Thread-Level Attack-Surface Reduction

Florian Rommel
rommel@sra.uni-hannover.de
Leibniz Universität Hannover

Christian Dietrich
christian.dietrich@tuhh.de
Hamburg University of Technology

Andreas Ziegler
ziegler@cs.fau.de
Friedrich-Alexander-Universität Erlangen-Nürnberg

Illia Ostapyshyn
ostapyshyn@sra.uni-hannover.de
Leibniz Universität Hannover

Daniel Lohmann
lohmann@sra.uni-hannover.de
Leibniz Universität Hannover
Return-Oriented Programming (ROP)

Stack:

- Buffer
- ...
- Return addr

Buffer overflow

Existing code:

- Return
- Return
- Return

Gadget addresses:

- Gadget 1
- Gadget 2
- Gadget 3

Attack / Exploit Surface
Binary Debloating

Remove unnecessary code from dynamic libraries

Why?

→ Smaller binaries
→ Reduced attack/exploit surface

Still a lot of code present
(all potentially needed code from libs + whole main executable)

Ziegler et al. '19
"Honey, I Shrunk the ELFs: Lightweight Binary Tailoring of Shared Libraries."
ACM Transactions on Embedded Computing Systems

Use of musl libc functions by vsftpd

Executions

- unused
- 1 - 10
- ≤ 100
- ≤ 1000
- ≤ 10000
- max 7068
Dynamic Debloating

*Remove unnecessary code from running processes*

**Why?**
- Smaller binaries
- Reduced attack/exploit surface

Porter et al. '20
“BlankIt Library Debloating” PLDI

significant code reduction, but still *only* for libraries

→ Poor overall attack-surface reduction
Code Usage (Function Granularity)

MariaDB: **52%** of the code in bytes (≈ **60%** of the functions) stems from the main binary

- "read-only" client
  - only 44% shared
  - worker thread
  - 8% code used
    - 11% of the main binary

- "insert" client
  - worker thread
  - 7% code used
    - 9% of main b.

- "write-only" client
  - worker thread
  - 8% code used
    - 11% of main b.

... Consider
- main binary
- thread contexts
Thread-Level Attack-Surface Reduction (TLASR)

- **Dynamic** – Eliminates functions during runtime / on-demand restoration Enhanced by static call-graph analysis
- **Whole process** – Considers the main executable and libraries
- **Per thread** – Works on the context of individual threads
TLASR: Basic Concept

entry function

eliminated functions
(lazily) restored function

main binary

same process

entry function

eliminated functions
(lazily) restored function

main binary

entry function

eliminated functions
(lazily) restored function

main binary
CTE: Eliminating and Restoring Functions

CTE: context-based .text elimination

→ Binary analysis tool (CTEmeta) + runtime library (libCTE)

- **CTEmeta** - ahead of time
  - Gathers static callgraph information
  - Requires only ELF symbols (no source code, no debug information)

- **libCTE** - runtime
  - Eliminates functions at runtime, restores them on call
  - Uses call-graph info (collected by CTEmeta) to validate function restores
  - (Currently) manual integration into the program
CTE: Killing, Wiping, and Restoring Functions

**Function 1:**
```assembly
lea eax, [rsi+rsi*8]
add eax, edi
ret
```
**Kill**

**Function 1:**
```assembly
int3
int3
int3
```
**ROP attack**

**Function 2:**
```assembly
sub rsp, 0x8
mov edx, 0x30
mov edi, 0x1
xor eax, eax
lea rsi, [rip+0xca3]
...```
**Wipe**

**Function 2:**
```assembly
mov rax, restore
call rax
.word <fn index>
int3
int3
...```
**Implant**

**Function 2:**
```assembly
sub rsp, 0x8
mov edx, 0x30
mov edi, 0x1
xor eax, eax
lea rsi, [rip+0xca3]
...```
**Restore**

**Dynamic Restoration:**
- **Validation** → **call site** → **call graph** (full CFI)

**Static CTEmeta-generated info**
Adress-Space Views (for Linux)  
(Rommel et al. '20)

Kernel extension: Multiple synchronized address-space clones per process

- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2
- Kernel extension: Multiple synchronized address-space clones per process
- Address-Space Views (for Linux)
- Thread-Level Attack-Surface Reduction
- Address-Space Views
- shared mapping
- Address-Space Views (for Linux)
- copy-on-write mapping
- shared mapping
- create(
- new view
- text
- data
- heap
- migrate(
- th1
- th2

MariaDB: Per-Worker Views

```c
int main(...) {
    ...
    + cte_init();
    + cte_view_unshare();
    ...
}
```

```c
void do_handle_one_connection(...) {
    // mariadb thread/connection init
    ...
    while (thd_is_connection_alive(thd)) {
        do_command(thd);
    }
    // mariadb thread/connection cleanup
    ...
}
```

```c
int main(...) {
    ...
    + cte_init();
    + cte_view_unshare();
    ...
}
```

**mysql_thread_create(...)**

```c
+ long thread_view = view_create();
+ view_migrate(thread_view);
```

```c
+ cte_rules *R = cte_rules_init(CTE_WIPE);
+ cte_wipe(R);
```

```c
while (thd_is_connection_alive(thd)) {
    do_command(thd);
}
```

// load/kill/wipe (~policy) individual functions
c te_rules_set_func(R, policy, func_addr, recursive_callgraph);
c te_rules_set_fnmatch(R, policy, fnmatch_pattern, recursive_callgraph);
c te_rules_set_indirect(R, policy);
```

configure & perform wipe

create view & migrate

// load/kill/wipe (~policy) individual functions

Attack-Surface Reduction

- MariaDB *(client: all sysbench SQL benchmarks)*
  
  Original: 55614 functions (16.91 MiB)
  
  TLASR: min: 1822 functions (1.09 MiB) \(-97\% (-94\%)\) - `oltp_point_select`
  
  max: 2929 functions (1.65 MiB) \(-95\% (-90\%)\) - `oltp_read_write`

  *wiping per connection*

- memcached *(client: memtier benchmark)*

  Original: 5562 functions (2063 KiB)

  TLASR: min: 111 functions (46 KiB) \(-98\% (-98\%)\) - `slabs thread`
  
  max: 264 functions (115 KiB) \(-95\% (-94\%)\) - `worker thread`

  *wiping per thread*
Attack-Surface Reduction

Is an auto-ROP generator\(^1\) able to conduct ROP chain attacks?

- **MariaDB**
  - Original: yes
  - Static Debloating\(^2\): yes (-33% of functions)
  - TLASR: no (-95% to -97% of functions)

- **memcached**
  - Original: yes
  - Static Debloating\(^2\): yes (-65% of functions)
  - TLASR: no (-95% to -98% of functions)

\(^1\) ROPgadget tool: http://shell-storm.org/project/ROPgadget
\(^2\) Ziegler et al. '19
Performance Overhead

**memcached** (thread pool, CPU count)

**MariaDB** (thread per connection)
Performance Overhead

**memcached** (thread pool, CPU count)

**MariaDB** (thread per connection)

*re-wipe between transactions*
Summary

Thread-Level Attack-Surface Reduction (TLASR)

- **Goal:** Reduce attack surface of processes

- **Approach:** Context-based elimination of code
  - *On-demand* function elimination & restoration of the *whole process*
  - Works on *thread-level* via *address-space views* in Linux

- **Results:**
  - Reduced attack surface up to -98%
  - Auto-ROP utility turned ineffective with TLASR
  - Reasonable overhead