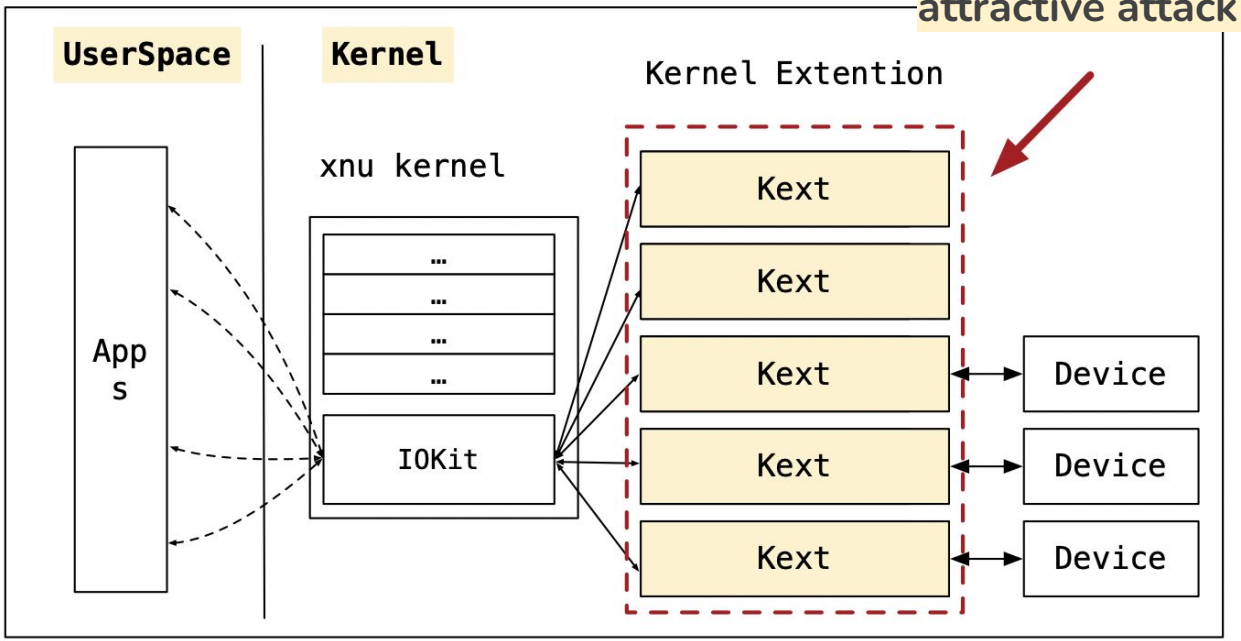
KextFuzz: Fuzzing macOS Kernel EXTensions on Apple Silicon via Exploiting Mitigations

Tingting Yin[1,3], *Zicong Gao*[4], *Zhenghang Xiao*[5], *Zheyu Ma*[1], *Min Zheng*[3], *Chao Zhang*[1,2*]

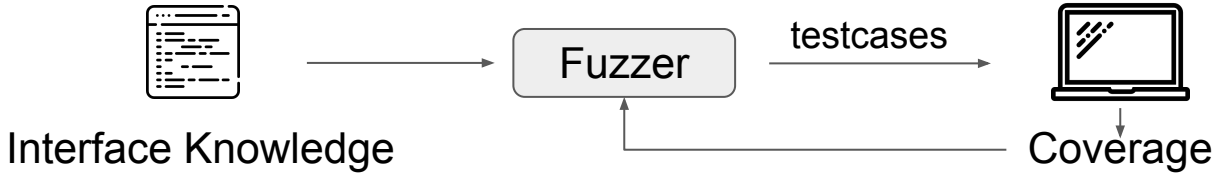
[1]*Tsinghua University* [2]*Zhongguancun Laboratory* [3]*Ant Group* [5]*Hunan University*
[4]*State Key Laboratory of Mathematical Engineering and Advanced Computing*

Background

Kernel Extensions are attractive attack surfaces



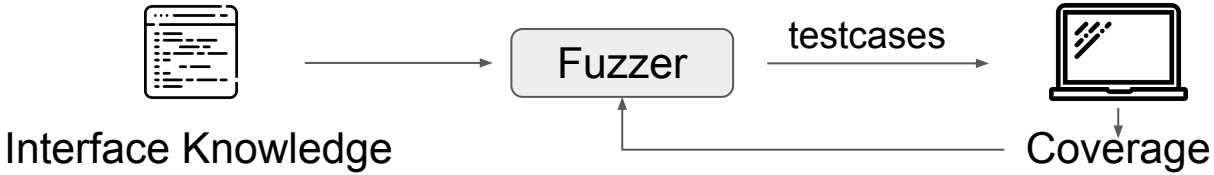
Driver Fuzzing



Source Code Analysis
(e.g. DIFUZE, Moonshine, HFL, KSG, …)

Source-code Level Instrumentation
(e.g. coverage: kcov)

Closed-source Driver Fuzzing



Source Code Analysis
(e.g. DIFUZE, Moonshine, HFL, KSG, …)

Source-code Level Instrumentation
(e.g. coverage: kcov)

Closed-Source

Interaction Trace
(e.g. SyzGen, IMF)

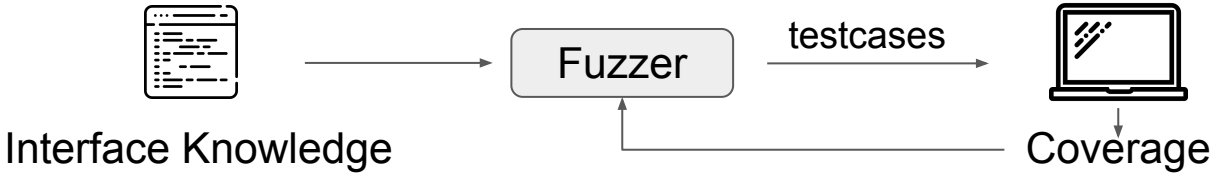
Hardware
(e.g. Intel-PT)

Header File & WIN32
(e.g. NTFuzz)

Hypervisor
(e.g. Digtool)

Kernel Debugger
(e.g. LLDB-Fuzzer, SyzGen)

macOS Driver Fuzzing



Source Code Analysis
(e.g. DIFUZE, Moonshine, HFL, KSG, …)

Interaction Trace
(e.g. SyzGen, IMF)

Header File & WIN32
(e.g. NTFuzz)

Poor Toolchain Support

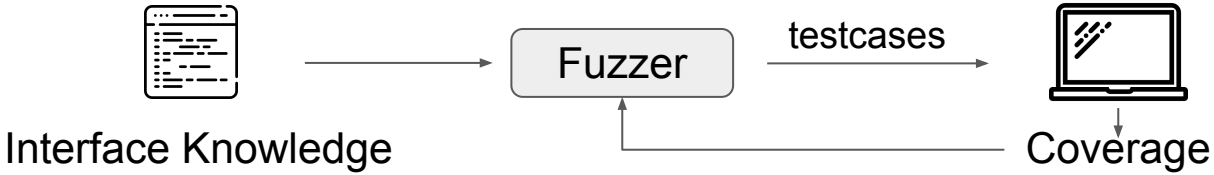
Source-code Level Instrumentation
(e.g. coverage: kcov)

Hardware
(e.g. Intel-PT)

Hypervisor
(e.g. Digtool)

Kernel Debugger
(e.g. LLDB-Fuzzer, SyzGen)

macOS Driver Fuzzing



Source Code Analysis
(e.g. DIFUZE, Moonshine, HFL, KSG, …)

Interaction Trace
(e.g. SyzGen, IMF)

Header File & WIN32
(e.g. NTFuzz)

:(

:(

Foundamental Limitaions

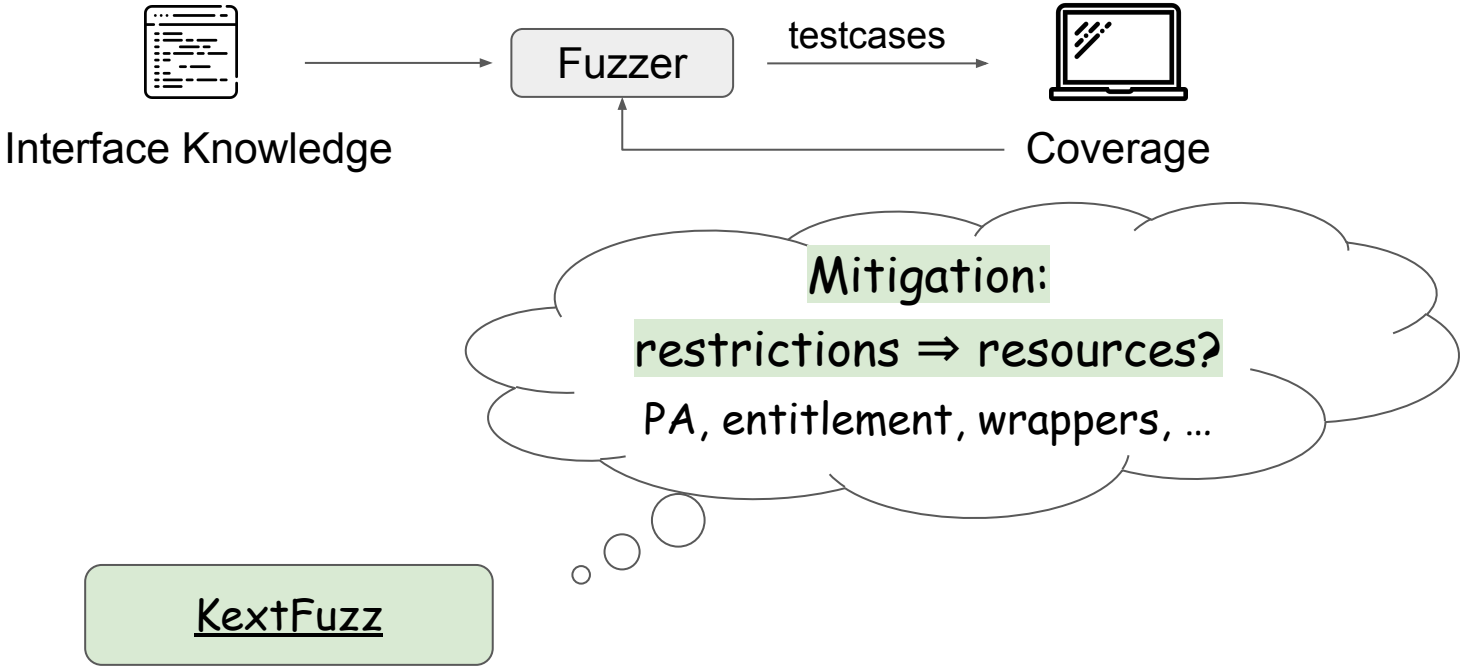
Source-code Level Instrumentation
(e.g. coverage: kcov)

Hardware
(e.g. Intel-PT)

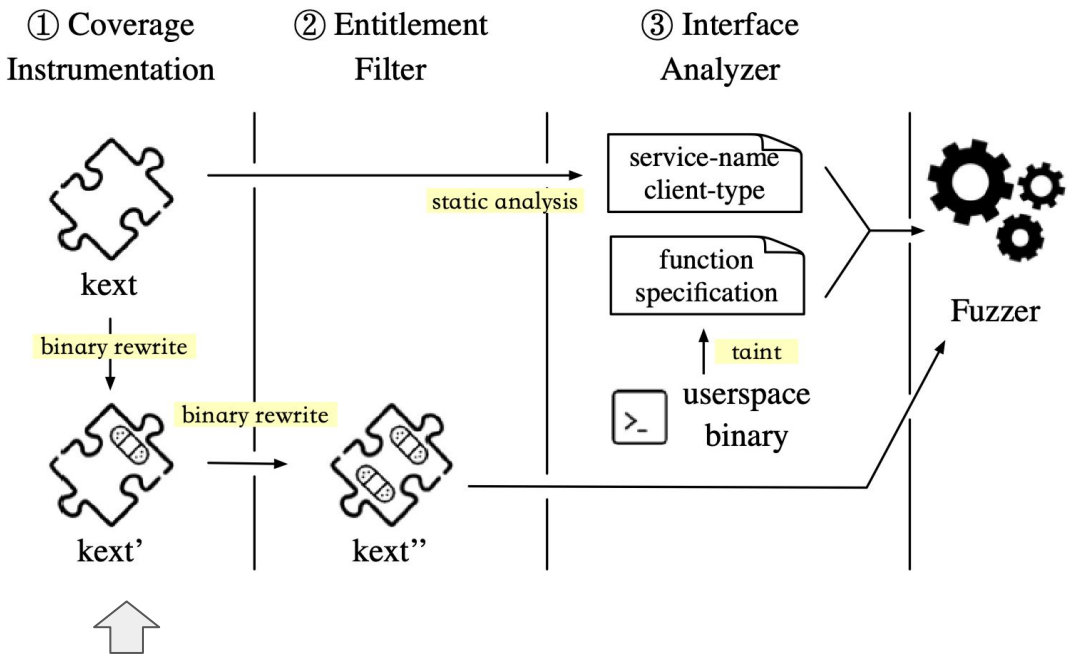
Hypervisor
(e.g. Digtool)

Kernel Debugger
(e.g. LLDB-Fuzzer, SyzGen)

How to Make Fuzzing Greate Again?



KextFuzz

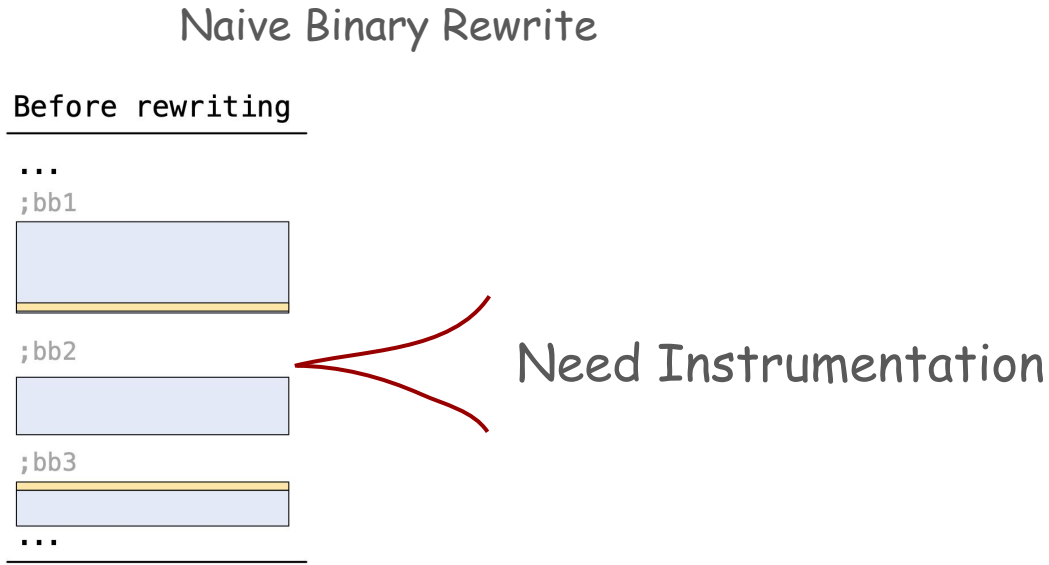


1. Pointer Authentication Mitigation

⇒ binary level instrumentation

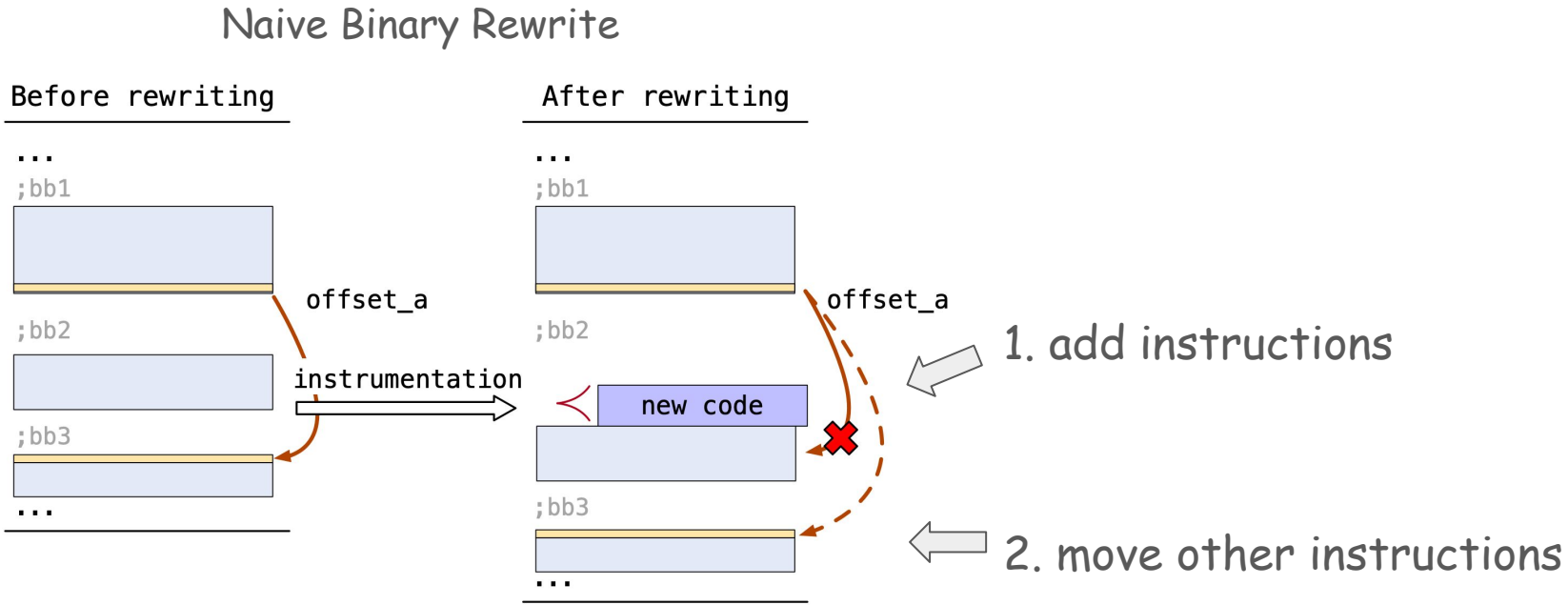
KextFuzz - Coverage Collector

Q: How to do binary level instrumentation in kexts?



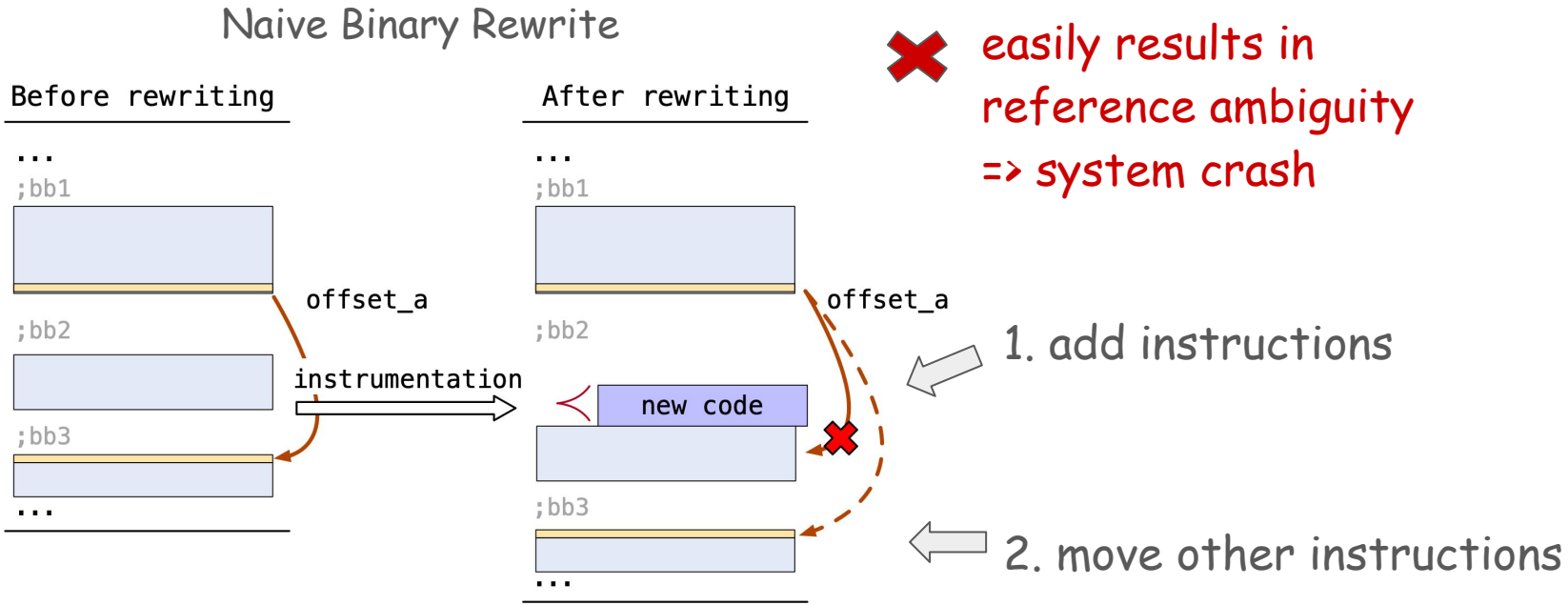
KextFuzz - Coverage Collector

Q: How to do binary level instrumentation in kexts?



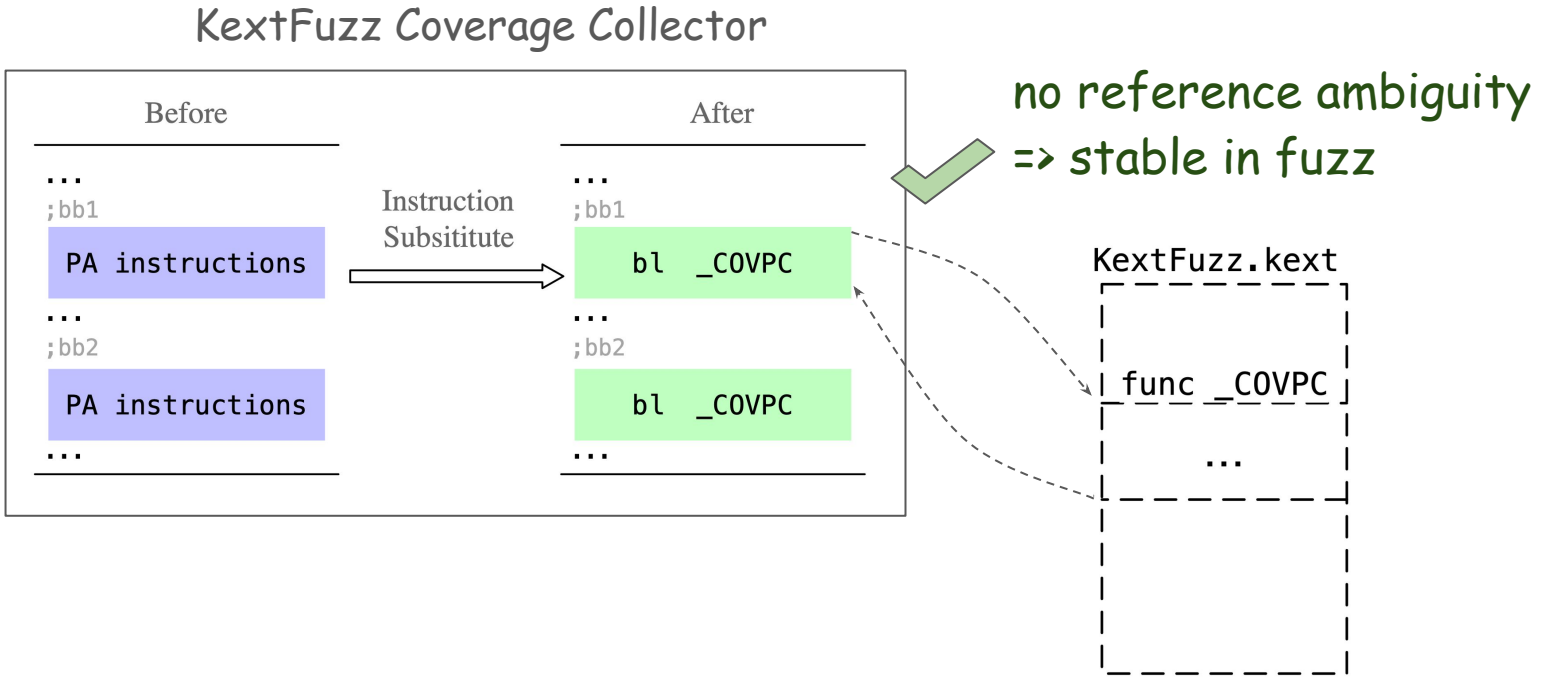
KextFuzz - Coverage Collector

Q: How to do binary level instrumentation in kexts?

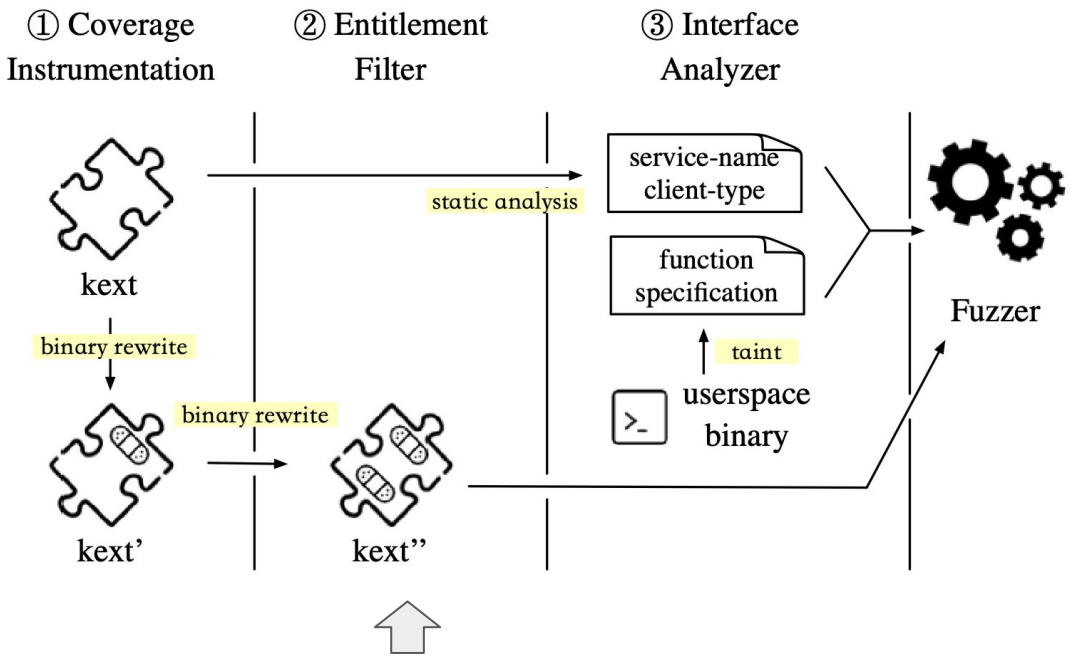


KextFuzz - Coverage Collector

Q: How to do binary level instrumentation in kexts?



KextFuzz



2. Remove Entitlement Mitigation

⇒ testing privileged code

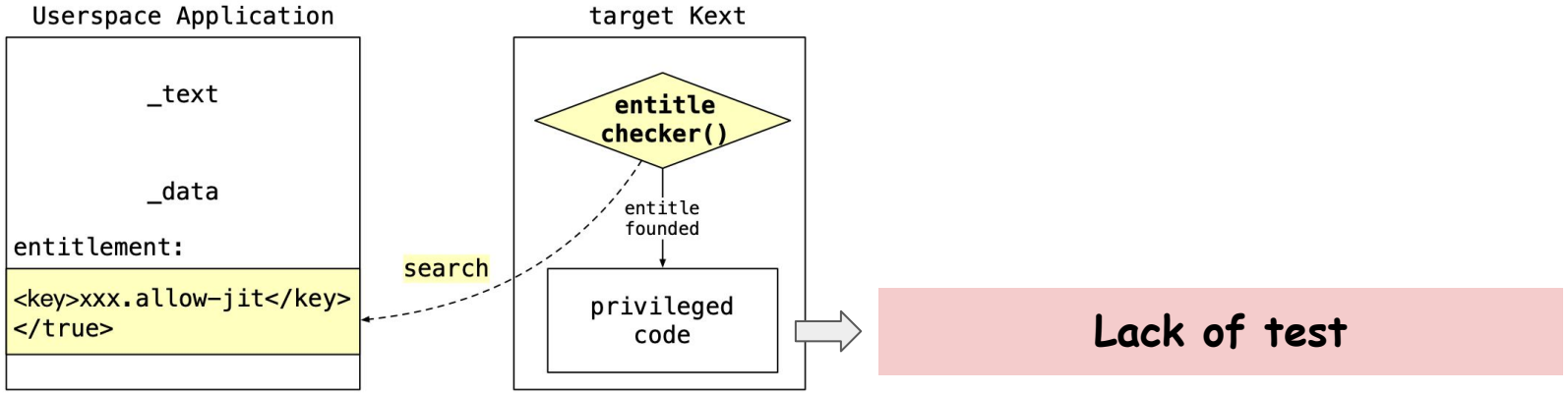
KextFuzz - Entitlement Filter

- What is the Entitlement?

1. Capabilities that hard-coded in binary code signature.

2. Kexts check entitlements to restrict applications invoking privileged code.

=> leaving privileged code lack of testing.



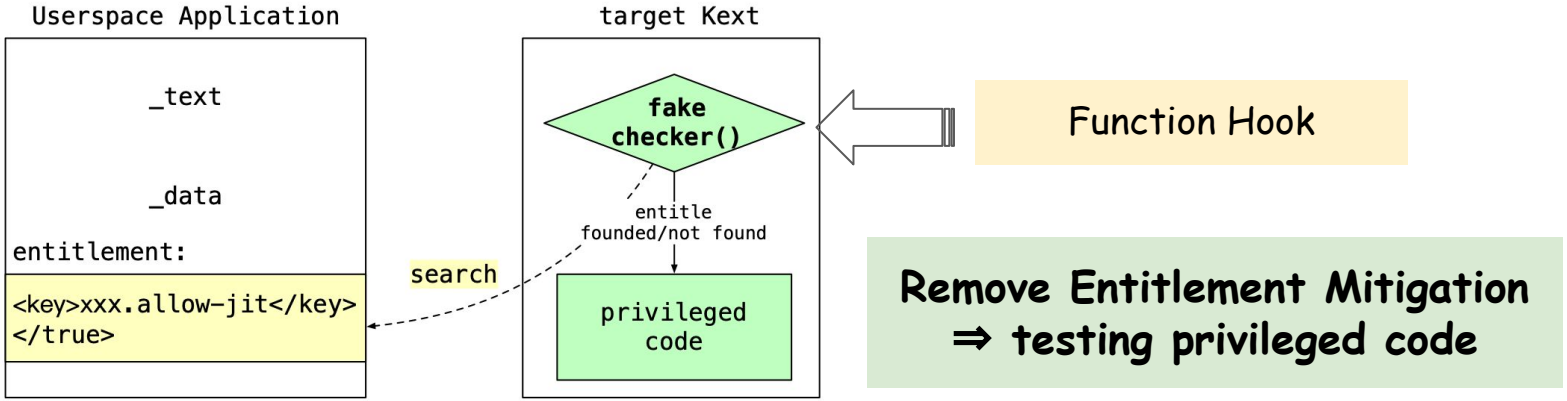
KextFuzz - Entitlement Filter

- What is the Entitlement?

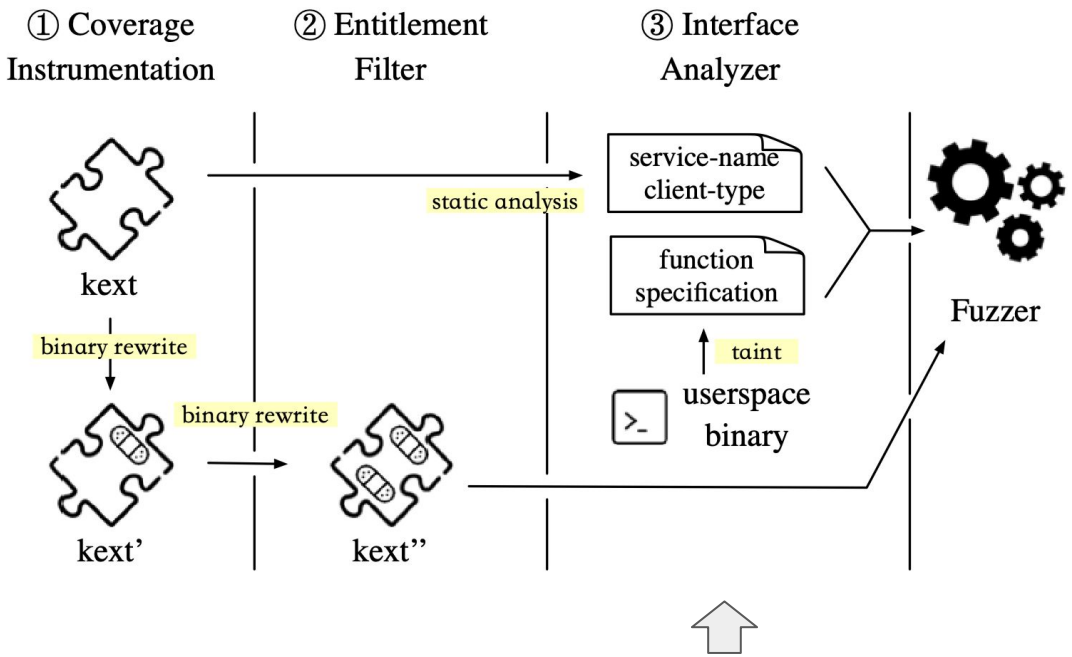
1. Capabilities that hard-coded in binary code signature.

2. Kexts check entitlements to restrict applications invoking privileged code.

=> leaving privileged code lack of testing.



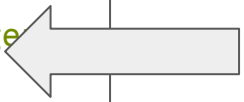
KextFuzz



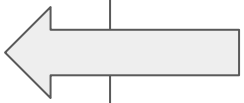
3. Kernel function isolation layer
⇒ Interface knowledge

KextFuzz - Interface Indentification

```
*** step1: create surface ***
input_1 = "<dict>
              <key>IOSurfaceWidth</key>
              <integer size=\"32\">0x40</intege
              <key>IOSurfaceIsGlobal</key>
              <true/>
              ...
           </dict>"
IOConnectCallMethod(conn, 0, input_1, ..., output);
int surface_id = output[0];

*** step2: set value ***
input_2[0] = surface_id;
value = "<array>
            <string>kCGColorSpaceSRGB</string>
            <string>...</string>
         </array>";
memcpy(input_2 + 8, value)
IOConnectCallMethod(conn, 9, input_2);
```



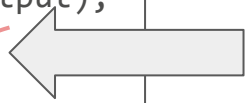
Corefoundation Dictionary



Corefoundation Array

KextFuzz - Interface Indentification

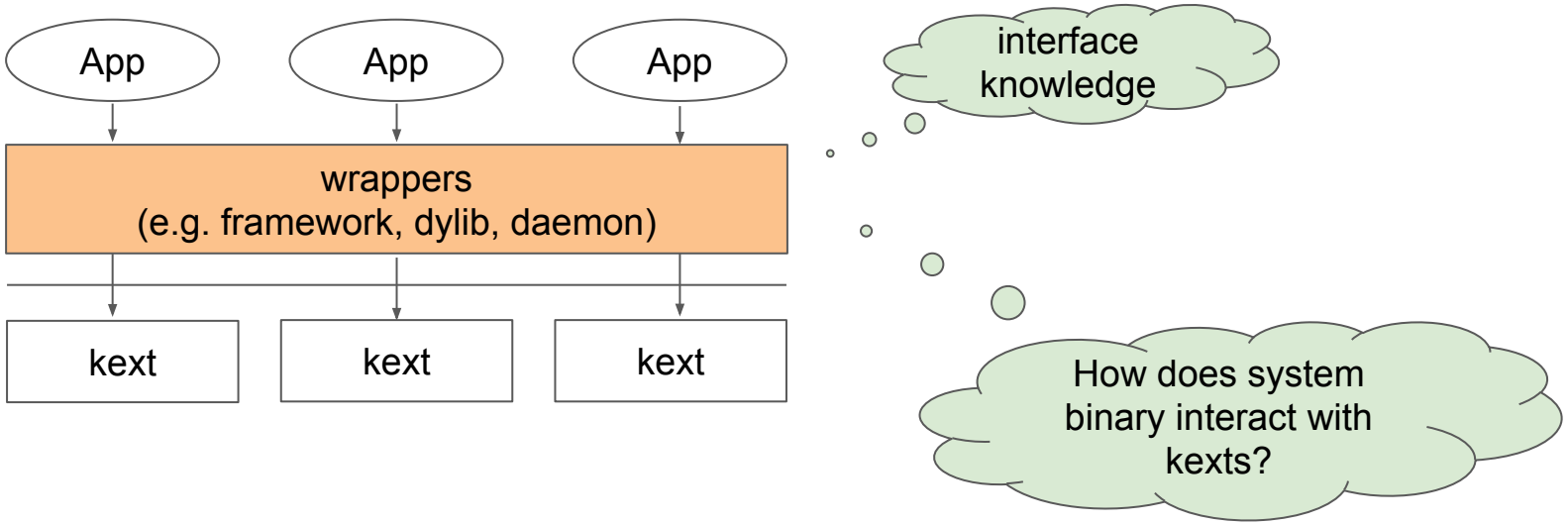
```
*** step1: create surface ***
input_1 = "<dict>
              <key>IOSurfaceWidth</key>
              <integer size=\"32\">0x40</integer>
              <key>IOSurfaceIsGlobal</key>
              <true/>
              ...
           </dict>"
IOConnectCallMethod(conn, 0, input_1, ..., output);
int surface_id = output[0];

*** step2: set value ***
input_2[0] = surface_id;
value = "<array>
            <string>kCGColorSpaceSRGB</string>
            <string>...</string>
         </array>";
memcpy(input_2 + 8, value)
IOConnectCallMethod(conn, 9, input_2);
```



Resource Variable
surface_id

KextFuzz - Interface Indentification

macOS uses userspace wrappers to reduce direct kext invocations.



KextFuzz - Interface Indentification

KextFuzz: light-weight taint analysis

Taint Source: Type, Value

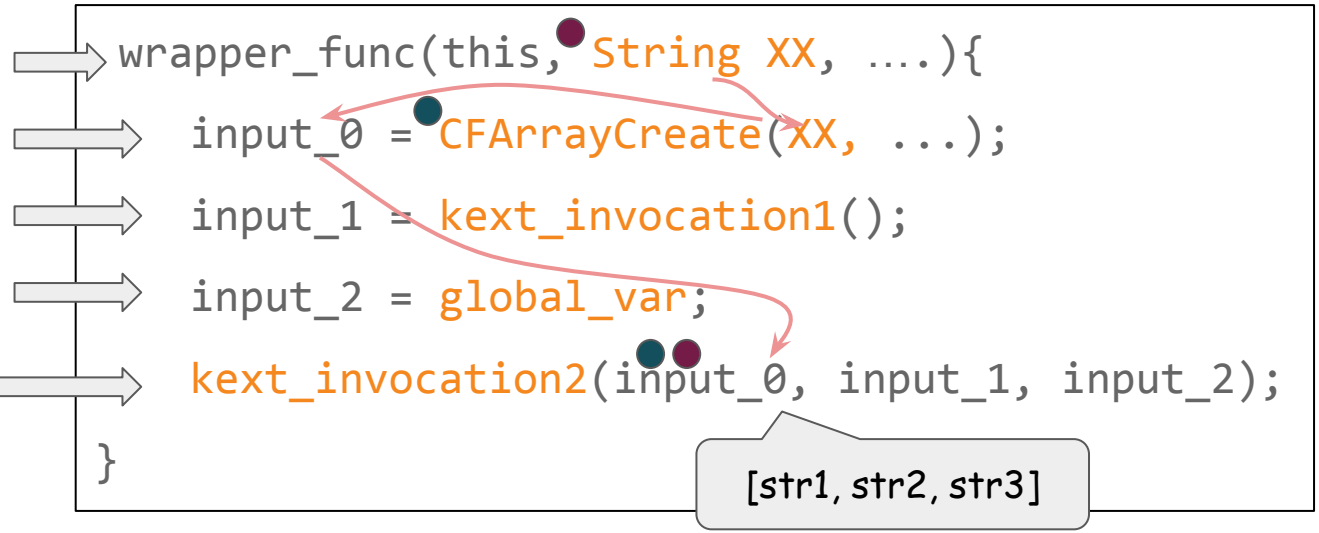
s1: caller argument

s2: creation function

s3: output

s4: global variable

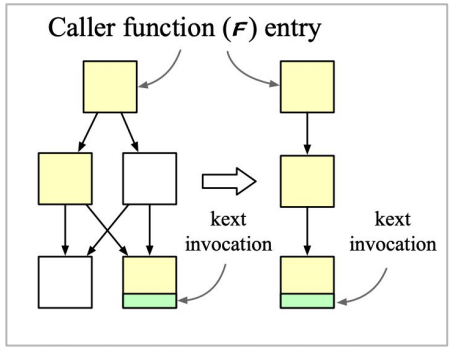
Taint Sink:



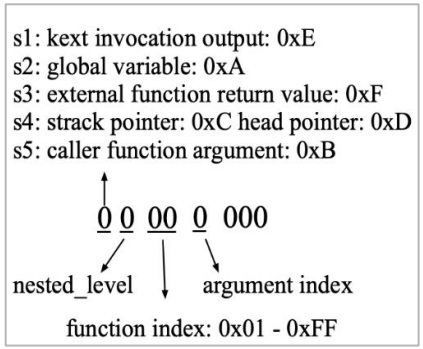
KextFuzz - Interface Indentification

KextFuzz: light-weight taint analysis

- step1: extract kext invocation related code from wrappers.
- step2: initial the memory and argument registers with taint tags.
- step3: emulation execution



code snippets

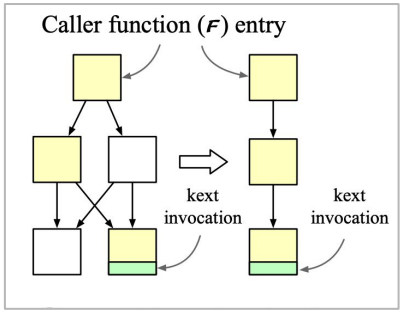


encoded taint information

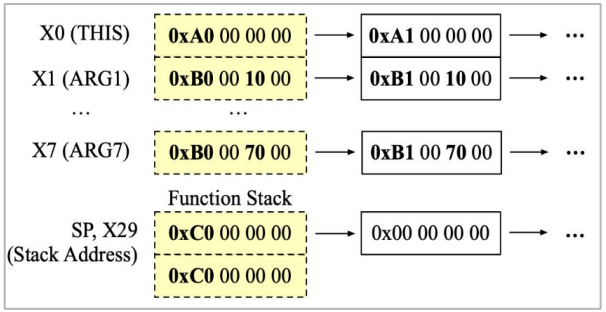
KextFuzz - Interface Indentification

KextFuzz: light-weight taint analysis

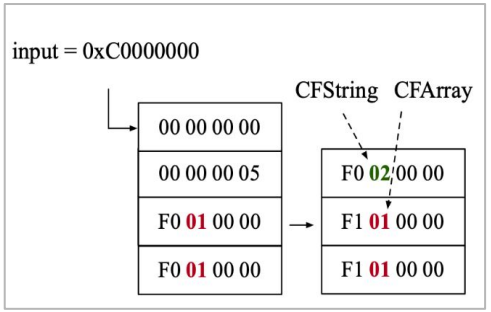
- step1: extract kext invocation related code from wrappers.
- step2: initial the memory and argument registers with taint tags.
- step3: emulation execution



code snippets



initial state



tainted invocation argument

KextFuzz - Evaluation

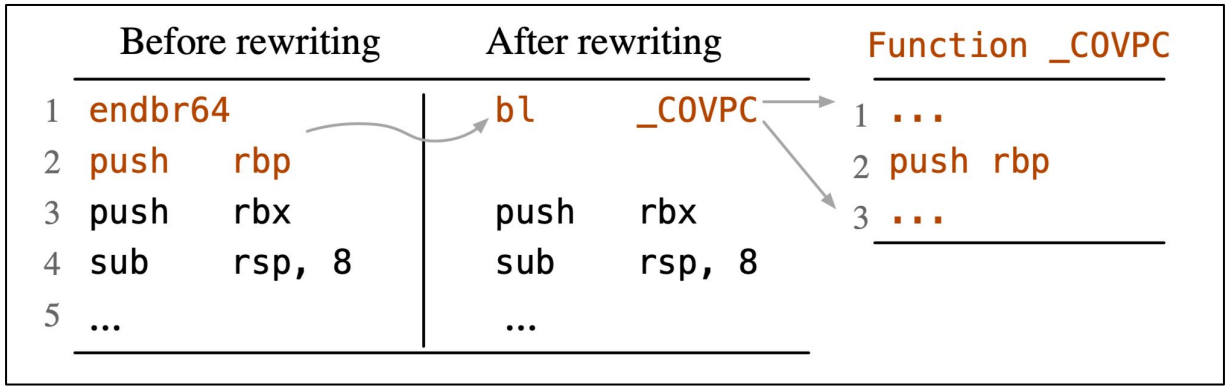
KextFuzz - Coverage Collector

Instruments 34.71% basic blocks with 2.03x overhead

kext	instrumented	cov-aware	overhead
IOSurface	26.86%	32.09%	3.23x
IOMobileGraphicsFamily-DCP	24.09%	30.10%	3.74x
AppleH13CameraInterface	35.81%	38.63%	4.74x
AUC	28.36%	35.45%	3.76x
IONetworkingFamily	31.88%	37.35%	1.40x
AppleBCMWLANCore	16.19%	18.98%	1.02x
AppleIPAppender	33.80%	41.59%	2.29x
IOUSBHostFamily	33.20%	35.88%	2.24x
IOUSBDeviceFamily	32.70%	37.62%	2.57x
IOAudioFamily	37.81%	41.65%	1.17x
IOAVBFamily	75.26%	78.95%	-
AppleAOPVoiceTrigger	49.91%	55.22%	0.96x
AppleMultitouchDriver	37.74%	41.98%	2.78x
IOHIDFamily	34.84%	39.42%	1.37x
EndpointSecurity	18.44%	25.44%	1.07x
AppleBluetoothDebug	38.80%	43.82%	0.85x
AppleBluetoothModule	22.66%	28.05%	0.97x
IOBluetoothFamily	31.89%	34.99%	0.76x
IOReportFamily	49.23%	51.69%	1.62x
Average	34.71%	39.42%	2.03x

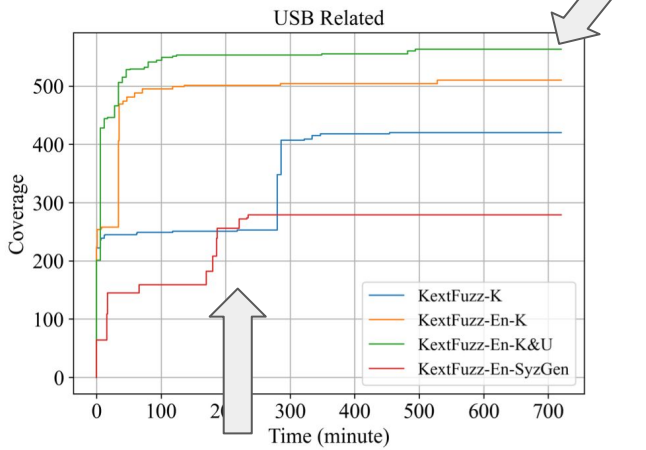
KextFuzz - Coverage Collector

Instrument ELF binaries by replacing CET & Canary instructions



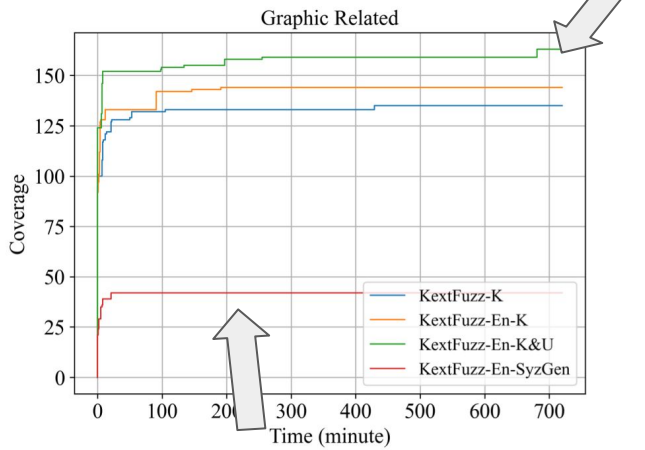
KextFuzz - Interface Identifier

KextFuzz



SyzGen

KextFuzz



SyzGen

KextFuzz - Bug Finding

- Finds 48 unique kernel crashes.
  - Five of them get CVEs.
  - Three of them get bounties.
- **Coverage Collector**: 6 times more bugs in 24 hours compared with black box fuzzing
- **Interface analyzer**: finds two complex bugs and finds two bugs faster
- **Entitlement Filter**: finds 18 more bugs in the privileged code
- Fuzzing in Apple Silicon macOS: find 13 bugs in arm only kexts.

• • •

Take aways

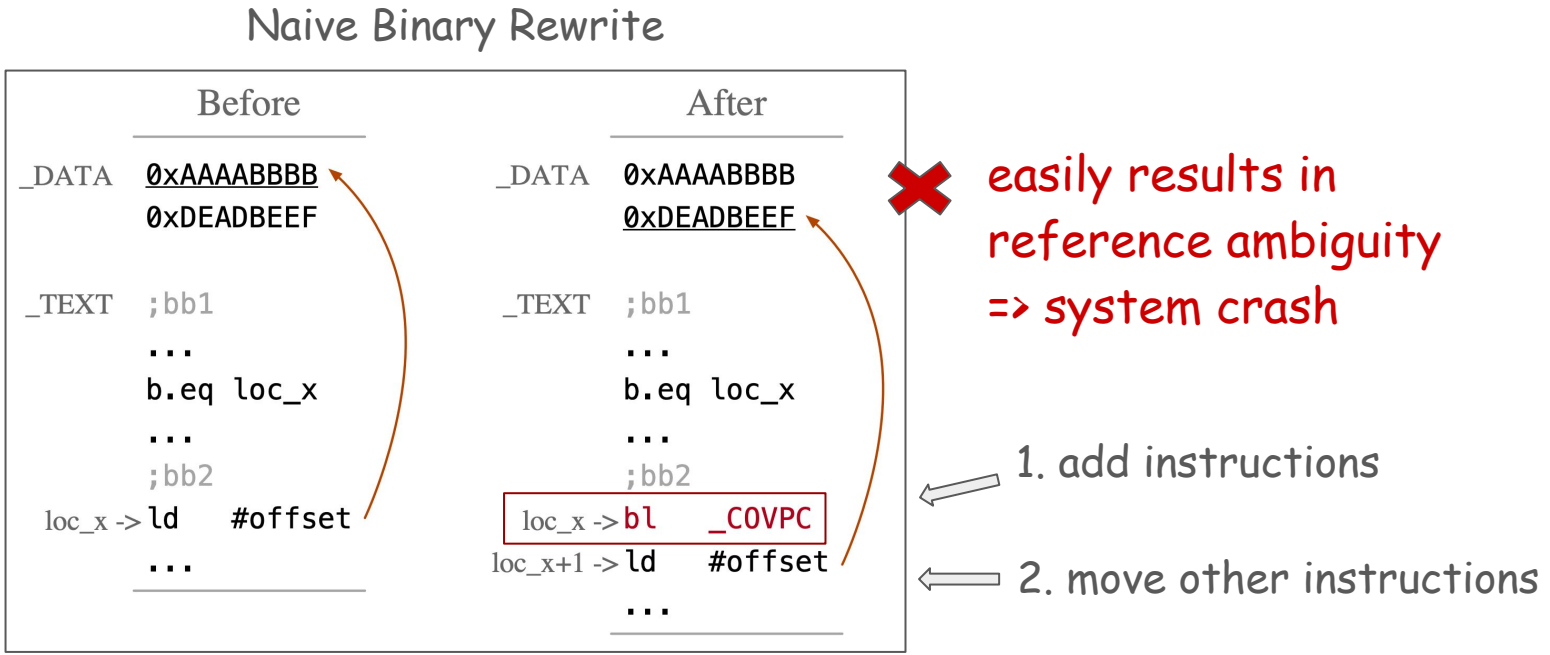
- **KextFuzz**: a fuzzer does not needs source code, traces, hardware support, and hypervisors.
- Removing mitigation instructions can release space for instrumentation.
- Removing privilege check enrich code can be tested.
- Interface information can be collected from the code calling them.

Thanks for listening!
Q & A

Contact: Tingting Yin ttea.yin@gmail.com

KextFuzz - Coverage Collector

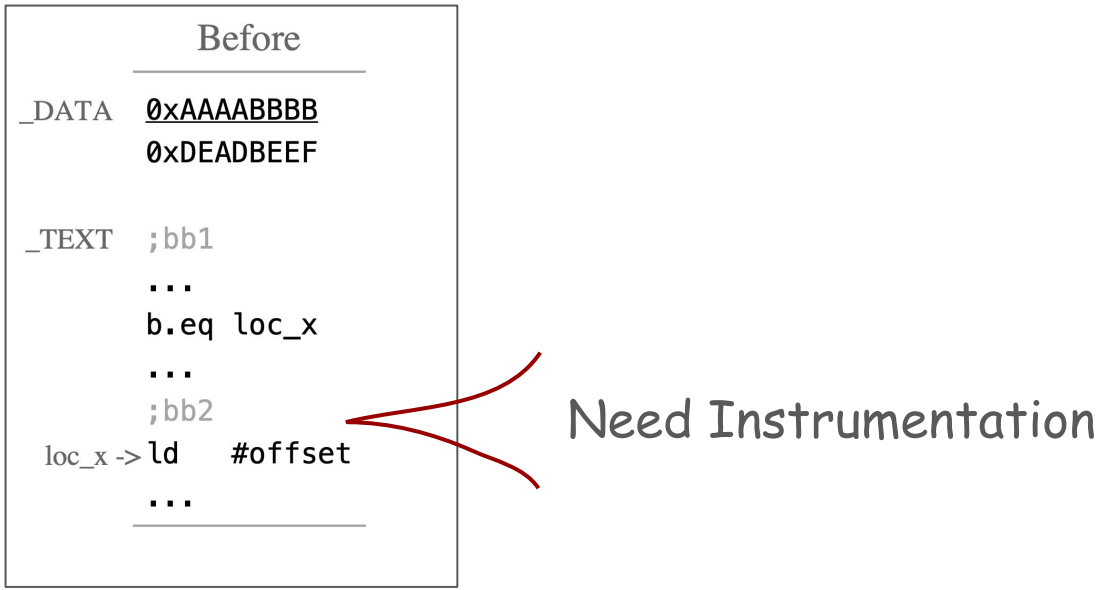
Q: How to do binary level instrumentation in kexts?



KextFuzz - Coverage Collector

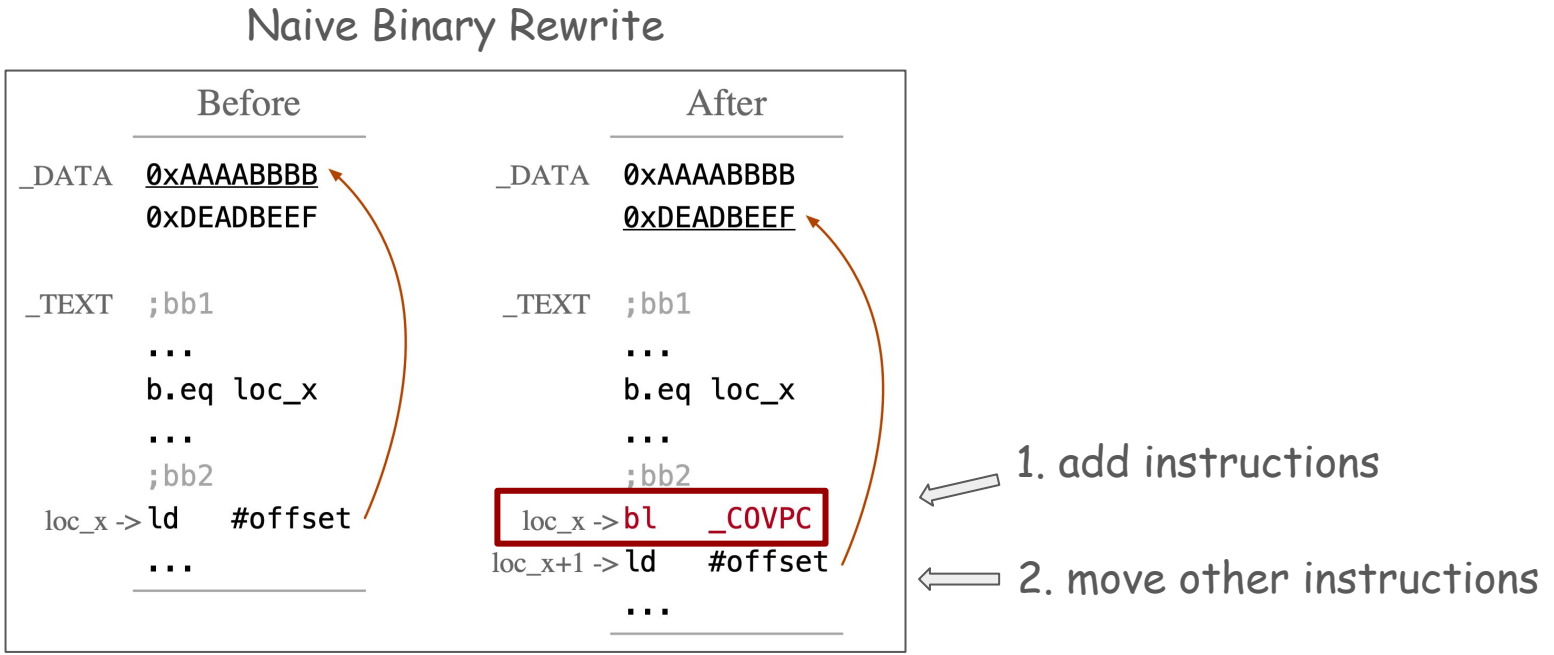
Q: How to do binary level instrumentation in kexts?

Naive Binary Rewrite



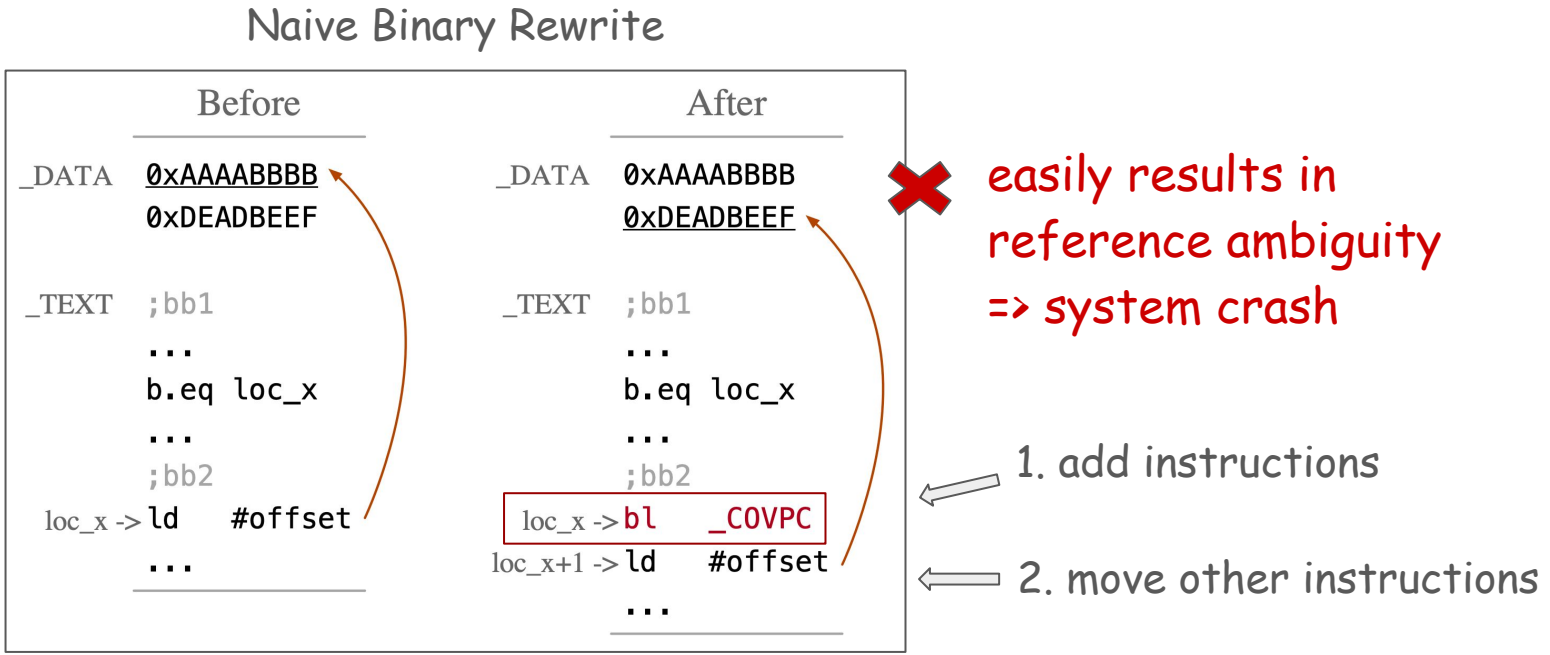
KextFuzz - Coverage Collector

Q: How to do binary level instrumentation in kexts?



KextFuzz - Coverage Collector

Q: How to do binary level instrumentation in kexts?



KextFuzz - Bug Finding

Case Study:

- calling interface1 (createController) to get the controller id with XML input
- calling interface2 (setMask) to trigger the bug

```
1  client::createController(client* this, void* input){
2    if (this->controller){ return ERROR; }
3    v0 = OSUnserializeXML(input, ...);
4    properties = TypeCast(v0, OSDictionary::metaClass);
5    con = create_controller(properties);
6    if (con){ this->controller = con; }
7  }
8  client::setMask(client* this, void* input){
9    if (!this->controller){ /* vulnerable code */ }
10 }
```

Listing 2: An example of the bug found by KextFuzz

KextFuzz - Coverage Collector

Q: How to do binary level instrumentation in kexts?