Reshaping Input Spaces to Fuzz Complex Targets

Alexander Bulekov **BU Seclab** ECE Department PhD Thesis Defense March 19, 2024

Thesis Committee: Prof. Manuel Egele Prof. Gianluca Stringhini Prof. Orran Krieger **Prof. Mathias Payer**



1

VULNERABILITIES

Firefox 116 Patches High-Severity Vulnerabilities

Firefox 116 was released with patches for 14 CVEs, including nine high-severity vulnerabilities, some of which can lead to remote code execution or sandbox escapes.



By Ionut Arghire August 2, 2023



1

VULNERABILITIES

Firefox 116 Patches High-Severity Vulnerabilities

Firefox 116 was released with patches for 14 CVEs, including nine high-severity vulnerabilities, some of which can lead to remote code execution or sandbox escapes.



onut Arghire



linux-cve-announce.vger.kernel.org archive mirror search help / color / mirror / Atom feed CVE

Search results ordered by [date|relevance] view[summary|nested|Atom feed] download mbox.gz: results only | full threads

1. CVE-2024-26626: ipmr: fix kernel panic when forwarding mcast packets - by Greg Kroah-Hartman @ 2024-03-06 6:46 UTC [4%]

2. CVE-2024-26628: drm/amdkfd: Fix lock dependency warning - by Greg Kroah-Hartman @ 2024-03-06 6:46 UTC [6%]

3. CVE-2024-26627: scsi: core: Move scsi_host_busy() out of host lock for waking up EH handler - by Greg Kroah-Hartman @ 2024-03-06 6:46 UTC [6%]

1

4. CVE-2023-52592: libbpf: Fix NULL pointer dereference in bpf_object__collect_prog_relos - by Greg Kroah-Hartman @ 2024-03-06 6:45 UTC [6%]

5. CVE-2023-52591: reiserfs: Avoid touching renamed directory if parent does not change - by Greg Kroah-Hartman @ 2024-03-06 6:45 UTC [6%]

6. CVE-2023-52590: ocfs2: Avoid touching renamed directory if parent does not change - by Greg Kroah-Hartman @ 2024-03-06 6:45 UTC [6%]

VULNERABILITIES

Firefox 116 Patches High-Severity Vulnerabilities

Firefox 116 was released with patches for 14 CVEs, including nine high-severity vulnerabilities, some of which can lead to remote code execution or sandbox escapes.



onut Arghire ugust 2, 2023



linux-cve-announce.vger.kernel.org archive mirror search help / color / mirror / Atom feed CVE

Search results ordered by [date|relevance] view[summary|nested|Atom feed] download mbox.gz: results only | full threads



1. CVE-2024-26626: ipmr: fix kernel panic when forwarding mcast packets - by Greg Kroah-Hartman @ 2024-03-06 6:46 UTC [4%]

2. CVE-2024-26628: drm/amdkfd: Fix lock dependency warning - by Greg Kroah-Hartman @ 2024-03-06 6:46 UTC [6%]

3. CVE-2024-26627: scsi: core: Move scsi_host_busy() out of host lock for waking up EH handler - by Greg Kroah-Hartman @ 2024-03-06 6:46 UTC [6%]

4. CVE-2023-52592: libbpf: Fix NULL pointer dereference in bpf_object__collect_prog_relos - by Greg Kroah-Hartman @ 2024-03-06 6:45 UTC [6%]

5. CVE-2023-52591: reiserfs: Avoid touching renamed directory if parent does not change - by Greg Kroah-Hartman @ 2024-03-06 6:45 UTC [6%]

6. CVE-2023-52590: ocfs2: Avoid touching renamed directory if parent does not change - by Greg Kroah-Hartman @ 2024-03-06 6:45 UTC [6%]

VULNERABILITIES

Firefox 116 Patches High-Severity Vulnerabilities

Firefox 116 was released with patches for 14 CVEs, including nine high-severity vulnerabilities, some of which can lead to remote code execution or sandbox escapes.



```
onut Arghire
ugust 2, 2023
```





Search results ordered by [date|relevance] view[summary|nested|Atom feed] download mbox.gz: results only | full threads

Bosch Smart Thermostat Feels the Heat From Firmware Bug The vulnerability in a popular hospitality industry gadget allows attackers to take over the device, pivot into the user's network, or brick the device entirely, rendering HVAC unusable. Nathan Eddy, Contributing Writer ③ 3 Min Read anuary 16, 2024 BOSCH Ξ 11111 12:00 pm 72 Until 12:00pm 70° 78° 40%

POTUS executive order aims to keep U.S. ports safe from cyberattacks

Steve Zurier February 21, 2024



port of seattle with downtown skyline early morning

linux-cve-announce.vger.kernel.org archive mirror search help / color / mirror / Atom feed

1. CVE-2024-26626: ipmr: fix kernel panic when forwarding mcast packets - by Greg Kroah-Hartman @ 2024-03-06 6:46 UTC [4%]

2. CVE-2024-26628: drm/amdkfd: Fix lock dependency warning - by Greg Kroah-Hartman @ 2024-03-06 6:46 UTC [6%]

3. CVE-2024-26627: scsi: core: Move scsi_host_busy() out of host lock for waking up EH handler - by Greg Kroah-Hartman @ 2024-03-06 6:46 UTC [6%]

4. CVE-2023-52592: libbpf: Fix NULL pointer dereference in bpf_object__collect_prog_relos - by Greg Kroah-Hartman @ 2024-03-06 6:45 UTC [6%]

5. CVE-2023-52591: reiserfs: Avoid touching renamed directory if parent does not change - by Greg Kroah-Hartman @ 2024-03-06 6:45 UTC [6%]

6. CVE-2023-52590: ocfs2: Avoid touching renamed directory if parent does not change - by Greq Kroah-Hartman @ 2024-03-06 6:45 UTC [6%]

VULNERABILITIES

Firefox 116 Patches High-Severity Vulnerabilities

Firefox 116 was released with patches for 14 CVEs, including nine high-severity vulnerabilities, some of which can lead to remote code execution or sandbox escapes.



```
nut Arghire
ugust 2, 2023
```





Search results ordered by [date|relevance] view[summary|nested|Atom feed] download mbox.gz: results only | full threads

Bosch Smart Thermostat Feels the Heat From Firmware Bug The vulnerability in a popular hospitality industry gadget allows attackers to take over the device, pivot into the user's network, or brick the device entirely, rendering HVAC unusable. Nathan Eddy, Contributing Writer ③ 3 Min Read uary 16, 2024 BOSCH Ξ 12:00 pm 11111 72 Until 12:00pm 40%

POTUS executive order aims to keep U.S. ports safe from cyberattacks

Steve Zurier February 21, 2024



port of seattle with downtown skyline early morning

linux-cve-announce.vger.kernel.org archive mirror search help / color / mirror / Atom feed

1. CVE-2024-26626: ipmr: fix kernel panic when forwarding mcast packets - by Greg Kroah-Hartman @ 2024-03-06 6:46 UTC [4%]

2. CVE-2024-26628: drm/amdkfd: Fix lock dependency warning - by Greg Kroah-Hartman @ 2024-03-06 6:46 UTC [6%]

3. CVE-2024-26627: scsi: core: Move scsi_host_busy() out of host lock for waking up EH handler - by Greg Kroah-Hartman @ 2024-03-06 6:46 UTC [6%]

4. CVE-2023-52592: libbpf: Fix NULL pointer dereference in bpf_object__collect_prog_relos - by Greg Kroah-Hartman @ 2024-03-06 6:45 UTC [6%]

5. CVE-2023-52591: reiserfs: Avoid touching renamed directory if parent does not change - by Greg Kroah-Hartman @ 2024-03-06 6:45 UTC [6%]

6. CVE-2023-52590: ocfs2: Avoid touching renamed directory if parent does not change - by Greg Kroah-Hartman @ 2024-03-06 6:45 UTC [6%]





design flaw in the DNSSEC specification.



FORBES > INNOVATION > CYBERSECURITY

SolarWinds Is A Game Changer - You Cannot Sugarcoat Cybersecurity

Stewart Room Contributor 3

I write about Data Protection, Privacy and Cyber Security



law



FORBES > INNOVATION > CYBERSECURITY

SolarWinds Is A Game Changer - You Cannot Sugarcoat Cybersecurity

Stewart Room Contributor ()

law

I write about Data Protection, Privacy and Cyber Security



An analysis of an in exploit

By Ian Beer



An analysis of an in-the-wild iOS Safari WebContent to GPU Process

A graph representation of the sandbox escape NSExpression payload

A PATH TOWARD SECURE AND MEASURABLE SOFTWARE

FEBRUARY 2024



We depend on tools to automatically find bugs





LLVM Home | Documentation » Reference » libFuzzer – a library for coverage-guided fuzz

libFuzzer – a library for coverage-guided fuzz testing.

oss-fuzz	oss-fuzz	• New issue	All iss	ues	▼ Q Type=	Bug label:clusterf	uzz -status:Duplicate,WontFix 🔹 🕏 Sign in	
							1 - 100 of 38361 Next > List Grid Chart	
ID 👻	Туре 👻	Component -	Status 👻	Proj 🔻	Reported -	Owner 👻	Summary + Labels 👻 🚥	
17	Bug		Verified	freetype2			Out-of-memory in freetype2_fuzzer ClusterFuzz Reproducible	
47	Bug		Verified	sqlite3	- <u></u> 0		Crash in sqlite3ExprCodeTemp ClusterFuzz Reproducible	
52	Bug		Verified	freetype2			Crash in t1_builder_add_point ClusterFuzz Reproducible	
62	Bug		Verified	libchewing			Heap-buffer-overflow in _Inner_InternalSpecialSymbol ClusterFuzz Reproducible	
64	Bug		Verified	libchewing			Heap-buffer-overflow in ChewingIsChiAt ClusterFuzz Reproducible	
65	Bug		Verified	libchewing	<u></u>		Crash in GetUint24 ClusterFuzz Reproducible	
67	Bug	3 373 2	Verified	libchewing		17777	Heap-buffer-overflow in ueStrNBytes ClusterFuzz Reproducible	
68	Bug	5 111 5	Verified	libchewing	17775)		Negative-size-param in ChewingKillChar ClusterFuzz Reproducible	
69	Bug		Verified	libchewing			Heap-use-after-free in GetUint16 ClusterFuzz Reproducible	
70	Bug		Verified	libchewing	1	(2012	Floating-point-exception in OpenSymbolChoice ClusterFuzz Reproducible	

syzbot Linux
* Open [946] = Subsyst
Name
<u>ci-qemu-upstream</u>
<u>ci-qemu-upstream-386</u>
<u>ci-qemu2-arm32</u>
<u>ci-qemu2-arm64</u>
ci-qemu2-arm64-compat
ci-qemu2-arm64-mte
ci-qemu2-riscv64
ci-upstream-bpf-kasan-gce



~				
ems	🐞 Fixed [5111]	🐞 Invalid [12230]	Missing Backports [83]	$\overline{}$

Instances	tested	repos	:
Inotaneco	reored	I CPOU	

Last active	Uptime	Corpus	Coverage 🕕	Crashes	Execs
now	3h02m	31360	173077	10	769655
now	2h53m	30998	481789	77	518064
now	3h11m	19480	28089	3	196968
now	2h36m	9853	15986	1	49283
now	2h48m	9701	15525		47246
now	3h02m	37199	<u>50867</u>		174128
now	3h09m	381	36935	51	3317
now	14h32m	17911	139168	13	795393

Automated Bug Finding Techniques



Static Data-flow Analysis Formal Methods Model Checking Static Symbolic Execution Compilers

Automated Bug Finding Techniques



Static Data-flow Analysis Formal Methods Model Checking Static Symbolic Execution Compilers



- Unit Testing **Memory Error Detection**
- Sanitization
- **Dynamic Symbolic Execution Fuzz Testing**

Dynamic

Automatically providing unexpected data to a program

Automatically providing unexpected data to a program

Normal Tests

bash -c "echo this is a test"

message="Hello World"; bash -c echo \$message

Automatically providing unexpected data to a program

Normal Tests

bash -c "echo this is a test"

message="Hello World"; bash -c echo \$message

Fuzz Tests

 $A='() \{ 0 \ll a \ll b \ll c \ll d \ll e \ll f \ll g \ll h \ll i \ll j \ll k \ll l \ll m; \}':$ $A='() \{ x() \{ _; \}; x() \{ _; \} \ll a; \}' bash -c :$ $A='() \{ _; \} > [\{((()))] \{ echo hi; id; \}' bash -c :$

Automatically providing unexpected data to a program

Normal Tests

bash -c "echo this is a test"

message="Hello World"; bash -c echo \$message

Fuzz Tests

 $A='() \{ 0 \ll a \ll b \ll c \ll d \ll e \ll f \ll g \ll h \ll i \ll j \ll k \ll l \ll m; \}':$ $A='() \{ x() \{ _; \}; x() \{ _; \} \ll a; \}' bash -c :$ A='() { _; } >_[\$(\$())] { echo hi; id; }' bash -c :

Shellshock 2014

5



Number of Publications about Fuzzing



5

How does Fuzzing Work?

Generate Input

Execute Input





How does Fuzzing Work?

Generate Input

Execute Input





Fuzzing large systems is difficult







Fuzzing large systems is difficult



syzkaller ([siːzˈkɔːlə]) is an unsupervised coverage-guided kernel fuzzer. Supported OSes: Akaros, FreeBSD, Fuchsia, gVisor, Linux, NetBSD, OpenBSD, Windows.

~600k Lines of Code to fuzz a single system

Generating Inputs to Large Systems is Difficult

Generate Input

Execute Input





Security Researchers spend effort generating better inputs





Reshaping makes large targets conducive to fuzzing with small modifications to the execution and feedback stages











Grammar: Teaching the fuzzer to play Basketball



Reshaping: Making Basketball Easier to play



Reshaping: Making Basketball Easier to play

Thesis:

Input-space reshaping is more effective than grammar-based harnessing approaches for fuzzing complex targets.

Research Questions

Is reshaped fuzzing... **RQ1: effective at finding bugs**?

8

Research Questions

Is reshaped fuzzing... **RQ1:** effective at finding bugs? **RQ2**: competitive with other approaches on coverage-achieved?
Research Questions

- Is reshaped fuzzing...
- **RQ1:** effective at finding bugs?
- **RQ2**: competitve with other approaches on coverage-achieved?
- **RQ3:** applicable to a diverse set of targets?

on coverage-achieved? ?

Research Questions

- Is reshaped fuzzing...
- **RQ1:** effective at finding bugs?
- **RQ2**: competitve with other approaches on coverage-achieved?
- **RQ3:** applicable to a diverse set of targets?
- **RQ4:** beneficial even when grammars exist?

on coverage-achieved? ? st?

Research Questions

- Is reshaped fuzzing...
- **RQ1:** effective at finding bugs?
- **RQ2**: competitive with other approaches on coverage-achieved?
- **RQ3:** applicable to a diverse set of targets?
- **RQ4:** beneficial even when grammars exist?
- **RQ5:** compatible with other SoTA fuzzing techniques?

Outline

- Introduction
 - Motivation and Background

- Thesis Statement

- Source-based Hypervisor Fuzzing
- Binary Hypervisor Fuzzing
- Large-Scale Kernel Fuzzing
- Conclusion
 - Summary of Contributions















- Virtual Devices



- Virtual Devices

3D Assets: CC-BY-4.0 foxplay382





3D Assets: CC-BY-4.0 foxplay382



>Physical Memory



3D Assets: CC-BY-4.0 foxplay382



> Physical Memory



3D Assets: CC-BY-4.0 foxplay382



>Physical Memory

VGA



char* vga_mmio = 0xB8000; vga_mmio[0] = 'H';



3D Assets: CC-BY-4.0 foxplay382



> Physical Memory



char* vga_mmio = 0xB8000; vga_mmio[0] = 'H';

3D Assets: CC-BY-4.0 foxplay382



> Physical Memory

VGA

MMIO

MMIO Write



char* vga_mmio = 0xB8000; vga_mmio[0] = 'H';

3D Assets: CC-BY-4.0 foxplay382



VGA

MMIO

MMIO Write

> Physical Memory



char* vga_mmio = 0xB8000; vga_mmio[0] = 'H';

3D Assets: CC-BY-4.0 foxplay382



VGA

MMIO

MMIO Write

> Physical Memory

DMA Example







DMA Example





























outl 0xcf8 0x80001010 out1 0xcfc 0xe1020000 outl 0xcf8 0x80001014 outl 0xcf8 0x80001004 outw 0xcfc 0x7 outl 0xcf8 0x800010a2 write 0xe102003b 0xff write 0xe1020118 0xfffffff write 0xe1020420 0xfffffff write 0xe1020424 0xfffffff write 0xe102042b 0xff write 0xe1020429 0x5 0x0015c5e5c0 write 0x5c041 0x0402e1 write 0x5c048 0x8a write 0x5c04a 0x31 write 0x5c04b 0xff write 0xe1020403 0xff

outl 0xcf8 0x80001010 outl Oxcfc Oxe1020000 outl 0xcf8 0x80001014 outl 0xcf8 0x80001004 outw Oxcfc Ox7 outl 0xcf8 0x800010a2 write 0xe102003b 0xff write 0xe1020118 0xfffffff write 0xe1020420 0xfffffff write 0xe1020424 0xfffffff write 0xe102042b 0xff write 0xe1020429 0x5 0x0015c5e5c0 write 0x5c041 0x0402e1 write 0x5c048 0x8a write 0x5c04a 0x31 write 0x5c04b 0xff write 0xe1020403 0xff



Port IO



Specifies Addr of outl 0xcf8 0x80001010 outl Oxcfc Oxe1020000 outl 0xcf8 0x80001014 outl 0xcf8 0x80001004 outw Oxcfc Ox7 outl 0xcf8 0x800010a2 write 0xe102003b 0xff write 0xe1020118 0xfffffff write 0xe1020420 0xfffffff write 0xe1020424 0xfffffff write 0xe102042b 0xff write 0xe1020429 0x5 0x0015c5e5c0 write 0x5c041 0x0402e1 write 0x5c048 0x8a write 0x5c04a 0x31 write 0x5c04b 0xff write 0xe1020403 0xff





Port IO

MMIO





Port IO

DMA

MMIO





Port IO

DMA

MMIO




Input Semantics



alxndr@bu.edu, bsd@redhat.com, stefanha@redhat.com, megele@bu.edu

The security of the entire cloud ecosystem crucially de Ine security of the entire cloud ecosystem clucially depends on the isolation guarantees that hypervisors provide $V_{M_{o}}$ and the host overlaw T_{o} allow $V_{M_{o}}$ to pends on the isolation guarantees that hypervisors provide between guest VMs and the host system. To allow VMs to communicate with their antironment hypervisore provide oetween guest vivis and the host system. To allow vivis to communicate with their environment, hypervisors provide a start of virtual davisas including nativark interface cards communicate with their environment, hypervisors provide a slew of virtual-devices including network interface cards a slew of virtual-devices including network interface cards and performance-optimized VIRTIO-based SCSI adapters. and performance-opumized VIK110-based SUST adapters As these devices sit directly on the hypervisor's isolation houndary and account not on tight, attack an controlled input (ac As these devices sit directly on the hypervisor's isolation boundary and accept potentially attacker controlled input (e.g. come a malinion aloud tanont) burge and values abilities in ooundary and accept potentially anacker control input (e.g., from a malicious cloud tenant), bugs and vulnerabilities in the devices, implementations have the potential to reader the trom a mancious ciouu remany, ougs and vumeraumues in the devices' implementations have the potential to render the homoscience of the potential to render the une devices unprementations have the potential to reduce the hypervisor's isolation guarantees moot. Prior works applied hypervisor's isolation guarantees moot. Filor works applied fuzzing to simple virtual-devices, focusing on a narrow subset of the work input on and the state of the sart virtual-device

Alexander Bulekov*†

of the vast input-space and the state-of-the-art virtual-device of the vast input-space and the state-of-the-art virtual-to-vict fuzzer, Nyx, requires precise, manually-written, specifications to exercise complex devices. ^b exercise complex aevices. In this paper we present MORPHUZZ, a generic approach that lavarage incidite about hypervices design combined with In this paper we present intoken 0.4.2., a generic approach that leverages insights about hypervisor design combined with ouror and and forging to find burge in wirthed device imple-

utat teverages insignits about inspervisor design combined with coverage-guided fuzzing to find bugs in virtual device imple-mentations. Crucially, Mappinizza does not valy on overage Coverage-guided nuzzing to thid bugs in virtual device imple-mentations. Crucially MORPHUZZ does not rely on expert unourladea chaoife to apph device Menophilitizzio the first Includious. Crucially MORPHUZZ uoes not iely on experi knowledge specific to each device. MORPHUZZ is the first Anownedge specific to each device. MORPHUZZ is the unst approach that automatically elicits the complex I/O behaviors approacn that automatically encurs the complex we becavity of the real-world virtual devices found in modern clouds. To or the real-world virtual devices found in modern clouds. Io demonstrate this capability, we implemented MoRPHUZZ in OFMIT and block and forged 22 different virtual devices for aemonstrate this capability, we implemented MUKPHUZZ in QEMU and bhyve and fuzzed 33 different virtual devices (a VEWU and onyve and juzzed 33 different virtual devices (a superset of the 16 devices analyzed by prior work). Addition-ouver the Mon puterza is not find to a specific Optisuperset of the 10 devices analyzed by prior work). Addutionally, we show that MORPHUZZ is not tied to a specific CPU auy, we show that MORPHUZZ Is not trea to a specific CPU architecture, by fuzzing 3 additional ARM devices. MOR-httirzz matches or avagade advising a chine of his More for autocounce, by inzering a automation in Artistic devices. Works PHUZZ matches or exceeds coverage obtained by Nyx, for 12/16 wirtual davridge and identified a sumaroat (110) of all PHUZZ Intactics of exceeds coverage obtained by 1975, 101 13/16 virtual devices, and identified a superset (110) of all 15/10 VIIIIai uevices, and identified a superset (110) 01 au crashes reported by Nyx (44). We reported all newly discov ered bugs to the respective developers. Notably, MORPHUZZ erea bugs to the respective aevelopers. Notably, MIORPHUZZ achieves this without initial seed-inputs, or expert guidance. hile the cloud unveils unique opportunities to IT businesses, Presents a host of fundamental security issues. From a

VMMs) multiplex the hardware resources of a physical ma-V MINIS IIIUIIIpiex uie naraware resources of a physical ma chine (the host), between multiple Virtual Machines (VMs or macte) Cloud roady hypomytopic are complex microsoft and Cline (the nost), between multiple virtual indentities (virts of success). Cloud-ready hypervisors are complex pieces of softguests). Cloud-ready hypervisors are complex pieces of solu-ware, tasked with isolating the software running inside a VM (i = 0 migot) from the other migots and the hypervisor itself Ware, tasked with isolating the solitware running inside a via (i.e., a guest), from the other guests, and the hypervisor itself. (i.e., a guesu, from the other guesus, and the river visor fisch Beyond the cloud, hypervisors are commonly used to sandbox and for declifon the applications (e.g., for malware research), and for desktop use, applications (c.g., 101 manwate research), and 10r desktop use, to run applications not supported by the host OS. Regard-less the annification hypervisions are trusted with moviding of to tun applications not supported by the nost OS. Regard less the application, hypervisors are trusted with providing a norm of instant hot visor visors and the boot Oc less une application, hypervisors are trusted whit providing a layer of isolation between virtual machines and the host of solution to the second state to access to a layer or isolation between virtual machines and the nosi of crucially, to provide their functionality to guests, hypervisors include a class of implementations for virtual devices and the Cuciauy, to provide their functionarity to guests, hypervisors include a slew of implementations for virtual devices, and the output the state devices of the miniplement Include a slew of implementations for virtual devices, and uncode for these devices commonly executes at the privilege code for these devices commonly executes at the privilege level of the hypervisor itself. Virtual devices play a critical main anomning that the guest is isolated but due to the com level of the hypervisor lised. Virtual devices play a cruical role in ensuring that the guest is isolated, but due to the com-navity of these devices it can be difficult to cofely implement I to te ut cusuring utat me guest is isolated, but due to the com-plexity of these devices, it can be difficult to safely implement thair functionality in a fortune of the fortune to be com-Picany of unse devices, it can be duffed to safely implement their functionality in software. Unfortunately exploits compromising this layer of isolation (and specifically the virtual daviage) and a topolitic in 2015 VIENTON (14) was promising unis layer of isolation (and specifically the virtual devices) are a tangible reality. In 2015, VENOM [14] was highly multicized as a VM. Feerane virtuarehility which allows aevices) are a tangible reality. In 2013, VEINOW 1141 was highly publicized as a VM-Escape vulnerability, which allows nigniy puolicized as a vivi-tiscape vulnerability, which allows an attacker running within an untrusted guest to compromise the underlying hypervisor and events and an anacker running wrunn an unnusien guest to compromise the underlying hypervisor and execute code outside the secu-rity confines of the V/M VENTOM is containly not a union the underlying nypervisor and execute code outside the security confines of the VM. VENOM is certainly not a unique inty commes of the vivie v Elvolvi is certainly not a unique example, and security researchers have identified many vul-nershilition loading to motortical var Economic Destroit to the example, and security researchers have identified many vul-nerabilities leading to potential VM-Escape. Ranked by vul- $M_{A} = 0$ and M_{A nerabilities leading to potential VIVI-Escape. Kalikeu vy ule size of bug bounties, VM-escapes are considered among the most oritical classes of vulnerabilities along with iOC An size of oug boundes, vivi-escapes are considered among un most critical classes of vulnerabilities, along with iOS, Anmost cruicat classes of vumeraoniues, along with 103, Au-droid, and browser bugs [58]. Though VM-escape attacks arola, and browser bugs [Jo]. Hubugu Vivi-escape auacka can take advantage of weaknesses in other hypervisor com-mante euch ac chodow page toblac our work fromces on Can take auvantage of weaknesses in other hypervisor com-ponents, such as shadow page tables, our work focuses on wirthal davieae which are reconciliant for the work focuses on Ponents, such as snadow page lables, our work locuses on virtual-devices which are responsible for the vast majority of reported VM-escape vulnerabilities [37]. Software fuzz testing has proven to be nique, capable of exposing vuln

Manuel Egele*

of software 13 12 15 virtual dev

MORPHUZZ: Bending (Input) Space to Fuzz Virtual Devices

MORPHUZZ Bending (Input) Space to Fuzz Virtual Devices

USENIX Security 2022 Bending (Input) Space to Fuzz Virtual Devices

Reshaping the Input Space



Reshaping the Input Space



Reshaping the Input Space



Achieve a precise view of the regions that are actively engaged in IO















Star 000000 000A00

000000

FEBC0

•t	End	Name
000	0009FFFF	RAM
000	000BFFFF	VGA
000	FEBBFFFF	RAM
000	FEBDFFFF	NETWORK
000	0000000	



•t	End	Name
000	0009FFFF	RAM
000	000BFFFF	VGA
000	FEBBFFFF	RAM
000	FEBDFFFF	NETWORK
000	0000000	



Star 000000 000A0 000000 **FEBC0** 00000

`t	End	Name
000	0009FFFF	RAM
000	000BFFFF	VGA
000	FEBBFFFF	RAM
000	FEBDFFFF	NETWORK
000	0000000	

- Hypervisors must track PIO/MMIO regions to trap and emulate accesses.
- The memory layout table provides a perfect view of active IO regions.

Reshaping DMA





Reshaping DMA

1. Populate DMA Memory























- Two Hypervisors (QEMU and Bhyve)
- 33 Virtual Devices
- Coverage
 - Fuzzed for 24 Hours
 - 81% Overall
 - Equal/Higher coverage for 13/16 Devices
- DMA Evaluation
 - Improves Coverage for 24/33 Devices

- Two Hypervisors (QEMU and Bhyve)
- Coverage
 - Fuzzed for 24 Hours
 - 81% Overall
 - Equal/Higher coverage for 13/16 Devices
- DMA Evaluation
 - Improves Coverage for 24/33 Devices

Audio
ac97
cs4231a
es1370
intel-hda
sb16
IBM PC
fdc
parallel
serial
Block
ide/core
ahci
sdhci
virtio-blk
virtio-sesi
megasas
sd
sesi-disk
Network
eepro100
e1000
e1000e core
ne2000
ncnet
rt18130
vmxnet3
virtio-net
Graphics
virtio-gou
cirrus voa
hcd-ehci
hed-xhei
arm gic
smc91c111
xgmac
hhvve
nci vhci
virtio block

Two Hypervisors (QEMU and Bhyve)

Device Source File Block ide/core ahci sdhci sdhci virtio-blk virtio-scsi

• DMA Evaluation

Average Average (All devices)

• Improves Coverage for 24/33 Devices

VDF [‡]	Hyper-Cube [*]	Nyx‡	QMorphuz		
25-65 Days	24 Hours	24 Hours	24 Hours	Bug	
Cov.	Cov.	Cov.	Cov.		
27.5%	74.87%	74.69%	78.63%	~	
			80.86%	~	
90.5%	81.15%	88.93%	84.8%	~	
			68.51%	V	
			66.78%	V	
61.67%	76.35%	78.16%	85.76%		
			81.08%		

• Two Hypervisors (QEMU and Bhyve)

Device Source Fi Networ eepro10 e1000e_cor ne200

Average67.51%69.42%81.08%

	No-DMA	Scratch-Buffer	QMORPHUZZ					
	24 Hours	24 Hours	24 Hours					
ile	Cov.	Cov.	Cov.					
rk								
00	87.13%	87.13% (0.00)	89.26% (+2.13)					
00	65.77%	66.14% (+0.37)	89.23% (+23.09)					
ore	75.24%	75.84% (+0.60)	90.54% (+14.70)					
00	82.95%	83.47% (+0.52)	98.71% (+15.24)					
let	71.38%	72.72% (+1.34)	96.35% (+23.63)					

Bugs

66 New (110 Total)

- **29 Assertion Failures**
- 8 Stack Overflow
- 8 Null-Ptr Deref
- 7 UAF
- 7 Buffer Overflow
- 7 Other

```
Assertion-failure in audio_bug
Assertion-failure in mch_update_pciexbar
Assertion-failure in vmxnet3_validate_interrupt_idx
Assertion-failure in vmxnet3_validate_queues
Assertion-failure in address_space_stw_le_cached through virtio-net
Assertion-failure in address_space_stw_le_cached through virtio-blk
Assertion-failure in address_space_cache_invalidate through virtio-gpu
Assertion-failure in address_space_unmap through ahci_map_clb_address
Assertion-failure in address_space_unmap through virtio-blk
Assertion-failure in virtio blk reset
Assertion-failure in bdrv_aio_cancel
Assertion-failure in bmdma_active_if
Assertion-failure in e1000e_write_lgcy_rx_descr
Assertion-failure in e1000e_write_rx_descr
Assertion-failure in e1000e_write_to_rx_buffers
Assertion-failure in e1000e_intrmgr_on_throttling_timer
Assertion-failure in e1000e_intmgr_collect_delayed_causes
Assertion-failure in eth_get_gso_type through e1000e
Assertion-failure in iov_from_buf_full through e1000e
Assertion-failure in net_tx_pkt_add_raw_fragment through vmxnet3
Assertion-failure in net_tx_pkt_reset through vmxnet3
Assertion-failure in pci_bus_get_irg_level
Assertion-failure in scsi_dma_complete, with megasas
Assertion-failure in usb_detach
Assertion-failure in ati_reg_read_offs and ati_reg_write_offs
Assertion in modify_bar_registration
Assertion in unregister_mem
Assertion in pci_vtnet_proctx
Assertion in pci_vtnet_cfgwrite
Assertion-failure in gic_clear_pending_sgi
Assertion-failure in bcm2835_thermal_read
Assertion-failure in dwc2_hsotg_write
Stack-overflow in ahci_cond_start_engines
Stack-overflow in _eth_get_rss_ex_dst_addr
Stack-overflow in rt1NUMBER_transmit_one
Stack-overflow in pcnet_poll_timer
Stack-overflow in e1000_receive_iov
Stack-overflow in flatview_do_translate through e1000
Stack-overflow in intel_hda_corb_run
Stack-overflow in xhci_pci_intr_raise
Null-Ptr Deref in virtio_write_config
Null-Ptr Deref in address_space_to_flatview through ide
Null-Ptr Deref in blk bs
Null-Ptr Deref in megasas_command_complete
Null-Ptr Deref in megasas_handle_frame
Null-Ptr Deref in tcg handle interrupt
Null-Ptr Deref in usb_bus_from_device
Null-Ptr Deref in vq_getchain
Null-Ptr Deref in smc91c111_writeb
Heap-use-after-free in e1000e_write_packet_to_guest
Heap use-after-free in e1000e_write_to_rx_buffers
Heap-use-after-free in ehci_flush_gh
Heap-use-after-free in usb_packet_copy
Heap-use-after-free in usb_packet_unmap
Heap-use-after-free in virtio_gpu_ctrl_response
Heap-use-after-free through double-fetch in ehci
Heap-use-after-free in gic_dist_writeb
Buffer-underflow in xhci_runtime_write
Global-buffer-overflow in mode_sense_page
Heap-buffer-overflow in sdhci_write_dataport
Heap-buffer-overflow in sdhci_data_transfer
Heap-buffer-overflow in sd_erase
Heap-buffer-overflow in msix_table_mmio_write
Heap-buffer-overflow in pcnet_receive
Memcpy-param-overlap in flatview_write_continue
Memcpy param-overlap in ip_stripoptions
Memcpy param-overlap through e1000e_write_to_rx_buffers
Memory Exhaustion in vmxnet3_activate_device
Memory Exhaustion in hpet_timer
Infinite Loop in sdhci_data_transfer
Floating-point exception in ide_set_sector
```

Morphuzz is Upstream in QEMU

- Continuously fuzzed on OSS-Fuzz
- 200+ Issues Reported
- Reproducers are simple to use
- Bugs are caught before release

git clone git.qemu.org/qemu.git



Author: Thomas Huth <thuth@redhat.com> Date: Thu Jul 15 21:32:19 2021 +0200

hw/net/net_tx_pkt: Fix crash detected by fuzzer

QEMU currently crashes when it's started like this:

cat << EOF | ./gemu-system-i386 -device vmxnet3 -nodefaults -gtest stdio</pre> outl 0xcf8 0x80001014 outl 0xcfc 0xe0001000 outl 0xcf8 0x80001018 outl 0xcf8 0x80001004 outw 0xcfc 0x7 outl 0xcf8 0x80001083 write 0x0 0x1 0xe1 write 0x1 0x1 0xfe write 0x2 0x1 0xbe write 0x3 0x1 0xba writeg 0xe0001020 0xefefff5ecafe0000 writeg 0xe0001020 0xffff5e5ccafe0002 EOF It hits this assertion: gemu-system-i386: ../gemu/hw/net/net_tx_pkt.c:453: net_tx_pkt_reset: Assertion `pkt->raw' failed. This happens because $net_tx_pkt_init()$ is called with $max_frags == 0$ and

thus the allocation

commit 283f0a05e24a5e5fab78305f783f06215390d620

Bend the virtual-device input space to make it conducive to fuzzing

Use time-tested **off-the-shelf fuzzers**

Fuzz *any* **device**, across all PIO, MMIO, and DMA interfaces. No per-device analysis or descriptions, needed

MORPHUZZ



HYPERPILL: Fuzzing for Hypervisor-bugs by Leveraging the Hardware qiang.liu@epfl.ch megele@bu.edu *EPFL [†]Boston University [‡]Zhejiang University [§]Amazon Mathias Payer* mathias.payer@nebelwelt.net Abstract Ine security guarantees of crown computing uspend on the iso-lation guarantees of the underlying hypervisors. Prior works have presented effective methods for automatically identify. lation guarantees of the undertying hypervisors. Frior works have presented effective methods for automatically identify in a submanchilition in hypothesis to automatically identify. liave presented effective methods for automatically lucituity ing vulnerabilities in hypervisors. However, these approaches are limited in econe Econ instance their implementation is Introduction Ing vumeraumues in mypervisors. However, mese approaches are limited in scope. For instance, their implementation is temicolly, hypervisor enocide and limited by requirements are unnieu in scope. ror instance, uier implementation is typically hypervisor-specific and limited by requirements for detailed anomare appages to compare and accume

typically uppervisor-specific and influence by requirements for detailed grammars, access to source-code, and assump-tione about hymeryticor heboyilore. In prosting complex along Ior aetailea grammars, access to source-coue, and assumptions about hypervisor behaviors. In practice, complex closed uous about uyper visor venaviors. In practice, complex closed source and recent open-source hypervisors are often not suitable for off-the-shelf fuzzing techniques. HYPERPILL introduces a generic approach for fuzzing ar-IT Y PEKFILL IIII OUUCES à generic approach ion iuzzang at bitrary hypervisors. HYPERPILL leverages the insight that outpour involvementations are diverged all hypervisor oluary hypervisors. If YPERFILL reverages we mistry utan although hypervisor implementations are diverse, all hypervisor come rate on the identical underlying hordering viewed viewed in the although hypervisor implementations are unverse, an inverse sors rely on the identical underlying hardware-virtualization interference to monocol virtual monthings to take advantage of Sors reiy on the identical underlying hardware-virtualization interface to manage virtual-machines. To take advantage of the hordware virtualization interface. If votto Drive makane the hardware-virtualization interface, HYPERPILL makes a une nauware-viruanzaron mileriace, mirekritte makes a snapshot of the hypervisor, inspects the snapshotted hardware of the to animorota the hypervisor's input ongoes and lavar Suapsilot of the hypervisor, inspects the shapsholicul naturate state to enumerate the hypervisor's input-spaces, and lever-anae foodbook-manchot-firzaing within an amulated State to enumerate the hypervisor's input-spaces, and lever ages feedback-guided snapshot-fuzzing within an emulated anvironment to identify the home bilities in orbitrory by by ages teedback-guided snapsnot-tuzzing within an chimaton environment to identify vulnerabilities in arbitrary hypervienvironment to tuentity vumeraumites in automaty hypervis sors. In our evaluation, we found that beyond being the first Sors. In our evaluation, we toution that beyond being the instance of identifying vulnerabilities in arbitrary by power and analog all major attack ourfaces (i.e. aromary mypervisors across an major anack-surfaces (i.e., PIO/MMIO/Hypercalls/DMA), HypERPILL also outperforms FIG/IVIIVIO/TYPEICalls/DIVIA), TIYEKFILL also outperions state-of-the-art approaches that rely on access to source-code, due to the original arity of feedback arounded by Union Division State-or-ute-an approaches that rely on access to source-code due to the granularity of feedback provided by HypeRPILL's uue to the granularity of needback provided by $\pi_Y PEKPILL's$ emulation-based approach. In terms of coverage, HYPERPILL'semulation-based approach. In terms of coverage, it i PERFILL Outperformed past fuzzers for 10/12 QEMU devices, without the API hooking or source-code instrumentation techniques equired by prior works. HypeRPILL identified 26 new bugs quired by prior works. HYPERFILL Identified 20 new bugs recent versions of QEMU, Hyper-V, and macOS Virtual. I work was completed prior to author ioining Amar

macOS Virtuali

Hypervisors provide the security foundations necessary for Hypervisors provide the security ioundations necessary ion the cloud. They enable efficient use of hardware resources, by colociting trouble of communicipal tononto on the commu ute croud. They endore entrient use of nardware resources, by colocating workloads from multiple tenants on the same how motor monthings included in individual visual visual monthings. Us constants workroads it out introductions on the same bare-metal machines, isolated in individual virtual-machines $(\chi_{M_0}) = \Lambda_0$ such hypervisions of the solid succession in χ_{M_0} Vare-metal machines, isolated in murricular virtual-machines (VMs). As such, hypervisors ensure that code running in VMs (VIVIS). As such, hypervisors ensure mai coue rummis in Vivis cannot violate the virtualization boundary (e.g., by performing cannot violate the virtualization boundary (e.g., by perioriting a VM escape attack) and compromise the workloads of the other tenants on the by perioritical f Unfortunately, VM escape attacks are a tangible reality. Hundreds of bugs have been identified in the complex hyper-Humareus or ougs nave been nuemineu in uie complex uyper visor code. Due to the severity of these bugs, hypervisor com-Visor coue. Due to the severity of these ougs, hypervisor com-promises are awarded large bug bounties, similar to other high-value torrate ouch as web browcore and mobile doutions [52] Promises are awarded large bug boundes, similar to other high-value targets such as web browsers and mobile devices [53]. value targets such as web browsers and mobile devices [33]. In parallel, fuzzing has emerged as one of the most powerful toohniquee for automatically incovering valueschilities in a ¹¹¹ paranet, 11221118 has enterged as one of the most powerful techniques for automatically uncovering vulnerabilities in a large range of coffigare [A 10 14 12 20 21 27 22 20 45 40 52] large range of software [4,10,14,18,20,21,27,28,39,45,49,52]. As such, a significant amount of academic research has for cused on leveraging fuzzing to automatically identify bugs in homosofies and another that the proposition for proposition for the theory of the proposition for the pro Cused on reveraging fuzzing to automatically identity ougs in hypervisor code, so that they can be promptly fixed, prevent-ing malicity avaluation 16 13 26 20 32 37 301 ing malicious exploitation [6, 13, 26, 29, 32, 37, 38]State-of-the-art approaches [6, 26] are capable of automati-State-of-the-art approaches [0, 40] are capable of automati-cally finding complex bugs across most major attack-surfaces $i_{a} = Di \cap AAAAI \cap AAAA \cup H_{OVAVAVAT}$ there are a subscaled with the surfaces cauy maing complex bugs across most major attack-surfaces (i.e., PIO/MMIO/DMA). However, these approaches rely on (i.e., FIU/MIVIU/DIVIA). FIUWEVEL, LIESE approaches Iciy ou access and manual modifications to hypervisor source-code access and manual mounications to hypervisor source-code to effectively fuzz virtual-devices. Even with access to source-ode posting arready mothed to post t to enecurery fuzz virtual-devices. Even with access to solution code, porting current methods to new targets is a non-trivial managed that manifest and analy a fear by an average Coue, porting current methods to new targets is a non-trivial process that requires considerable manual effort by an expert. Process that requires considerable manual entity by an experi-Furthermore, most fuzzers do not handle the hypercall attack-Furthermore, most nuzzers up not nantice the hypercall analysis surface as hypercalls are often implemented in a separate surface as hypercaus are oncer impremented in a separate component from the core device-emulation (e.g. in the Oc kernel), for performance reasons. Thus even the open-source targets such as OFMI1 and

HYPERPILL Fuzzing for Hypervisor-bugs by Leveraging the Hardware Virtualization Interface

USENIX Security 2024



<pre>129 (vmw_shmem_read(_d, shpa + offsetof(struct Vmxnet3_DriverShared, field</pre>	<pre>129 NetClientState *nc = qemu_get_queue(n->nic); 138 static const MACAddr zero = { .a = { 0, 0, 0, 0, 0, 0 } };</pre>
130	131
131 #define VMXNET_FLAG_IS_SET(field, flag) (((field) & (flag)) == (flag))	132 int ret = 0;
132	<pre>133 memset(&netcfg, 0 , sizeof(struct virtio_net_config));</pre>
133 struct VMXNEI3Class {	134 virtio_stw_p(vdev, &netcfg.status, n->status);
134 PCIDeviceClass parent_class;	<pre>135 virtio_stw_p(vdev, &netcfg.max_virtqueue_pairs, n->max_queue_pairs);</pre>
135 DeviceRealize parent_dc_realize;	<pre>136 virtio_stw_p(vdev, &netcfg.mtu, n->net_conf.mtu);</pre>
136 };	<pre>137 memcpy(netcfg.mac, n->mac, ETH_ALEN);</pre>
137 typedef struct VMXNET3Class VMXNET3Class;	<pre>138 virtio_stl_p(vdev, &netcfg.speed, n->net_conf.speed);</pre>
138	<pre>139 netcfg.duplex = n->net_conf.duplex;</pre>
139 DECLARE_CLASS_CHECKERS(VMXNET3Class, VMXNET3_DEVICE,	<pre>140 netcfg.rss_max_key_size = VIRTIO_NET_RSS_MAX_KEY_SIZE;</pre>
140 TYPE_VMXNET3)	141 virtio_stw_p(vdev, &netcfg.rss_max_indirection_table_length,
141	<pre>142 virtio_host_has_feature(vdev, VIRTIO_NET_F_RSS) ?</pre>
<pre>142 static inline void vmxnet3_ring_init(PCIDevice *d,</pre>	<pre>143 VIRTIO_NET_RSS_MAX_TABLE_LEN : 1);</pre>
143 Vmxnet3Ring *ring,	<pre>144 virtio_stl_p(vdev, &netcfg.supported_hash_types,</pre>
144 hwaddr pa,	<pre>145 VIRTIO_NET_RSS_SUPPORTED_HASHES);</pre>
145 uint32_t size,	<pre>146 memcpy(config, &netcfg, n->config_size);</pre>
146 uint32_t cell_size.	147
147 bool zero region)	148 /*
148 {	149 * Is this VDPA? No peer means not VDPA: there's no way to
149 ring->na = na:	150 * disconnect/reconnect a VDPA neer.
150 ring-size = size:	151 */
151 ring->cell size:	152 if (nc->neer && nc->neer->info->type == NET CLIENT DRIVER VHOST VDPA)
152 ring->gen = VMXNET3 INIT GEN:	
153 ring-spect = \mathbf{P} .	153 ret = vhost net get config(get vhost net(nc-sneer) (uint8 t *)&n
154	tofa
155 if (zero region) {	15/
$\frac{155}{16} \qquad \qquad \text{vmu chrom cot}(d \text{ no } \theta \text{ citro t coll citro})$	154 $11-201119_{12}(129),$
150 Villw_Slillell_Set(u, pa, 0, Size " ceii_Size);	155 II (IEC :I) { 154 /*
10/ 5	150 /* 157 * Some NTC/kernel combinations present 0 on the res address
150 5	157 Solie Wit/Kernet Combinations present 6 as the mac address.
157 160 #define VMVNET2 DING DUMD(means ning name nidy n)	159 * that and address try to present with the
ioo #deline vixincis_kind_bonr(macro, ling_name, liux, l)	150 that the regar address, thy to proteed with the
1/1 manne("WettWdy been W" DDTy// " size Wy sell size Wy gen Wd newt Wy"	157 add and an analysis and a separately allowbare just not
Indered with the set of FRIXO4 Size we ceri_size we get we next we ,	
1/2 (ning page) (nidy)	
loz (ring_name), (riux),	
142 (n) + nn (n) + ni (n) + nn (n) +	165 etcry: e 2eor(2ero)) 8) {
103 (r)->pa, (r)->st	104 van it audress detected. Ignoring
104 145 static inline word a pat	
	105 mac, cin_ALEN);
	107 memory (char in-scontrig_size);
100 Fing->next	
109 ring->gen =	
	170 }
	1/2 static void virtio_net_set_config(virtiouevice ^vdev, const uint8_t ^confi
1/3 static inline void vmxnet3_rit (SRing *ring)	g)
1/5 1f (ring->next == 0) {	1/4 VirtlUNet *n = VIRIIU_NEI(vdev);
<pre>1/6 ring->next = ring->size - 1;</pre>	<pre>1/b struct virtio_net_config netcfg = {};</pre>
1// ring->gen ^= 1;	<pre>1/6 NetClientState *nc = qemu_get_queue(n->nic);</pre>
1/8	177
179 }	<pre>178 memcpy(&netcfg, config, n->config_size);</pre>
180	179
181 static inline hwaddr vmxnet3_ring_curr_cell_pa(Vmxnet3Ring *ring)	<pre>180 if (!virtio_vdev_has_feature(vdev, VIRTIO_NET_F_CTRL_MAC_ADDR) &&</pre>
182 {	181 !virtio_vdev_has_feature(vdev, VIRTIO_F_VERSION_1) &&
183 return ring->pa + ring->next * ring->cell_size;	<pre>182 memcmp(netcfg.mac, n->mac, ETH_ALEN)) {</pre>
184 }	<pre>183 memcpy(n->mac, netcfg.mac, ETH_ALEN);</pre>
185	<pre>184 qemu_format_nic_info_str(qemu_get_queue(n->nic), n->mac);</pre>
186 static inline void vmxnet3_ring_read_curr_cell(PCIDevice *d, Vmxnet3Ring *	185 }
ri <mark>n</mark> g,	186
187 void *buff)	187 /*
188 {	188 * Is this VDPA? No peer means not VDPA: there's no way to
<pre>189 vmw_shmem_read(d, vmxnet3_ring_curr_cell_pa(ring), buff, ring->cell@@@</pre>	189 * disconnect/reconnect a VDPA peer.







Reshaping

```
static bool tulip_rx_stopped(TULIPState *s)
       return ((s->csr[<mark>5</mark>] >> CSR5_RS_SHIFT) & CSR5_RS_MASK) == CSR5_RS_STOPP
          D:
 3 static void tulip_dump_tx_descriptor(TULIPState *s,
          struct tulip_descriptor *desc)
      trace_tulip_descriptor("TX ", s->current_tx_desc,
                  desc->status, desc->control >> 22,
                   desc->control & 0x7ff, (desc->control >> 11) & 0x7ff,
                  desc->buf_addr1, desc->buf_addr2);
140
42 static void tulip_dump_rx_descriptor(TULIPState *s,
          struct tulip_descriptor *desc)
      trace_tulip_descriptor("RX ", s->current_rx_desc,
                  desc->status. desc->control >> 22.
                  desc->control & 0x7ff, (desc->control >> 11) & 0x7ff,
                   desc->buf_addr1, desc->buf_addr2);
   static void tulip_next_rx_descriptor(TULIPState *s,
      struct tulip_descriptor *de
      if (desc->control & 
           s->current_rx_
      } else if (desc->d
                                   ->buf
          s->current_r>
        else {
          s->current ry
                                                     descriptor) +
                                                  ) & CSR0_DSL_MASK) << 2);
                  (((s->
        ->current_rx_desc &= ~<mark>3</mark>
 5 static_void tulip_copy_rx_bytes(TULIPState *s, struct tulip_descriptor *
      sc)
       int len1 = (desc->control >> RDES1_BUF1_SIZE_SHIFT) & RDES1_BUF1_SIZ
          MASK:
68
      int len2 = (desc->control >> RDES1_BUF2_SIZE_SHIFT) & RDES1_BUF2_SIZ
          MASK;
169
      int len;
      if (s->rx_frame_len && len1) {
          if (s->rx_frame_len > len1)
len = len1;
         } else {
              len = s->rx_frame_len;
          pci_dma_write(&s->dev, desc->buf_addr1, s->rx_frame +
              (s->rx_frame_size - s->rx_frame_len), len);
          s->rx_frame_len -= len:
      if (s->rx_frame_len && len2) {
          if (s->rx_frame_len > len2) {
              len = len2;
         } else {
              len = s->rx_frame_len;
```

















Port IO













00007fe0: 8b35 5d44 dcd9 c3a6 e007 8ca5 01fc d5df .5]D. 00007ff0: 68eb 956d 69f2 3dff ccba c94d ba95 931c h..mi 00008000: 6f99 64a3 8888 13a3 b84f dcb6 cc9e 5071 o.d.. 00008010: cdba c94d 6c9e 7aaa cfba c94d b85b 938b ...Ml 00008020: f93c 81a9 d0ba c94d f9cb 8d8d 5a9b e8b0 .<... 00008030: cac2 d3b8 dcb7 5e4c e15f 8598 74fe 7497 00008040: fc39 d79c 5b76 053d d2ba c94d 8883 b0e8 .9..[00008050: 2575 a29f 2551 64ff e0d9 fa85 b7bf 4d8c %u..% 00008060: 7e4f 3c2a 7413 4b0b c2b1 14b7 2e43 d893 ~0<*t 00008070: 6d16 bc16 9879 87af f2c1 2a2b a19f a90f m.... 00008080: 265a e8ab 6828 5bb3 555f 2673 3a0d 85b3 &Z..h 00008090: 3140 4702 d79a 4813 5c52 6a8d c95a 414e 1@G.. 000080a0: 10e3 77ea 62cc e665 edda c8b7 6044 ce3e ..w.b 000080b0: a883 3d41 6083 4d73 42dc 4e7e fe8d 698e ..=A` 000080c0: 30bb d143 3bc2 cee6 a444 6ecb e4a6 d4dc 0..C: 000080d0: 7d3b f166 9580 42cb 0ddb 60ea 7e5d 5d1e };.f. 000080e0: c421 5845 be42 31e0 6352 6a8d df4b 7d81 .!XE. 000080f0: e0d7 9f9a 8c34 feea 7bc9 1403 6fea da25 00008100: 4237 5fe9 67c9 421c 315b 851a 4ec5 a38c B7_.g 00008110: b21b 1360 f345 7fba 17bc 55e2 f670 205e ...`. 00008120: b9c7 5bdf 648b d75a 051e 5969 527d 3165 ..[.d 00008130: 23c6 d3b3 fa88 668a 0ddf 5683 309c dadf #.... 00008140: 1421 d944 d1e5 d412 4b04 750f 4aba 9a7c .!.D. 00008150: 43cd ffa5 2a4b 1521 9ecf a439 375d 1154 C...* 00008160: 4704 581e 04d1 5a69 335d 8757 df96 45b7 G.X.. 00008170: 74a7 d4e9 4a68 1266 82e6 5d1e fbdf daa8 t...J 00008180: 68ff 6a03 0cf1 39ad 770a a100 eec1 2935 h.j. 00008190: 238b 8fa6 b2e4 b68a e158 1fc4 ba9a 58ec #.... 000081a0: 29ee 64bc 5d47 79f0 87c5 cd59 3f2d c613).d.] 000081b0: 5e88 3678 6872 cb13 8241 8060 e0ad e8c3 ^.6xh 000081c0: d1f6 5710 37fc 9aea 0f31 8c35 f6b6 f928 ...W.7 000081d0: 892e 893c ea0e 9eca d165 ce6d ef5e 11a5 ...<. 000081e0: 64fd 1810 460c 670b 693a f4e9 e9fc 483e d...F 000081f0: ca6c 73e3 c42e c99d 96ed 1881 bb54 7cf0 .ls.. 00008200: 8be2 c613 861f 85fc 72fa 811b 092d 4dea 00008210: 567b 80e7 095a 9248 3cc8 7a5c fc25 7c53 V{.... 00008220: 95de 5151 3c27 ca2b ba07 bc5a c210 c9c9 ...Q< 00008230: 0519 bc3e 1cb5 b2c4 4be3 cf89 4727 6934 ...>. 00008240: 88b9 e2c5 b2ac 8773 ece9 432b 99c2 fb28 00008250: 82d4 b218 cd6a 98bb 7a4a 1595 da85 c370j..zJ....p

	00007fe0:	8b35	5d44	dcd9	c3a6	e007	8ca5	01fc	d5df	.5]D
	00007ff0:	68eb	956d	69f2	3dff	ccba	c94d	ba95	931c	hmi.=M
	0008000:	6f99	64a3	8888	13a3	b84f	dcb6	cc9e	5071	o.d0Pq
= M	00008010:	cdba	c94d	6c9e	7aaa	cfba	c94d	b85b	938b	Ml.zM.[
0 Pa	00008020:	f93c	81a9	d0ba	c94d	f9cb	8d8d	5a9b	e8b0	. <mz< td=""></mz<>
7 M [00008030:	cac2	d3b8	dcb7	5e4c	e15f	8598	74fe	7497	^Lt.t.
M 7	00008040:	fc39	d79c	5b76	053d	d2ba	c94d	8883	b0e8	.9[v.=M
1t.t.	00008050:	2575	a29f	2551	64ff	e0d9	fa85	b7bf	4d8c	%u%QdM.
v.=M	00008060:	7e4f	3c2a	7413	4b0b	c2b1	14b7	2e43	d893	~O<*t.KC
0dM.	00008070:	6d16	bc16	9879	87af	f2c1	2a2b	a19f	a90f	my*+
.KC	00008080:	265a	e8ab	6828	5bb3	555f	2673	3a0d	85b3	&Zh([.U_&s:
V*+	00008090:	3140	4702	d79a	4813	5c52	6a8d	c95a	414e	1@GH.\RjZAN
([.U &s:	000080a0:	10e3	77ea	62cc	e665	edda	c8b7	6044	ce3e	w.be`D.>
.H.\RiZAN	000080b0:	a883	3d41	6083	4d73	42dc	4e7e	fe8d	698e	=A`.MsB.N~i.
e`D.>	000080c0:	30bb	d143	3bc2	cee6	a444	6ecb	e4a6	d4dc	0C;Dn
.MsB.N~i.	000080d0:	7d3b	f166	9580	42cb	0ddb	60ea	7e5d	5d1e	<pre>};.fB`.~]].</pre>
Dn	000080e0:	c421	5845	be42	31e0	6352	6a8d	df4b	7d81	.!XE.B1.cRjK}.
.B`.~]].	000080f0:	e0d7	9f9a	8c34	feea	7bc9	1403	6fea	da25	4{%
B1.cRjK}.	00008100:	4237	5fe9	67c9	421c	315b	851a	4ec5	a38c	B7g.B.1[N
4{%	00008110:	b21b	1360	f345	7fba	17bc	55e2	f670	205e	`.EUp ^
.B.1[N	00008120:	b9c7	5bdf	648b	d75a	051e	5969	527d	3165	[.dZYiR}1e
EUp ^	00008130:	23c6	d3b3	fa88	668a	Øddf	5683	309c	dadf	#fV.0
ZYiR}1e	00008140:	1421	d944	d1e5	d412	4b04	750f	4aba	9a7c	.!.DK.u.J
.fV.0	00008150:	43cd	ffa5	2a4b	1521	9ecf	a439	375d	1154	C*K.!97].T
K.u.J	00008160:	4704	581e	04d1	5a69	335d	8757	df96	45b7	G.XZi3].WE.
K.!97].T	00008170:	74a7	d4e9	4a68	1266	82e6	5d1e	fbdf	daa8	tJh.f]
.Zi3].WE.	00008180:	68ff	6a03	0cf1	39ad	770a	a100	eec1	2935	h.j9.w)5
h.f]	00008190:	238b	8fa6	b2e4	b68a	e158	1fc4	ba9a	58ec	#XX.
.9.w)5	000081a0:	29ee	64bc	5d47	79f0	87c5	cd59	3f2d	c613).d.]GyY?
XX.	000081b0:	5e88	3678	6872	cb13	8241	8060	e0ad	e8c3	^.6xhrA.`
GyY?	000081c0:	d1f6	5710	37fc	9aea	0f31	8c35	f6b6	f928	W.71.5(
rA.`	000081d0:	892e	893c	ea0e	9eca	d165	ce6d	ef5e	11a5	<e.m.^< td=""></e.m.^<>
1.5(000081e0:	64fd	1810	460c	670b	693a	f4e9	e9fc	483e	dF.g.i:H>
e.m.^	000081f0:	ca6c	73e3	c42e	c99d	96ed	1881	bb54	7cf0	.lsT .
.g.i:H>	00008200:	8be2	c613	861f	85fc	72fa	811b	092d	4dea	M.
T .	00008210:	567b	80e7	095a	9248	3cc8	7a5c	fc25	7c53	V{Z.H<.z\.% S
rM.	00008220:	95de	5151	3c27	ca2b	ba07	bc5a	c210	c9c9	QQ<'.+Z
Z.H<.z\.% S	00008230:	0519	bc3e	1cb5	b2c4	4be3	cf89	4727	6934	>KG'i4
'.+Z	00008240:	88b9	e2c5	b2ac	8773	ece9	432b	99c2	fb28	sC+(
KG'i4	00008250:	82d4	b218	cd6a	98bb	7a4a	1595	da85	c370	jzJp
sC+(- 1











U Shyve








Parallels



IU Bhyve

HUL	EPT MMIO INT	Nechamism
TIO	RAX	V+M
CIO	RAX	V+M
VMBus	RAX/RCX	Μ
/are	RDX	
/are	RDX	
CIO	RAX	V
Bus	RCX	V











U WARE[®] Bhyve





Intel[®] 64 and IA-32 Architectures Software Developer's Manual

Volume 3C: System Programming Guide, Part 3







Parallels



U Shyve



Intel[®] 64 and IA-32 Architectures Software Developer's Manual

Volume 3C: System Programming Guide, Part 3

	Virtualize APIC access	es Er	Enable EPT		Descriptor-table exiting			Enable RDTSCP		
	Virtualize x2APIC mo	de En	Enable VPID		WBINVD exiting		ing	Unrestricted guest		
Secondary	APIC-register virtu	alization	ation Virtual-inte		errupt delivery		PAUSE-loop exiting			
VM execution	RDRAND exiting	Enal	Enable INVPCID		Enable VM func		tions VMCS shadowing			
controls	Enable ENCLS exitin	g RDS	RDSEED exiting		Enable PML		L	EPT-violation #VE		
controls	Conceal VMX non-ro	oot operation	eration from Intel PT			Enable XSAVES/XRSTORS				
	Mode-based ex	ecute control for EPT			Use TSC scaling					
Excepti	on Bitmap	I/O-В	I/O-Bitmap Addresses			TSC-offset				
Guest/Host Masks for CR0 Guest/Host		t Masks for C	lasks for CR4		shadows for CR0		Read Shadows for CR4			
CR3-target value 0	CR3-target value	1 CR3-t	arget v	alue 2	CR3-target valu		ue 3	CR3-target count		
	APIC-access a	ddress		Virtual-APIC addres		ss		TPR threshold		
APIC Virtualization	EOI-exit bitmap	EOI-e	exit bitr	map 1	EOI-e	xit bitma	xit bitmap 2 EOI-exit bitmap 3			
6	Posted-interr	Posted-interrupt notificatio			P	osted-int	errupt des	scriptor address		
Read bitmap for low	v MSRs Read bitma	ap for high MS	for high MSRs Write			tmap for low MSRs		Write bitmap for low MSRs		
Executive-	Extended	Extended-Page-Table Poin			Virtual-Processor Identifier					
PLE_Gap	PLE_Window	VM-fur	nction o	controls	VMR	EAD bitm	ар	VMWRITE bitmap		
ENCLS-exiting bitmap					PML address					
Virtualization-except	ion information address		EPTP index			XSS-exiting bitmap				
	١	/M-EXIT	CON	ITROL	FIELDS					
	Save debug co	Host address space si			ize Load IA32_PERF_GLOBAL_CTRL					
VM-Exit Controls	Acknowledge interrupt on exit		ve IA32	IA32_PAT Load IA32_		PAT Sa	Save IA32_EFER Load IA32_E			
	Save VMX preemption timer value			Clear IA32_BNDCFGS Conceal VM exits from Int				VM exits from Intel PT		
VM-Exit Controls	VM-exit MSR-store count VM-exit MSR-store address						(
for MSRs	for MSRs VM-exit MSR-load count VM-exit MSR-load address									
VM-EXIT INFORMATION FIELDS										
Basic VM-Exit	Exit reason				Exit qualification					
Information	Guest-linear address				Guest-physical address					
VM Exits Due to Vectored Events VN			VM-exit interruption information			VM-exit interruption error code				
		IDT-vectoring information			IDT-vectoring error code					
VM Exits That Occur	During Event Delivery	IDT-v	ectorin	ig informa	ition		IDI-vecto	oring error code		
VM Exits That Occur	During Event Delivery	IDT-v VM-e	ectorin xit inst	ruction le	ngth	VN	A-exit instr	uction information		
VM Exits That Occur VM Exits Due to In	During Event Delivery	IDT-v VM-e I/O RC	ectorin xit insti X	ruction le	ngth O RSI		A-exit instr O RDI	ruction information		







Parallels



vmware[®] Bhyve



Intel[®] 64 and IA-32 Architectures Software Developer's Manual

Volume 3C: System Programming Guide, Part 3

	Virtualize APIC accesse	s Ena	Enable EPT		tor-table e	exiting	Enable RDTSCP		
Secondary processor-based	Virtualize x2APIC mod	e Enak	Enable VPID		/BINVD exiting		Unrestricted guest		
	APIC-register virtua	lization	Virtual-int	errupt deliv	delivery PAUSE-loop exiting				
	RDRAND exiting	Enable	e INVPCID	Enable	Enable VM functio		VMCS shadowing		
controls	Enable ENCLS exiting	RDSF	mg	Enable PML EPT-violat			EPT-violation #VE		
controis	Conceal VMX non-roo	ot operation		Enable XSAVES/XRSTORS					
	Mode-based exe	ecute co			Use TSC scaling				
Excepti	on Bitmap					TSC-c	offset		
Guest/Host Masks f	or CR0 Guest/Host	Masks fo	IC	hadows for	r CR0	I Shadows for CR4			
CR3-target value 0	CR3-target value 1	CR3-ta		CR3-ta	arget valu	e 3	CR3-target count		
3.	APIC-access ac	Idress	nrtual-	APIC addres	SS		TPR threshold		
APIC Virtualization	EOI-exit bitmap 0	EOI-exit bitmap 0 EOI-exit bit		EOI-exit bitmap 2			EOI-exit bitmap 3		
	Posted-interru	pt notification	vector	P	osted-inte	errupt de	address		
Read bitmap for low	MSRs Read bitmag	o for high MSR	s Write bit	map for lov	v MSRs	Writ	low MSRs		
Executive-	VMCS Pointer	Extended-F	Page-Table Poi	nter	Virt	ual 🖌			
PLE_Gap	PLE_Window VM-function controls			VMREAD bitmap bitmap					
E	ENCLS-exiting bitmap	63.			PML	addre			
Virtualization-except	ion information address		EPTP index		XSS-or cmap				
	V	M-EXIT C	ONTROL	FIELDS					
	debug con	trols	ress space size Load IA32_PERF_GLOBAL_CTRL						
VM-Exit Controls	A interrupt	on exit Save IA32_PAT		Load IA32_PAT Save IA		ve IA32_EI	2_EFER Load IA32_EFER		
	s otion	timer value	ner value Clear IA32_E			NDCFGS Conceal VM exits from Intel PT			
VM-Exit Controls	it l e cou	Int VM-exit MSR-store address							
for MSRs	d d cour	nt		VM-exit N	ASR-load a	address			
VM-EXIT INFORMATION FIELDS									
Basic VM-Exit	xit Exit reason				Exit gualification				
Information	Guest-linear address				Guest-physical address				
VM Exits Due to Vectored Events VM-exit interruption inform			nation	on VM-exit interruption error code					
VM Exits That Occur During Event Delivery IDT-vectoring infor			toring informa	tion IDT-vectoring error code					
		VM-exit instruction length			VM-exit instruction information				
VIVI EXITS Due to In	I/O RCX	RCX I/O RSI		1/0	O RDI	I/O RIP			
VM-instruction error field									







Overview





List of PIO/MMIO Regions

1. Make a Snapshot



1. Run the target hypervisor (L1) nested in KVM (L0) 2. Configure/Start a VM in L1 (L2) 3. Invoke a special "snapshot" hypercall in (L2) 4. Collect a memory/register snapshot of L1, just as it is about to handle the hypercall VM exit from L2.

2. Enumerate the Input Spaces



allocated to L2's memory and exits to userspace.

- 1. Load the snapshot into an emulator (Bochs) 2. Inspect the VMCS that L1 created for L2 to identify **MMIO Regions** and physical **frames** 3. Perform probing of IO input-space to identify
- active ports by tracking icounts, covered PCs

3. Fuzz the Hypervisor



1. Load the snapshot into the emulator generated **PIO/MMIO** the VMExit. 4. Instead of running the L2 VM, immeditately inject another fuzzer-provided VMexit. fill the read with fuzzer-provided data. (DMA) 6. Once the whole input is executed, reload the snapshot

2. Modify the register/VMCS state to inject fuzzer-

- 3. Resume the hypervisor and wait for it to handle
- 5. When the hypervisor reads from L2's memory,

Results: **QEMU Coverage**

	Morphuzz		ViDeZZo		HyperPill			
		12 Cores 24 Hours						
Device		Branch Coverage (Executions/Second)						
Block								
ahci	42.43%	(25.68)	30.42%	(562.24)	45.90%	(26.18)	✓	
nvme	29.12%	(23.82)			36.44%	(14.45)	1	
sdhci	69.81%	(22.98)	72.37%	(107.22)	66.85%	(32.34)		
virtio-scsi	27.96%	(23.83)	11.73%	(217.28)	48.83%	(51.68)		
Display					•			
cirrus	88.10%	(19.06)	83.42%	(138.78)	88.67%	(32.18)	✓	
qxl					59.68%	(26.96)	1	
virtio-gpu	24.37%	(26.21)	2.77%	(222.42)	45.52%	(36.53)	1	
Networking								
e1000e	50.27%	(24.83)	41.52%	(53.04)	55.99%	(42.22)	✓	
igb	29.73%	(25.63)			35.93%	(60.85)	\checkmark	
vmxnet	50.75%	(27.01)	19.64%	(145.73)	56.89%	(48.14)		
USB			-					
ehci	73.76%	(24.58)	74.38%	(177.08)	73.32%	(10.46)		
xhci	55.54%	(28.83)	29.25%	(1061.36)	76.64%	(69.26)	√	
Geo. Mean	45.20%	(24.65)	28.00%	(203.07)	55.45%	(33.20)		

Results: Bugs

Hyper-V

Heap-corruption in EthernetCard::HandleTransmitSetupFrame Abort in EthernetCard::PollForTransmitDataTimer Abort after IdeChannel::EnlightenedHddCommand EthernetCard::SetupEthernetCardModeFromRegisters Out-of-bounds write in GuestStateAccess::SetDeviceInfo Abort after PitDevice::NotifyIoPortRead Abort in I8042Device::HandleCommand Abort after HvCallDetachDevice Abort after HvCallGetGpaPagesAccessState macOS Virtualization Framework Memory-privilege violation in xHCI Out-of-bounds write in virtio-gpu Out-of-bounds write in virtio-audio Out-of-bounds access in virtio-block Out-of-bounds access in virtio-console Out-of-bounds access in virtio-net **QEMU** Arbitrary memory-access in e1000e_start_xmit Heap-overflow in usb_mouse_poll Heap-overflow in virtqueue_alloc_element Heap-overflow in qxl_cookie_new Heap-overflow in igb_tx_pkt_switch Out-of-bounds memory access in nvme_process_sq Out-of-bounds memory access in nvme_io_mgmt_send DoS via arbitrary-sized allocation in qxl DoS via arbitrary-sized allocation in virtio_gpu DoS in process_ncq_command DoS in icmp_input

Reshape hypervisors by modifying the CPU virtualization interface No modification to hypervisors code needed!

Fuzz any hypervisor across its PIO, MMIO, DMA, and Hypercall interfaces

More precise than source-level reshaping









31

No Grammar, No Problem: Towards Fuzzing the

Abstract—The integrity of the entire computing ecosystem enends on the security of our onerating systems (O.Ses) IInfor-Abstract—The integrity of the entire computing ecosystem depends on the security of our operating systems (OSes). Unfor-tunately, due to the scale and complexity of OS code, hundreds depends on the security of our operating systems (OSes). Unfor-tunately, due to the scale and complexity of OS code, hundreds of security issues are found in OSes. every vear [32]. As such. tunately, due to the scale and complexity of OS code, hundreds of security issues are found in OSes, every year [32]. As such, onerating systems have constantly been brime use-cases for [†]Boston University of security issues are found in OSes, every year [32]. As such, operating systems have constantly been prime use-cases for annlying security-analysis tools. In recent years, fuzz-testing has operating systems have constantly been prime use-cases for applying security-analysis tools. In recent years, fuzz-testing has anneared as the dominant technique for automatically finding applying security-analysis tools. In recent years, fuzz-testing has appeared as the dominant technique for automatically finding security issues in software. As such fuzzing has been adapted appeared as the dominant technique for automatically finding security issues in software. As such, fuzzing has been adapted to find thousands of buos in kernels [14]. However, modern OS security issues in software. As such, fuzzing has been adapted to find thousands of bugs in kernels [14]. However, modern OS fuzzers, such as Svzkaller, relv on nrecise, extensive, manually, to find thousands of bugs in kernels [14]. However, modern US fuzzers, such as Syzkaller, rely on precise, extensive, manually-created harnesses and grammars for each interface fuzzed within Iuzzers, such as Syzkaller, rely on precise, extensive, manually created harnesses and grammars for each interface fuzzed within the kernel. Due to this reliance on prammars, current OS fuzzed within created harnesses and grammars for each interface fuzzed within the kernel. Due to this reliance on grammars, current OS fuzzers are faced with scaling-issues. In this paper, we present FUZZNG, our generic approach to http://www.seem-calls on OSec. Unlike Svzkaller. FUZZNG does not In this paper, we present FUZZNG, our generic approach to fuzzing system-calls on OSes. Unlike Syzkaller, FUZZNG does not require intricate descriptions of system-call interfaces in order to fuzzing system-calls on OSes. Unlike Syzkaller, FUZZNG does not require intricate descriptions of system-call interfaces in order not function. Instead FUZZNG leverages fundamental kernel design

require intricate descriptions of system-call interfaces in order to function. Instead FUZZNG leverages fundamental kernel design features in order to reshape and simplify the fuzzer's input-space. function. Instead FUZZNG leverages fundamental kernel design features in order to reshape and simplify the fuzzer's input-space. As such FUZZNG only requires a small config for each new features in order to reshape and simplify the fuzzer's input-space. As such FUZZNG only requires a small config, for each new taroet. essentially a list of files and system-call numbers the fuzzer As such FUZZNG only requires a small config, for each new target: essentially a list of files and system-call numbers the fuzzer should explore. We implemented FUZZNG for the Linux kernel. Testing We implemented FUZZNG for the Linux kernel. Iesting FUZZNG over 10 Linux components with extensive descrip-tions in Svzkaller showed that, on average, FUZZNG achieves FUZZNG over 10 Linux components with extensive descrip-tions in Syzkaller showed that, on average, FUZZNG descrip-102.5% of Svzkaller's coverage. FUZZNG found 9 new hugs tions in Syzkaller showed that, on average, FUZZNG achieves 102.5% of Syzkaller's coverage. FUZZNG found 9 achieves (5 in components that Svzkaller had already fuzzed extensively) 102.5% of Syzkaller's coverage. FUZZNG found 9 new bugs (5 in components that Syzkaller had already fuzzed extensively, for vears) Additionally. FUZZNG's lightweight configs are less (5 in components that Syzkaller had already fuzzed extensively, for years). Additionally, FuzzNG's lightweight configs are less than 1.7% the size of Syzkaller's manually-written or ammars. for years). Additionally, FUZZNG's lightweight configs are less than 1.7% the size of Syzkaller's manually-written grammars. Crucially, FUZZNG achieves this without initial seed-inputs, or than 1.7% the size of Syzkaller's manually-written grammars. Crucially, FUZZNG achieves this without initial seed-inputs, or expert guidance.

The Operating System continues to serve as one of the The Operating System continues to serve as one of the most security-critical building blocks in modern computing. The OS' role in managing recources and enforcing isolation most security-critical building blocks in modern computing. The OS' role in managing resources and enforcing isolation hetween applications makes it a target for attackers who seek Ine OS role in managing resources and enforcing isolation between applications makes it a target for attackers who seek to violate OS-provided onarantees Recoonizing the critical between applications makes it a target for attackers who seek to violate OS-provided guarantees. Recognizing the critical nature of OC contribution fuzzers have identified and helped to violate US-provided guarantees. Recognizing the critical nature of OS security, fuzzers have identified and helped for those of burge in OS vernele perently the success hature of US security, fuzzers have identified and helped fix thousands of bugs in OS kernels. Recently, the success of OS fuzzare has amphasized difficulty of writing socura fix thousands of bugs in OS kernels. Recently, the success of OS fuzzers has emphasized difficulty of writing secure initiatives such as of US tuzzers has emphasized difficulty of writing secure low-level code, and has even spurred initiatives such as to the Linux kernel and the W-level code, and has even spurred initiatives such as upport for safer languages in the Linux kernel, and the how have to enable IPport for safer languages in the Linux kernel, and the age of hardware-features such as Memory Tagging to enable to a sub-to a state of the memory community age of hardware-reatures such as Memory Lagging to enable vanced low-overhead defenses against memory-corruption rk and Distributed System Security (NDSS) Symposium 2023 ruary - 3 March 2023, San Diego, CA, USA

Linux Kernel without System-Call Descriptions bsd@redhat.com stefanha@redhat.com Manuel Egele[†] *Red Hat megele@bu.edu issues [23], [44]. Most OS fuzzers focus on the critical system-call interface which enables rear enage applications to request issues [23], [44]. Most US tuzzers tocus on the critical system call interface, which enables user-space applications to request carving from the kernel Syzkaller[14], the most prolific system-call fuzzer, has ecome an integral commonent of the Linux Kernel develop. Syzkaller[14], the most prolific system-call fuzzer, has become an integral component of the Linux Kernel develop-ment lifecycle with over 2700 mentions in Vernel developbecome an integral component of the Linux Kernel development lifecycle, with over 2,700 mentions in kernel developmentions in kernel commit development developmen ment litecycle, with over 2,700 mentions in kernel commit messages. As such, syzkaller, itself, has grown to be a sizeable project with over 200 contributors. Crucially Syzkaller can messages. As such, syzkaller, itself, has grown to be a sizeable project, with over 200 contributors. Crucially, Syzkaller can only firzz evetem_calle that are enfliciently decorihed by a project, with over 200 contributors. Crucially, Syzkaller can only fuzz system-calls that are sufficiently described by a "evzlano" orammar Thece orammarc encode and annotate the only fuzz system-calls that are sufficiently described by a "syzlang" grammar. These grammars encode and annotate the two of recourses provided as innuts and returned as outputs Syziang grammar. I nese grammars encode and annotate the types of resources provided as inputs and returned as outputs, hv evetam-calle Tharafora much of the evolution of the e types of resources provided as inputs and returned as outputs, by system-calls. Therefore, much of the syzkaller community, work is focused around developing and refining "cvzlang" by system-calls. I herefore, much of the syzkaller community's work is focused around developing and refining "syzlang" decorrintions for evetem-calle which are eccential to Syzlang" Work is focused around developing and remning "syziang" descriptions for system-calls, which are essential to Syzkaller's

Developing such grammars is a manual process, and re-nires detailed knowledge about the interface (i e set of Developing such grammars is a manual process, and re-quires detailed knowledge about the interface (i.e., set of system calle) in question Ac such grammare are prome to quires detailed knowledge about the interface (i.e., set of system calls) in question. As such, grammars are prone to human-error and can lead to gang in coverage or over-fitting system calls) in question. As such, grammars are prone to human-error, and can lead to gaps in coverage, or over-fitting (nraventing the fuzzer from exploring all states and scenarios) numan-error, and can lead to gaps in coverage, or over-fitting (preventing the fuzzer from exploring all states and scenarios in which code could be covered, additionally evolved Teventing the fuzzer from exploring all states and scenarios A which code could be covered). Additionally, syzkaller A writing cumplementary harneceing code to In which code could be covered). Additionally, syzkaller sometimes requires writing supplementary harnessing code to fuzz narticularly complex interfaces. For example to fuzz the sometimes requires writing supplementary harnessing code to fuzz particularly complex interfaces. For example, to fuzz the I inity Kernel Virtual Machine (KVM) interface which nowers tuzz particularly complex interfaces. For example, to fuzz the Linux Kernel Virtual Machine (KVM) interface, which powers security-critical virtualization software Syzkaller developers Linux Nernei virtuai Machine (N v M) interface, which powers security-critical virtualization software, Syzkaller developers committed 201 lines of detailed evecall descriptions 242 security-critical virtualization software, SyzKaller developers committed 891 lines of detailed syscall descriptions, 243 $KVM_{-rolated}$ conctante and a further R70 lines of KVM_{-} committed 891 lines of detailed syscall descriptions, 243 KVM-related constants, and a further 879 lines of KVM-snecific C harnessing code (illustrated in Figure 1) Even K v M-related constants, and a turther 8/9 lines of K v M-specific C harnessing code (illustrated in Figure 1). K v M-though Svzkaller features tene-of-thousands of hand-orafted specific C harnessing code (illustrated in Figure 1). Even though Syzkaller features tens-of-thousands of hand-crafted "evzlano" rulee the current process cannot coale to find though Syzkauer reatures tens-or-mousands or hand-craited "syzlang" rules, the current process cannot scale to fuzz the millions of lines of code added to the I inux Kernel "syzlang" rules, the current process cannot scale to fuzz the millions of lines of code added to the Linux Kernel each vear 1231 Academic works have recomized Swallen's the millions of lines of code added to the Linux Kernel each year [33]. Academic works have recognized Syzkaller's coalahility problem with manually-created cvylang orammarc each year [33]. Academic works have recognized Syzkaller's scalability problem with manually-created syzlang grammars, and have focused on automatically oenerating grammars. scalability problem with manually-created syziang grammars, and have focused on automatically generating grammars, worke ench as Diffize IMF SyzGen and KSG apply static and have focused on automatically generating grammars. Works such as Difuze, IMF, SyzGen, and KSG apply static and dynamic-analysis techniques to automatically openerate Works such as Dituze, IMF, SyzGen, and KSG apply static and dynamic-analysis techniques to automatically static system_call descriptions [12] [18] [0] [51] Difuze IMF and and dynamic-analysis techniques to automatically generate system-call descriptions [12], [18], [9], [51]. Difuze, IMF and SvzGen are designed and evaluated against interfaces such System-call descriptions [12], [18], [9], [51]. Difuze, IMF and SyzGen are designed and evaluated against interfaces, such line manual-descriptions exist. KSG'e description been released and instra-

FUZZNG No Grammar, No Problem: Towards Fuzzing the Linux Kernel without System-Call Descriptions

NDSS 2023

Userspace



syscall %rdi %rsi %rdx %r10 %r8 %r9

Userspace



syscall FUZZ FUZZ FUZZ FUZZ FUZZ FUZZ

Userspace

Adding a New System Call

For more sophisticated system calls that involve a larger number of arguments, it's preferred to encapsulate the majority of the arguments into a structure that is passed in by **pointer**. Such a structure can cope with future extension by including a size argument in the structure

• • •

If your new system call allows userspace to refer to a kernel object, it should use a file descriptor as the handle for that object -- don't invent a new type of userspace object handle when the kernel already has mechanisms and well-defined semantics for using file descriptors.

linux-kernel/Documentation/process/adding-syscalls.rst

syscall FUZZ FUZZ FUZZ FUZZ FUZZ FUZZ

Userspace

fd = open("/var/log/messages", O_RDONLY)
read(fd, buf, 100);

Userspace



fd = open("/var/log/messages", O_RDONLY)
read(fd, buf, 100);
syscall 0x2 0xce51020 0x0 0x0 0x0 0x0
syscall 0x0 0x55a4e3c2a000 100 0x0 0x0

Userspace



fd = open("/var/log/messages", O_RDONLY)
read(fd, buf, 100);
syscall 0x2 0xce51020 0x0 0x0 0x0 0x0
syscall 0x0 0x55a4e3c2a000 100 0x0 0x0





Memory

 0ce51010
 5548
 89e5
 9090
 5dc3
 3030
 3030
 3030
 0a00
 UH....j.000000..

 0ce51020
 2f76
 6172
 2f6c
 6f67
 2f6d
 6573
 7361
 6765
 /var/log/message

 0ce51030
 7300
 4572
 726f
 7220
 6f70
 656e
 696e
 6720
 s.Error opening

 0ce51040
 6669
 6c65
 2e00
 5265
 6164
 696e
 6720
 7468
 file...Reading th

fd = open("/var/log/messages", 0_RDUNLY)
read(fd, buf, 100);

syscall0x20xce510200x00x00x00x0syscall0x00x00x55a4e3c2a0001000x00x0



 0ce51010
 5548
 89e5
 9090
 5dc3
 3030
 3030
 3030
 0a00
 UH....j.000000..

 0ce51020
 2f76
 6172
 2f6c
 6f67
 2f6d
 6573
 7361
 6765
 /var/log/message

 0ce51030
 7300
 4572
 726f
 7220
 6f70
 656e
 696e
 6720
 s.Error opening

 0ce51040
 6669
 6c65
 2e00
 5265
 6164
 696e
 6720
 7468
 file...Reading th

fd = open("/var/log/messages", 0_RDUNLY)
read(fd, buf, 100);

 syscall
 0x2
 0xce51020
 0x0
 0x0
 0x0
 0x0

 syscall
 0x0
 0x55a+3c2a000
 100
 0x0
 0x0



fd = open("/var/log/messages", O_RDONLY)
read(fd, buf, 100);
syscall 0x2 0xce51020 0x0 0x0 0x0 0x0
syscall 0x0 0x55a4e3c2a000 100 0x0 0x0





Memory

fd = open("/var/log/messages", O_RDONLY)
read(fd, buf, 100);
syscall 0x2 0xce51020 0x0 0x0 0x0 0x0
syscall 0x0 0x55a4e3c2a000 100 0x0 0x0

Userspace



Memory



Kernel

fd = open("/var/log/messages", O_RDONLY)
read(fd, buf, 100);
syscall 0x2 0xce51020 0x0 0x0 0x0 0x0
syscall 0x0 0x55a4e3c2a000 100 0x0 0x0



fd = open("/var/log/messages", O_RDONLY)
read(fd, buf, 100);
syscall 0x2 0xce51020 0x0 0x0 0x0 0x0 = 0x3
syscall 0x0 0x0 0x0 0x0 0x0 0x0 0x0



fd = open("/var/log/messages", 0_RDONLY)
read(fd, buf, 100);
syscall 0x2 0xce51020 0x0 0x0 0x0 0x0 = 0x3
syscall 0x0 0x3 0x55a4e3c2a000 100 0x0 0x0



syscall FUZZ FUZZ FUZZ FUZZ FUZZ FUZZ

Userspace

Kernel



Memory



syscall FUZZ FUZZ FUZZ FUZZ FUZZ FUZZ

Pointers and **File-Descriptors** result in an enormous system-call input-space







syz_io_uring_setup(entries int32[1:IORING_MAX_ENTRIES], params ptr[inout, io_uring_params], addr_ring vma, addr_sges vma, ring_ptr ptr[out, ring_ptr], sges_ptr ptr[out, sges_ptr]) fd_io_uring

io_uring_setup(entries int32[1:IORING_MAX_ENTRIES], params ptr[inout, io_uring_params]) fd_io_uring io_uring_enter(fd fd_io_uring, to_submit int32[0:IORING_MAX_ENTRIES], min_complete int32[0:IORING_MAX_CQ_ENTRIES], flags flags[io_uring_enter_flags], sigmask ptr[in, sigset_t], size len[sigmask]) io_uring_register\$IORING_REGISTER_BUFFERS(fd fd_io_uring, opcode const[IORING_REGISTER_BUFFERS], arg ptr[in, array[iovec_out]], nr_args len[arg]) io_uring_register\$IORING_UNREGISTER_BUFFERS(fd fd_io_uring, opcode const[IORING_UNREGISTER_BUFFERS], arg const[0], nr_args const[0]) io_uring_register\$IORING_REGISTER_FILES(fd fd_io_uring, opcode const[IORING_REGISTER_FILES], arg ptr[in, array[fd]], nr_args len[arg]) io_uring_register\$IORING_UNREGISTER_FILES(fd fd_io_uring, opcode const[IORING_UNREGISTER_FILES], arg const[0], nr_args const[0]) io_uring_register\$IORING_REGISTER_EVENTFD(fd fd_io_uring, opcode const[IORING_REGISTER_EVENTFD], arg ptr[in, fd_event], nr_args const[1]) io_uring_register\$IORING_UNREGISTER_EVENTFD(fd fd_io_uring, opcode const[IORING_UNREGISTER_EVENTFD], arg const[0], nr_args const[0]) io_uring_register\$IORING_REGISTER_FILES_UPDATE(fd fd_io_uring, opcode const[IORING_REGISTER_FILES_UPDATE], arg ptr[in, io_uring_files_update], nr_args len[arg:fds]) io_uring_register\$IORING_REGISTER_EVENTFD_ASYNC(fd fd_io_uring, opcode const[IORING_REGISTER_EVENTFD_ASYNC], arg ptr[in, fd_event], nr_args const[1]) io_uring_register\$IORING_REGISTER_PROBE(fd fd_io_uring, opcode const[IORING_REGISTER_PROBE], arg ptr[inout, io_uring_probe], nr_args len[arg:ops]) io_uring_register\$IORING_REGISTER_PERSONALITY(fd fd_io_uring, opcode const[IORING_REGISTER_PERSONALITY], arg const[0], nr_args const[0]) ioring_personality_id io_uring_register\$IORING_UNREGISTER_PERSONALITY(fd fd_io_uring, opcode const[IORING_UNREGISTER_PERSONALITY], arg const[0], nr_args ioring_personality_id)

The mmap'ed area for SQ and CQ rings are really the same -- the difference is # accounted for with the usage of offsets. mmap\$IORING_OFF_SQ_RING(addr vma, len len[addr], prot flags[mmap_prot], flags flags[mmap_flags], fd fd_io_uring, offset const[IORING_OFF_SQ_RING]) ring_ptr mmap\$IORING_OFF_CQ_RING(addr vma, len len[addr], prot flags[mmap_prot], flags flags[mmap_flags], fd fd_io_uring, offset const[IORING_OFF_CQ_RING]) ring_ptr mmap\$IORING_OFF_SQES(addr vma, len len[addr], prot flags[mmap_prot], flags flags[mmap_flags], fd fd_io_uring, offset const[IORING_OFF_SQES]) sqes_ptr

If no flags are specified(0), the io_uring instance is setup for interrupt driven IO. io_uring_setup_flags = 0, IORING_SETUP_IOPOLL, IORING_SETUP_SQPOLL, IORING_SETUP_SQ_AFF, IORING_SETUP_CQSIZE, IORING_SETUP_CLAMP, IORING_SETUP_ATTACH_WQ io_uring_enter_flags = IORING_ENTER_GETEVENTS, IORING_ENTER_SQ_WAKEUP $_ = __NR_mmap2$

Once an io_uring is set up by calling io_uring_setup, the offsets to the member fields # to be used on the mmap'ed area are set in structs io_sqring_offsets and io_cqring_offsets. # Except io_sqring_offsets.array, the offsets are static while all depend on how struct io_rings # is organized in code. The offsets can be marked as resources in syzkaller descriptions but # this makes it difficult to generate correct programs by the fuzzer. Thus, the offsets are # hard-coded here (and in the executor). define SO_HEAD_OFFSET 0 define SQ_TAIL_OFFSET 64 define SQ_RING_MASK_OFFSET 256 define SO_RING_ENTRIES_OFFSET 264 define SO_FLAGS_OFFSET 276

io_uring_setup(entries int32[1:IORING_MAX_ENTRIES], params ptr[inout, io_uring_params]) fd_io_uring

io_uring_setup(entries int32[1:IORING_MAX_ENTRIES], params ptr[inout, io_uring_params]) fd_io_uring

io_uring_register\$IORING_REGISTER_PROBE(fd fd_io_uring, opcode const[IORING_REGISTER_PROBE],

arg ptr[inout, io_uring_probe], nr_args len[arg:ops])

io_uring_setup(entries int32[1:IORING_MAX_ENTRIES], params ptr[inout, io_uring_params]) fd_io_uring

io_uring_register\$IORING_REGISTER_PROBE(fd fd_io_uring, opcode const[IORING_REGISTER_PROBE],

```
io_uring_probe {
   last_op const[0, int8]
   ops_len const[0, int8]
   resv const[0, int16]
   resv2 array[const[0, int32], 3]
   ops array[io_uring_probe_op, 0:IORING_OP_LAST]
io_uring_probe_op {
   op const[0, int8]
   resv const[0, int8]
   flags const[0, int16]
   resv2 const[0, int32]
```

arg ptr[inout, io_uring_probe], nr_args len[arg:ops])

io uring enter(2) Linux Programmer's Manual io uring enter(2)

NAME

io_uring_enter - initiate and/or complete asynchronous I/0

SYNOPSIS

DESCRIPTION

io_uring_enter(2) is used to initiate and complete I/O using the shared submission and completion queues setup by a call to io_uring_setup(2). A single call can both submit new I/O and wait for completions of I/O initiated by this call or previous calls to io_uring_enter(2).

fd is the file descriptor returned by io_uring_setup(2).
to_submit specifies the number of I/Os to submit from the
submission queue. flags is a bitmask of the following values:

IORING_ENTER_GETEVENTS

If this flag is set, then the system call will wait for the specified number of events in *min_complete* before returning. This flag can be set along with *to_submit* to both submit and complete events in a single system call.
System-Call Grammars

Linux Programmer's Manual io uring enter(2) io uring enter(2)

NAME

io uring enter - initiate and/or complete asynchronous I/O

SYNOPSIS

#include <liburing.h> int io uring enter(unsigned int fd, unsigned int to submit, **unsigned int** *min complete*, **unsigned int** flags, sigset t *sig); int io uring enter2(unsigned int fd, unsigned int to submit, **unsigned int** *min complete*, **unsigned int** flags, sigset t *sig, size t sz);

DESCRIPTION

io uring enter(2) is used to initiate and complete I/O using the shared submission and completion gueues setup by a call to io uring setup(2). A single call can both submit new I/O and wait for completions of I/O initiated by this call or previous calls to io uring enter(2).

fd is the file descriptor returned by io uring setup(2). to submit specifies the number of I/Os to submit from the submission queue. *flags* is a bitmask of the following values:

IORING ENTER GETEVENTS

If this flag is set, then the system call will wait for the specified number of events in *min complete* before returning. This flag can be set along with to submit to both submit and complete events in a single system call. long ret = -EBADF; **int** submitted = 0; struct fd f;

Ł

io_run_task_work();

f = fdget(fd);if (!f.file)

ret = -EOPNOTSUPP:

ret = -ENXIO;

```
SYSCALL_DEFINE6(io_uring_enter, unsigned int, fd, u32, to_submit,
        u32, min_complete, u32, flags, const void __user *, argp,
        size_t, argsz)
    struct io_ring_ctx *ctx;
    if (flags & ~(IORING_ENTER_GETEVENTS | IORING_ENTER_SQ_WAKEUP |
            IORING_ENTER_SQ_WAIT | IORING_ENTER_EXT_ARG))
        return -EINVAL:
        return -EBADF;
    if (f.file->f_op != &io_uring_fops)
        goto out_fput;
    ctx = f.file->private_data;
    if (!percpu_ref_tryget(&ctx->refs))
        goto out_fput;
```

System-Call Grammars

4000+ Lines of Code to Describe a Single Subsystem (KVM)



Current System-Call Fuzzers rely on detailed grammars to describe pointer and file-descriptor arguments

Reshape the pointer and file-descriptor input-spaces to make system-calls conducive to off-the-shelf fuzzing methods









syscall FUZZ FUZZ FUZZ FUZZ FUZZ FUZZ

syscall FUZZ FUZZ FUZZ FUZZ FUZZ FUZZ

What will it take to make fuzzer-generated pointers and file-descriptors result in meaningful target behaviors?



Kernel



Memory







Memory





Memory



Files



Memory





Memory





copy_from_user()

Memory





copy_from_user()



Memory







copy_from_user()

Memory







Memory







Memory









c เ≣

















Config Setup

```
files = "/dev/kvm", O_RDWR
ioctl[-1, -1, -1]
mmap[0, 0xF000, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_POPULATE, 0xFFFF, -1]
close[-1]
fstat[-1, -1]
read[-1, -1, 0xFFFF]
write[-1, -1, 0xFFFF]
```





```
files = "/dev/kvm", O_RDWR
ioctl[-1, -1, -1]
mmap[0, 0xF000, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_POPULATE, 0xFFFF, -1]
close[-1]
fstat[-1, -1]
read[-1, -1, 0xFFFF]
write[-1, -1, 0xFFFF]
```

Setup Inflate Memory Start up threads to fill memory-accesses



Interpreter

```
files = "/dev/kvm", O_RDWR
ioctl[-1, -1, -1]
mmap[0, 0xF000, PROT_READ | PROT_WRITE, MAP_SHARED|MAP_POPULATE, 0xFFFF, -1]
close[-1]
fstat[-1, -1]
read[-1, -1, 0xFFFF]
write[-1, -1, 0xFFFF]
```



Interpreter


```
files = "/dev/kvm", O_RDWR
ioctl[-1, -1, -1]
mmap[0, 0xF000, PROT_READ | PROT_WRITE, MAP_SHARED|MAP_POPULATE, 0xFFFF, -1]
close[-1]
fstat[-1, -1]
read[-1, -1, 0xFFFF]
write[-1, -1, 0xFFFF]
```



Interpreter

```
files = "/dev/kvm", O_RDWR
ioctl[-1, -1, -1]
mmap[0, 0xF000, PROT_READ | PROT_WRITE, MAP_SHARED|MAP_POPULATE, 0xFFFF, -1]
close[-1]
fstat[-1, -1]
read[-1, -1, 0xFFFF]
write[-1, -1, 0xFFFF]
```





Interpreter







Interpreter




NG-Agent

Interpreter







NG-Agent

Interpreter

```
files = "/dev/kvm", O_RDWR
ioctl[-1, -1, -1]
mmap[0, 0xF000, PROT_READ | PROT_WRITE, MAP_SHARED|MAP_POPULATE, 0xFFFF, -1]
close[-1]
fstat[-1, -1]
read[-1, -1, 0xFFFF]
write[-1, -1, 0xFFFF]
```





NG-Agent

Interpreter

```
files = "/dev/kvm", O_RDWR
ioctl[-1, -1, -1]
mmap[0, 0xF000, PROT_READ | PROT_WRITE, MAP_SHARED|MAP_POPULATE, 0xFFFF, -1]
close[-1]
fstat[-1, -1]
read[-1, -1, 0xFFFF]
write[-1, -1, 0xFFFF]
```





• Linux 5.12

- Linux 5.12
- 13 Components

bpf video4linux rdma binder cdrom kvm vhost_net drm io_uring vt_ioctl ptmx nvme megaraid

- Linux 5.12
- 13 Components
- Coverage
 - Fuzzed for 7 Days on 20 Cores
 - 102.5% of Syzkaller's Coverage
 - Configurations <1.7% of Syzkaller's

bpf video4linux rdma binder cdrom kvm vhost_net drm io_uring vt_ioctl ptmx nvme megaraid



		Syzkaller		FUZZNG	
Component	Max Cov	Edge Count	Syzlang LoC	Edge Count	Config LoC
bpf	15359	3623	864	3572	1
video4linux	1004	563	381	567	4
rdma	4014	562	1474	591	5
binder	2506	340	272	344	6
cdrom	956	138	351	144	5
kvm	34924	9213	891	9468	7
vhost_net	415	218	157	225	9
drm	12503	2296	745	2138	7
io_uring	3413	982	343	1003	6
vt_ioctl	332	142	381	162	9
Average				102.53%	1.67%
Geo. Mean				102.41%	1.09%

Bend the system-call input space to make it conducive to fuzzing

Use time-tested off-the-shelf fuzzers

Competitive fuzzing performance with tiny component configs









Hypervisor Kernel



Reshaping applies to independent, complex targets



Target

Reshaping Level



Hypervisor Kernel Source-Level **ISA-Level**

Reshaping can be applied even without access to a target's source



Reshaping composes with diverse fuzzing techniques



Reshaping composes with diverse fuzzing techniques

- RQ1: effective at finding bugs?
- RQ2: competitive with other approaches on coverage-achieved?
- RQ3: applicable to a diverse set of targets?
- RQ4: beneficial even when grammars exist?
- RQ5: compatible with other SoTA fuzzing techniques?

Is reshaped fuzzing... RQ1: effective at finding bugs? RQ2: competitve with other approaches on coverage-achieved? RQ3: applicable to a diverse set of targets? RQ4: beneficial even when grammars exist? **RQ5:** compatible with other SoTA fuzzing techniques?

Over 100 new bugs found. Bugs found in code that was already "covered" by grammar-based fuzzers.

- RQ1: effective at finding bugs?
- RQ2: competitive with other approaches on coverage-achieved?
- RQ3: applicable to a diverse set of targets?
- RQ4: beneficial even when grammars exist?
- **RQ5:** compatible with other SoTA fuzzing techniques?

RQ1: effective at finding bugs? RQ2: competitive with other approaches on coverage-achieved?

RQ3: applicable to a diverse set of targets?

RQ4: beneficial even when grammars exist? **RQ5:** compatible with other SoTA fuzzing techniques?

Morphuzz, FuzzNG are extensible to other targets. HyperPill applies to an entire class of diverse targets.

RQ1: effective at finding bugs?

- RQ2: competitive with other approaches on coverage-achieved?
- **RQ3:** applicable to a diverse set of targets?
- RQ4: beneficial even when grammars exist? **RQ5:** compatible with other SoTA fuzzing techniques?
- Morphuzz and FuzzNG found bugs in code and components that had already been fuzzed by grammarbased approaches.

RQ1: effective at finding bugs?

- RQ2: competitive with other approaches on coverage-achieved?
- RQ3: applicable to a diverse set of targets?
- **RQ4:** beneficial even when grammars exist?
- RQ5: compatible with other SoTA fuzzing techniques?
- Our reshaping-based fuzzers integrate with diverse stateof-the-art techiques, such as fork-servers, full-systemfuzzing and emulator-based fuzzing.

- Is reshaped fuzzing...
- RQ1 effective at finding bugs?
- RQ2; competitve with other approaches on coverage-achieved?
- RQ3: applicable to a diverse set of targets?
- RQ4: beneficial even when grammars exist?
- RQ5: compatible with other SoTA fuzzing techniques?

Thesis: Input-space reshaping is more effective than grammar-based harnessing approaches for fuzzing complex targets.

Sok: Enabling Security Analyses of Embedded Systems via **Rehosting**



virtualizing embedded software

rehosting embedded systems.



Introduces the benefits and challenges of of

Identifies the essential steps in the rehosting process and a high-level, iterative process for

Saphire: Sandboxing PHP Applications with Tailored System Call Allowlists



kernel

Blocked every exploit in our dataset



Saphire: Sandboxing PHP Applications with Tailored System Call Allowlists Alexander Bulekov* Rasoul Jahanshahi* I contribution joint first authors Boston University Manuel Egele alxndr,rasoulj,megele}@bu.edu Abstract

er PHP, web ap.

1 Introduction

erpreted languages, s , such as PHP and JavaScr

ARTIFACT EVALUATED

Associa

nSSH, but it is not in common r such as Chrome, Firefox. Tor OF as been widely adopted b

- Three stage approach to automatically protect a vulnerable-web application against exploitation.
- Leverages, seccomp, a built-in feature of the

Fasano, Andrew, Tiemoko Ballo, Marius Muench, Tim Leek, Alexander Bulekov, Brendan Dolan-Gavitt, Manuel Egele et al. "Sok: Enabling security analyses of embedded systems via rehosting." In Proceedings of the 2021 ACM Asia conference on computer and communications security.

Bulekov, Alexander, Rasoul Jahanshahi, and Manuel Egele. "Saphire: Sandboxing {PHP} applications with tailored system call allowlists." In USENIX Security Symposium (USENIX Security 21)

Bulekov, Alexander, Bandan Das, Stefan Hajnoczi, and Manuel Egele. "MORPHUZZ: Bending (input) space to fuzz virtual devices." In USENIX Security Symposium (USENIX Security 22)

Bulekov, Alexander, Bandan Das, Stefan Hajnoczi, and Manuel Egele. "No grammar, no problem: Towards fuzzing the linux kernel without system-call descriptions." In Network and Distributed System Security (NDSS) Symposium 2023

Bulekov, Alexander, Qiang Liu, Manuel Egele, and Mathias Payer Hyperpill: Fuzzing for hypervisor-bugs by leveraging the hardware virtualization interface (to appear). In USENIX Security Symposium (USENIX Security 24)

Prof. Manuel Egele Prof. Gianluca Stringhini Prof. Orran Krieger Prof. Mathias Payer

Prof. Manuel EgeleProf. Gianluca StringhiniProf. Orran KriegerProf. Mathias Payer

Qiang Liu	Rasoul Jahanshahi	Stefan Ha
Andrew Fasano	Tiemoko Ballo	Jeremy
Marius Muench	Tim Leek	Brendan Do
Aurélien Francillon	Long Lu	Davide Ba

ajnoczi Bandan Das y Liu Darren Kenny lan-Gavitt William Robertson alzarotti Nick Gregory

Prof. Manuel EgeleProf. Gianluca StringhiniProf. Orran KriegerProf. Mathias Payer

Qiang Liu	Rasoul Jahanshahi	Stefan Ha
Andrew Fasano	Tiemoko Ballo	Jeremy
Marius Muench	Tim Leek	Brendan Do
Aurélien Francillon	Long Lu	Davide Ba



ajnoczi Bandan Das y Liu Darren Kenny olan-Gavitt William Robertson alzarotti Nick Gregory

Prof. Manuel EgeleProf. Gianluca StringhiniProf. Orran KriegerProf. Mathias Payer

Qiang Liu Rasoul Jahanshahi Stefan Hajnoczi Bandan Das Darren Kenny Jeremy Liu Tiemoko Ballo Andrew Fasano Tim Leek Brendan Dolan-Gavitt William Robertson Marius Muench Long Lu Nick Gregory Davide Balzarotti Aurélien Francillon





Prof. Manuel EgeleProf. Gianluca StringhiniProf. Orran KriegerProf. Mathias Payer

Qiang Liu Rasoul Jahanshahi Stefan Hajnoczi Bandan Das Jeremy Liu Darren Kenny Andrew Fasano Tiemoko Ballo Tim Leek Brendan Dolan-Gavitt William Robertson Marius Muench Long Lu Nick Gregory Aurélien Francillon Davide Balzarotti









Prof. Manuel EgeleProf. Gianluca StringhiniProf. Orran KriegerProf. Mathias Payer

Qiang Liu Rasoul Jahanshahi Stefan Hajnoczi Bandan Das Jeremy Liu Darren Kenny Andrew Fasano Tiemoko Ballo Tim Leek Brendan Dolan-Gavitt William Robertson Marius Muench Long Lu Nick Gregory Aurélien Francillon Davide Balzarotti









My Family



Morphuzz



HyperPill



FuzzNG



Ignore DMA

Tavis Ormandy. **IOFuzz** (2007) Henderson et al. **VDF** (Raid '17)

Schumilo et al. HYPER-CUBE (NDSS '20)

Schumilo et al. **HYPER-CUBE** (Schumilo et al. **Nyx** (Sec '21)

Per-Device Grammars

Schumilo et al. Nyx (Sec '21)



Grammars - Virtual Devices



Endpoint Context

Ring Base Address

EP State
Dequeue Po
Enqueue Po

TD
Hending I Ha
=moty I HB
Link i HB
Legend

Transfer Descriptors (Chained TRBs) may cross Segment boundaries.

xHC.

As software advances its Enqueue Pointer and advances over a Link TRB, the Cycle (C) bit shall be updated with the value of the PCS flag.

The Interrupt On Completion (IOC) flag of a Link TRB may be used by system software to generate an event indicating the Dequeue Pointer has reached the Link TRB. This feature provides software with the ability to track the Dequeue Pointer as a function of segment boundary crossings.

When the Link TRB resides on a Transfer Ring the Interrupt On Completion (IOC) flag of a Link TRB may be used by system software to generate a Transfer Event, where the Transfer Event Slot ID and Endpoint ID shall reflect the slot and

eXtensible Host Controller Interface for **Universal Serial Bus**

(xHCI)

Requirements Specification

May 2019

Revision 1.2

Figure 4-15: Link TRB Example



Refer to section 4.11.7 for how the Chain (CH) flag shall be set in a Link TRB. In a Transfer Ring a Link TRB is always assumed to be linked to the first TRB of the next segment. If the Chain bit (CH) of the previous TRB is '1', then the multi-TRB TD that it defines spans segments and shall continue with the first TRB of the next segment. In a Command Ring the Link TRB Chain bit (CH) is ignored by the

Note: A TD Fragment shall not span segments. Refer to section 4.11.7.1.

Grammars - Virtual Devices

```
t_dcb_ctx = s.edge_type("dcb_ctx", c_type= "uint32_t")
new_dcb_ctx=" *output_0 = (uint32_t) slab_alloc_page_aligend(0x1000);\n" +\
    "uint32_t* dcbaa = (uint32_t* )*borrow_0;\n"+\
    "dcbaa[*data_slot] = *output_0;\n"
```

```
s.node_type("new_dcb_ctx", outputs=[t_dcb_ctx], borrows=[t_dcbaa], data=s.data_u8("slot"), codegen_args=new_dcb_ctx)
```

```
new_dcb_ctx_broken_1="*((uint32_t**)borrow_0)[*data_u8] = (uint32_t)0x%x;\n"%(base)
s.node_type("new_dcb_ctx_broken_1", borrows=[t_dcbaa], data=d_u8, codegen_args=new_dcb_ctx_broken_1)
```

```
new_dcb_ctx_broken_2=" *((uint32_t**)borrow_0)[*data_u8] = (uint32_t)0xFFFFFFFFFFFFF00;\n"
s.node_type("new_dcb_ctx_broken_2", borrows=[t_dcbaa], data=d_u8, codegen_args=new_dcb_ctx_broken_2)
```

```
d_slot_context = s.data_struct("d_slot_context")
d_slot_context.u32("data_a")
d_slot_context.u32("data_b")
d_slot_context.u32("data c")
d_slot_context.u32("data_d")
d_slot_context.finalize()
new slot context = "" +\
 "*((uint32 t*)(*borrow 0 + 0)) = (uint32 t)data d slot context->data a;\n"+\
 "*((uint32_t*)(*borrow_0 + 4)) = (uint32_t)data_d_slot_context->data_b;\n"+\
 "*((uint32_t*)(*borrow_0 + 8)) = (uint32_t)data_d_slot_context->data_c;\n"+\
 "*((uint32_t*)(*borrow_0 + 12)) = (uint32_t)data_d_slot_context->data_d;\n"
s.node_type("new_slot_context", borrows=[t_dcb_ctx], data=d_slot_context, codegen_args=new_slot_context)
d_slot_context = s.data_struct("d_ep_context")
d_slot_context.u32("data_a")
d_slot_context.u32("data_b")
d_slot_context.u32("data_c")
d_slot_context.u8("ep_identifier")
d_slot_context.finalize()
new ep context = "" +\
 "*((uint32_t*)(*borrow_0 + 0 + (0x20 * (1+(uint32_t)data_d_ep_context->ep_identifier%0x30)))) = (uint32_t)data_d_ep_context->data_a;\n"+\
 "*((uint32_t*)(*borrow_0 + 4 + (0x20 * (1+(uint32_t)data_d_ep_context->ep_identifier%0x30)))) = (uint32_t)data_d_ep_context->data_b;\n"+\
  "*((uint32_t*)(*borrow_0 + 16 + (0x20 * (1+(uint32_t)data_d_ep_context->ep_identifier%0x30)))) = (uint32_t)data_d_ep_context->data_c;\n"+\
 "*((uint32 t*)(*borrow 0 + 8 + (0x20 * (1+(uint32 t)data d ep context->ep identifier%0x30)))) = *borrow 1;\n" + \
  "*((uint32_t*)(*borrow_0 + 12 + (0x20 * (1+(uint32_t)data_d_ep_context->ep_identifier%0x30)))) = 0;\n"
s.node_type("new_ep_context", borrows=[t_dcb_ctx, t_trb_ring], data=d_slot_context, codegen_args=new_ep_context)
```
- 1 Hypervisor Hooks-
- ② Virtual Device Fuzzer Harness
 - libFuzzer
 - Input interpreter
 - We added a fork-server
- ③ Crash "Unbending" Transform DMA back into an asynchronous operation

1 Prepare the Hypervisor	
Identify and Hook:	Guest-Address Map 🔾
	DMA-Access API 🧲

1 Hypervisor Hooks



1 Hypervisor Hooks



1 Hypervisor Hooks





1 Hypervisor Hooks



1 Hypervisor Hooks



1 Hypervisor Hooks





Morphuzz is Upstream in QEMU

- Continuously fuzzed on OSS-Fuzz
- 200+ Issues Reported
- Reproducers are simple to use
- Bugs are caught before release

git clone git.qemu.org/qemu.git



commit 283f0a05e24a5e5fab78305f783f06215390d620

Author: Thomas Huth <thuth@redhat.com> Date: Thu Jul 15 21:32:19 2021 +0200

hw/net/net_tx_pkt: Fix crash detected by fuzzer

QEMU currently crashes when it's started like this:

cat << EOF | ./gemu-system-i386 -device vmxnet3 -nodefaults -gtest stdio</pre> outl 0xcf8 0x80001014 outl 0xcfc 0xe0001000 outl 0xcf8 0x80001018 outl 0xcf8 0x80001004 outw 0xcfc 0x7 outl 0xcf8 0x80001083 write 0x0 0x1 0xe1 write 0x1 0x1 0xfe write 0x2 0x1 0xbe write 0x3 0x1 0xba writeg 0xe0001020 0xefefff5ecafe0000 writeg 0xe0001020 0xffff5e5ccafe0002 EOF It hits this assertion: gemu-system-i386: ../gemu/hw/net/net_tx_pkt.c:453: net_tx_pkt_reset: Assertion `pkt->raw' failed. This happens because $net_tx_pkt_init()$ is called with $max_frags == 0$ and

thus the allocation