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
Example 2: Network service on top x86

Untrusted area

Eth0

DRV

Tcp/IP

Firewall

vEth

L4Linux

Trusted area

Eth1

DRV

Tcp/IP

App

vEth

L4Linux

uKernel
Example2: Performance tests (Genode)

Disclamer: This is a old experiment result, since that time Genodes rework event mechanism and add some new software
Example 2: 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

- Best case – Segfault
- Worst case - malicious execution

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

• Payload
• Execution flow

• Memory that devoted for data becomes set of instructions
Counteraction

• Canaries
• W xor X memory
• Address space layout randomization (ASLR)
Canaries (stack guard)

<table>
<thead>
<tr>
<th>0xFFFF</th>
<th>Return address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canary</td>
<td></td>
</tr>
<tr>
<td>EBP</td>
<td></td>
</tr>
<tr>
<td>.....</td>
<td></td>
</tr>
<tr>
<td>a[1]</td>
<td></td>
</tr>
<tr>
<td>a[0]</td>
<td></td>
</tr>
</tbody>
</table>

Prolog and epilog of function
char name[64];
printf("%p\n", name);
puts("What's your name?"urrets(name);
printf("Hello, %s!\n", name);
W xor X memory

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

- Canaries
- $W \text{ xor } X$ 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:

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

```c
L4Re::Env::env() -> rm() -> attach(&ptr, size,
L4Re::Rm::Search_addr | L4Re::Rm::Eager_map
| L4Re::Rm::Executable, ds)
```
KE 2: Starting, ELF

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

```plaintext
<table>
<thead>
<tr>
<th></th>
<th>PHDR off</th>
<th></th>
<th>memsz</th>
<th></th>
<th>flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0x0000000000000000</td>
<td>0x0000000000100000</td>
<td>0x00000000000100000</td>
<td>0x00000000000100000</td>
<td>r--</td>
</tr>
<tr>
<td>2</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>r--</td>
</tr>
<tr>
<td>3</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>r--</td>
</tr>
<tr>
<td>4</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>r--</td>
</tr>
<tr>
<td>5</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>r--</td>
</tr>
<tr>
<td>6</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>r--</td>
</tr>
<tr>
<td>7</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>r--</td>
</tr>
<tr>
<td>8</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>r--</td>
</tr>
<tr>
<td>9</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>r--</td>
</tr>
<tr>
<td>10</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
<td>r--</td>
</tr>
</tbody>
</table>
```
KE 2: Starting
KE 3: The Gentleman's Set of Tests

<table>
<thead>
<tr>
<th>Test</th>
<th>L4Re</th>
<th>L4Re + NX dataspace</th>
</tr>
</thead>
<tbody>
<tr>
<td>anonmap</td>
<td>Vulnerable</td>
<td>Killed</td>
</tr>
<tr>
<td>execbss</td>
<td>Vulnerable</td>
<td>Killed</td>
</tr>
<tr>
<td>execdata</td>
<td>Vulnerable</td>
<td>Killed</td>
</tr>
<tr>
<td>execstack</td>
<td>Vulnerable</td>
<td>Killed</td>
</tr>
<tr>
<td>mprotanon</td>
<td>Vulnerable</td>
<td>Killed</td>
</tr>
<tr>
<td>mprotbss</td>
<td>Vulnerable</td>
<td>Killed</td>
</tr>
<tr>
<td>mprotdata</td>
<td>Vulnerable</td>
<td>Killed</td>
</tr>
<tr>
<td>mprotheap</td>
<td>Vulnerable</td>
<td>Killed</td>
</tr>
<tr>
<td>mprotstack</td>
<td>Vulnerable</td>
<td>Killed</td>
</tr>
</tbody>
</table>
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