KSplit: Automating Device Driver Isolation

Yongzhe Huang¹, Vikram Narayanan², David Detweiler², Kaiming Huang¹, Gang Tan¹, Trent Jaeger¹, and Anton Burtsev²,³

¹Penn State University  ²University of California, Irvine  ³University of Utah
Driver vulnerabilities

- 16-50 % of all Linux kernel CVEs
VirtuOS: an operating system with kernel virtualization

Vikram N. Vats
University of Virginia
Sarah Spahni
University of Zurich

Tolering Malicious Device Drivers in Linux

Silas Boyd-Wickizer and Nickolai Zeldovich
MIT CSAIL

Final Report of the Multics Kernel Design Project

by

M.D. Schroer
D.B. Clark
J.H. Saltz
E.K. Wells

June 30, 1976

Lightweight Kernel Isolation with Virtualization and VM Functions

Matthew J. Renzelmann and Michael M. Swift
University of Wisconsin–Madison
{mjr, swift}@cs.wisc.edu

Decaf: Moving Device Drivers to a Modern Language

Microdrivers: A New Architecture for Device Drivers

V. Ganapathy, Arini Balakrishnan, Michael M. Swift
Computer Science Department, University of Washington

Nouks: An Architecture for Reliable Device Drivers

Steven Martin, Henry M. Levy, and Susan J. Eggers
Computer Science and Engineering
University of Washington

The SawMill Multiserver Approach

Alain Gehlaut, Trent Jaeger, Yoonho Park
Jochen Liedtke, Kevin J. Elphinstone, Volkan Uhlig
Driver Isolation Architecture

- Separate memory space

```c
int register_netdev(struct net_device *dev) {
    dev->features |= ...;
    ...
    return 0;
}
```

```c
int register_netdev(struct net_device *dev) {
    dev->features |= ...;
    dev->hw_features |= ...
}
```
Driver isolation architecture

- Separate memory space
- Two copies of object hierarchies
Driver isolation architecture

- Separate memory space
- Two copies of object hierarchies
- Keep them synchronized
Driver isolation architecture

- Separate memory space
- Two copies of object hierarchies
- Keep them synchronized
- Glue code
  - Marshal/unmarshal params
  - Interface definition language (IDL) spec
  - Generated with IDL compiler
Isolation performance

- Paging (834 cycles)
- Recent CPU mechanisms
  - VMFUNC - 396 cycles
  - MPK 11-260 cycles
  - Save/restore general/extended regs, pick a stack, etc.

Manually specifying the IDL for data synchronization between domains has become the major challenge
Challenge: Large interface boundary

- **Kernel**
  - `ixgbe_init_module()`
  - 134 kernel functions
    - `pci_register_driver(&ixgbe_driver)`

- **Driver**
  - 81 driver functions
    - `ixgbe_probe(struct pci_dev *dev, ...)`
Challenge: Complex data exchange

Kernel

ixgbe driver

send()

ixgbe_xmit_frame(struct sk_buff *skb, ...)

10
Challenge: Complex data structures

ixgbe_xmit_frame(struct sk_buff *skb, …)

- Represents a network packet
- Has 66 fields (5 pointers)
- 3,132 fields (1,214 pointers) are recursively reachable
- But only a small subset are accessed by both kernel and driver (shared)
  - 8 shared fields for this API
Challenge: Low-level kernel/C idioms

```c
int ixgbe_xmit_frame(struct sk_buff* skb, ...)
```

- Pointers
  - Singleton, array
  - Linked list
  - Collocated data structures
- Sized and sentinel arrays
- Special pointers (e.g., `__user`, `__iomem`)
- Tagged unions
- Return error as ptr (e.g., `ERR_PTR`)
Challenge: Concurrency primitives

- spin/mutex lock
- driver specific lock, e.g., rtnl_lock
- atomic operations, e.g., set_bit
- read-copy update (RCU)
- sequential lock
**KSplit goals**

- Build a set of static analyses to generate the IDL automatically (mostly) to
  - Isolate the complete driver
  - Identify shared/private data on the large interface boundary
  - Ensure each domain has the updated copy of the data structure
  - Identify marshaling requirements for the low-level kernel idioms
  - Identify atomic regions that access shared data
- Prior work
  - *Microdrivers* (isolated the control plane of the driver)
KSplit design choices

- Kernel is huge
  - Identify the relevant kernel code that the driver interacts with
- Aim to detect all shared data (sound)
  - We might classify some private data as shared
- Aim to infer marshaling requirements for low-level idioms
  - Provide warning for the cases that we cannot infer
- Aim to infer marshaling requirements for shared critical sections
  - Hypothesis: There are not many shared critical sections
KSplit workflow

- **Input**: source code of kernel and target isolated driver
- **Output**: IDL file that specifies the communication interfaces and data synchronization requirements
Shared field analysis
Shared field analysis

- **Input:**
  - data structure types on all the interface functions for the driver under analysis

- **Output:**
  - the set of struct fields accessed by both the kernel and this driver
Shared field analysis

```c
int register_netdev(struct net_device *dev) {
    dev->hw_features |= NETIF_F_SOFT_FEATURES;
    dev->features |= NETIF_F_SOFT_FEATURES;
    dev->wanted_features = dev->features & dev->hw_features;
}
```

```c
static int ixgbe_probe(struct pci_dev* pdev const struct pci_device_id *ent) {
    struct net_device *netdev = alloc_etherdev_mq(...);
    // initialize netdev struct fields
    netdev->hw_features |= NETIF_F_RXALL;
    netdev->features |= NETIF_F_HW_VLAN_CTAG_FILTER;
    register_netdev(netdev);
}
```
Program Dependence Graph

- **PDG**: represents program dependencies
  - inter-procedural pointer alias relations
  - field-sensitive
  - data dependencies
  - control dependencies/flow

**PtrSplit: Supporting General Pointers in Automatic Program Partitioning**

Shen Liu  
The Pennsylvania State University  
University Park, PA  
sxl465@cse.psu.edu

Gang Tan  
The Pennsylvania State University  
University Park, PA  
gtan@cse.psu.edu

Trent Jaeger  
The Pennsylvania State University  
University Park, PA  
tjaeger@cse.psu.edu
Boundary data access analysis
Boundary data access analysis

• **Purpose:**
  • Infer synchronization requirements for every interface call and return

• **How:**
  • figure out the subset of shared fields that are read/written in an interface function
  • synchronize data **read by callee** at the function call
  • synchronize data **updated by callee** at the function return
Boundary Data Access Analysis: example

```c
/* kernel code */
int register_netdev(struct net_device *dev) {
    dev->features |= NETIF_F_SOFT_FEATURES;
    dev->wanted_features = dev->features & dev->hw_features;
    if (dev->hw_features & NETIF_F_TSO)
        dev->hw_features |= NETIF_F_TSO_MANGLED;
    ...
    return ret;
}
```

```c
/* ixgbe driver */
int ixgbe_probe(struct pci_dev *pdev, ...) {
    struct net_device *dev = alloc_etherdev(...);
    /* initialize struct fields */
    dev->features |= NETIF_F_GSO_PARTIAL;
    dev->hw_features |= NETIF_F_HW_TC;
    register_netdev(dev);
    ...
}
```

Atomic Region Analysis
Atomic Region Analysis

• **Purpose:**
  - Find shared data accessed within the atomic regions
  - Infer synchronization requirements for shared atomic regions

• **When to synchronize:**
  - after/before the **entry/exit** of each atomic region.

```
spin_lock(&lock)
sync_call(read data)
...
sync_call(updated data)
spin_unlock(&lock)
```
**Atomic Region Analysis**

- Compute atomic regions using control flow graph

```cpp
spin_lock(&lock)
... 
spin_unlock(&lock)
```
Infer marshaling requirements for pointers

kernel | ixgbe driver

--- system call

ixgbe_xmit_frame(struct sk_buff *skb, ...)

point to a single instance or an array?
Classify Pointers with Nescheck

CCured: Type-Safe Retrofitting of Legacy Code

George C. Necula, Scott McPeak, Westley Weimer
University of California, Berkeley
{necula,smcpeak,weiner}@cs.berkeley.edu

POPL’2002

Memory Safety for Embedded Devices with nesCheck

Daniele Midi
Dept. of Computer Science
Purdue University
West Lafayette, IN, USA
dmidi@purdue.edu

Mathias Payer
Dept. of Computer Science
Purdue University
West Lafayette, IN, USA
mpayer@purdue.edu

Elisa Bertino
Dept. of Computer Science
Purdue University
West Lafayette, IN, USA
bertino@purdue.edu

AsiaCCS’17
Classify Pointers with Nescheck

- **safe**: synchronized by supplying correct type
- **seq**: require size and correct type
- **wild**: casted to other types
- infer size
  - generate array annotation with size info
  - IDL warning
- is void pointer?
  - IDL warning
  - only casted to one type?
    - IDL warning
    - infer type
Evaluation

- **Research questions:**
  - How much data synchronization can we reduce?
  - How much manual work required?
  - How to test correctness of the isolated drivers?
- Compare to *Microdrivers*
- Run KSplit on 354 drivers from 9 subsystems
- Fully isolate and 10 drivers and validate the correctness

- **Performance Overhead:**
  - Memcached benchmark
Case study: Ixgbe driver
Ixnge: data synchronization optimization

Deep copy fields: 999,000
Microdrivers shared fields: 4,238
KSplit shared fields: 3,146
Ixgbe: synchronization primitives

Critical Sections
- Private: 70
- Shared: 35

RCU
- Private: 33
- Shared: 35

Seqlock
- Private: 0
- Shared: 70

Atomic Operations
- Private: 173
- Shared: 35

private and shared kernel patterns
# Ixgbe: pointer classification

<table>
<thead>
<tr>
<th></th>
<th>singleton</th>
<th>array</th>
<th>string</th>
<th>wild pointer (void)</th>
<th>wild pointer (other)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>manual</strong></td>
<td>0</td>
<td>27</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td><strong>handled</strong></td>
<td>1261</td>
<td>92</td>
<td>2</td>
<td>142</td>
<td>1</td>
</tr>
</tbody>
</table>
Ixgbe: Manual work

- Source code - 27,000 lines
- Generated IDL spec - 2000 lines
- Pointer misclassifications - 7
- Warnings - 65 (33 anonymous unions, 16 arrays, wild pointers)
  - IDL (changes) - 53 lines
  - Driver (changes) - 19 lines
Manual Work (average across isolated drivers)

- Warnings: 16
- Pointer Misclassification: 2
- IDL changes (line): 14
- Driver code changes (line): 6
Performance overhead: memcached

- Memcached/memaslap
- 64B keys, 1024B values (90% set, 10% get)
- We report the bandwidth and transactions per second
- For 1-4 threads, KSsplit overhead (5-18%)
- With 10 threads, we saturate the network bandwidth
Conclusions

- We are moving closer to low-overhead isolation mechanisms
- Complexity of isolation becomes a major challenge
- Static analysis framework with small manual effort

The source code is available at: https://github.com/ksplit/ksplit-artifacts
Thank you