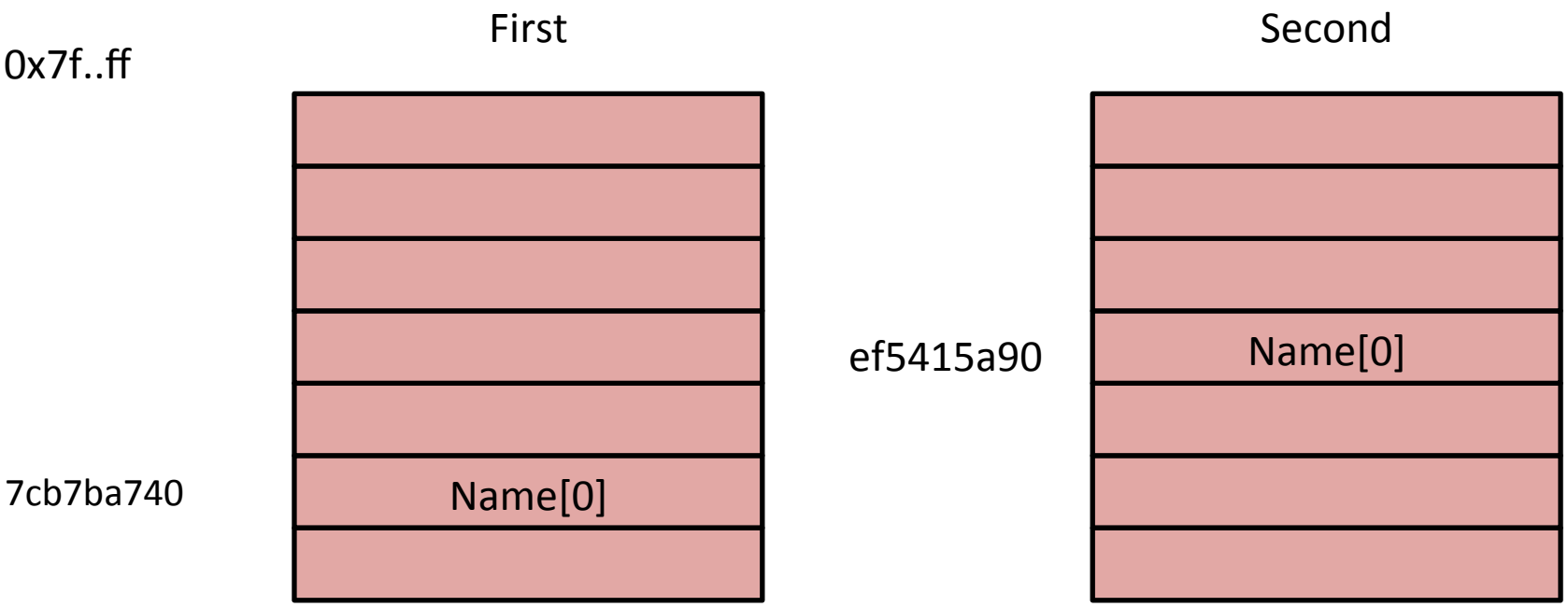
- Canaries
- W xor X memory
- Address space layout randomization (ASLR)

Canaries (stack guard)



Prolog and epilog of function

ASLR




```
char name[64];
  printf("%p\n", name);
  puts("What's your name?");
  gets(name);
  printf("Hello, %s!\n", name);
```

W xor X memory

- Hardware or software implementation
- Memory protected from execution
- Prevents payload uploading

Counteraction

- Canaries
- W xor X memory
- Address space layout randomization (ASLR)

Syuzhet

- Intro
- Myth about uKernel: Security vs Performance
- Attacks on stack
- W xor X memory support in L4Re
- Conclusion

W xor X memory

- Hardware support: AMD64, ARM,
- NX bit
- Disable execution
- Requires support by kernel and environment.

KE 1: memory objects

- Entities:
  - Dataspace
  - Region mapper
- Semantic:

```
1 L4Re::Env::env()->mem_alloc()->alloc(size, ds, L4Re::Mem_alloc::Executable)
```

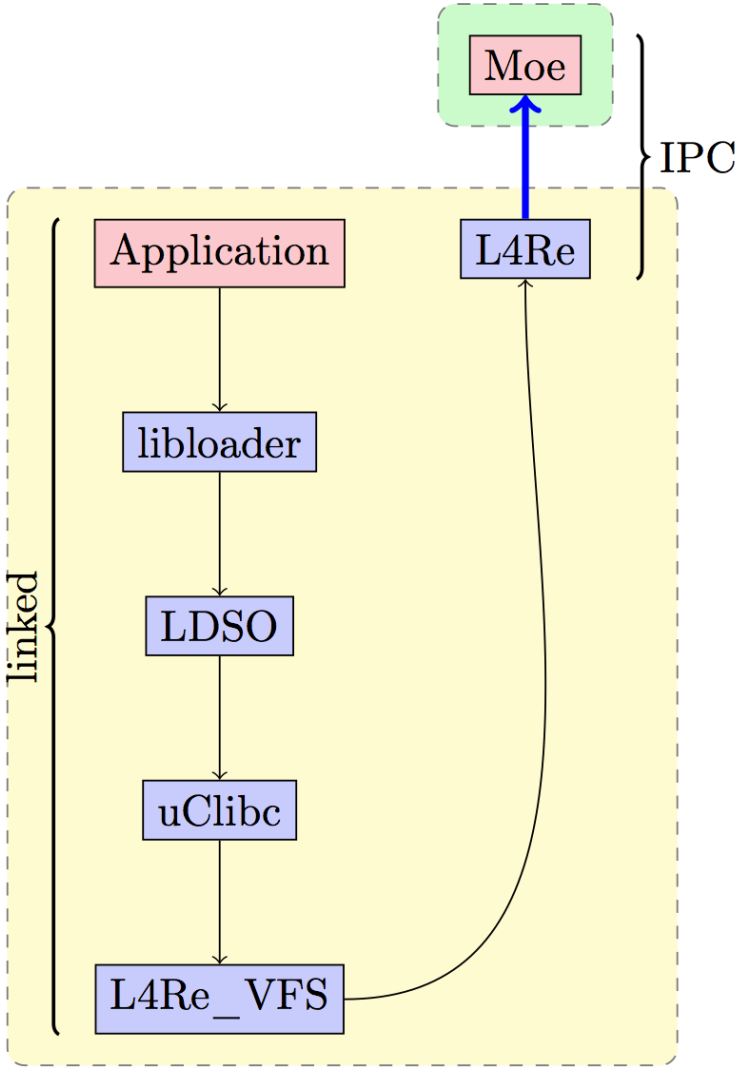
```
1 L4Re::Env::env()->rm()->attach(&ptr, size,
2         L4Re::Rm::Search_addr | L4Re::Rm::Eager_map
3         | L4Re::Rm::Executable, ds)
```

KE 2: Starting, ELF

- ELF file contains sections with access flags
- Elf-loader creates region according to ELF

```
1     PHDR off    0x0000000000000040 vaddr 0x0000000001000040 paddr
    0x0000000001000040 align 2**3
2           filesz 0x0000000000000118 memsz 0x0000000000000118 flags r--
3     LOAD off    0x0000000000000000 vaddr 0x0000000001000000 paddr
    0x0000000001000000 align 2**12
4           filesz 0x000000000001cbe8 memsz 0x000000000001cbe8 flags r-x
5     LOAD off    0x000000000001d000 vaddr 0x000000000101d000 paddr
    0x000000000101d000 align 2**12
6           filesz 0x0000000000000370 memsz 0x00000000000143e0 flags rw-
7      TLS off    0x000000000001d000 vaddr 0x000000000101d000 paddr
    0x000000000101d000 align 2**3
8           filesz 0x0000000000000000 memsz 0x0000000000000018 flags rw-
9 0x60000014 off    0x00000000000147a0 vaddr 0x00000000010147a0 paddr
    0x00000000010147a0 align 2**4
10          filesz 0x0000000000000018 memsz 0x0000000000000018 flags r--
```

KE 2: Starting



KE 3: The Gentleman's Set of Tests

Тест	L4Re	L4Re + NX dataspace
anonmap	Vulnerable	Killed
execbss	Vulnerable	Killed
execdata	Vulnerable	Killed
execstack	Vulnerable	Killed
mprotanon	Vulnerable	Killed
mprotbss	Vulnerable	Killed
mprotdata	Vulnerable	Killed
mprotheap	Vulnerable	Killed
mprotstack	Vulnerable	Killed

Restrictions

- For well protection all techniques should be used
  - W xor X
  - ASLR
  - Canaries
  - Other…
- I386 does not have a hardware NX

Restrictions: L4Linux

- L4Linux uses low level Fiasco.OC calls
- L4Linux starts program self
- Obtain one big dataspace from kernel.
- L4Linux has to manage W xor X allocation self
- Does not support AMD64
- A big hole in security

Thank you for attention

Sartakov A. Vasily
sartakov@ksyslabs.org

Ksys labs LLC
http://ksyslabs.com,
http://ksyslabs.org,
info@ksyslabs.com

* Please do not fork me on github