Hyntrospect: A Fuzzer for Hyper-V Devices

Diane Dubois - didu@google.com - @0xdidu
SSTIC
June 2 2021

Whoami?

- Security Engineer at Google
- 20% with Project Zero

- Passionate about vulnerability research on systems
- @0xdidu

Why Hyper-V?

- Project Zero mission: aims to reduce harm caused by targeted attacks on the Internet

- Hyper-V
  - Hypervisor running Azure, Microsoft Cloud
  - Modern versions of Windows run it (Virtualization-based security)
  - Possibly a high impact if there are 0-days in the wild

- … and because virtualization is a fun topic
  - Spans multiple layers from hardware to high level software
  - Some complex implementations

Microsoft Hyper-V

Goals

- Instrumenting Hyper-V for vulnerability research
  - A fuzzer called Hyntrospect was developed and open sourced
    https://github.com/googleprojectzero/Hyntrospect
    - Coverage-guided
    - On closed-source binaries
    - Pragmatic approach, using existing Hyper-V features and Windows tools
    - In a real execution environment
  - Its internals and current results (coverage...) are presented

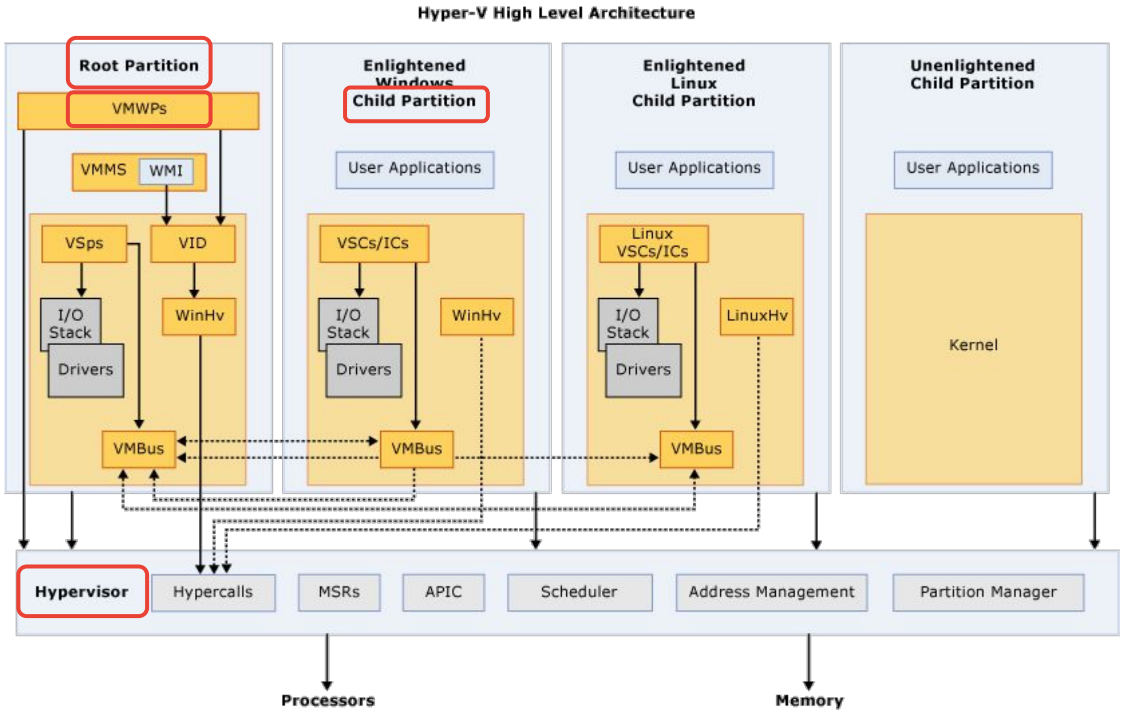
- Finding vulnerabilities and reporting them to Microsoft

Agenda

- Background on Hyper-V
- The Research Target
- Hyntrospect fuzzer
- Current results
- Future endeavours

Background on Hyper-V

Hyper-V Architecture Overview



Source: Microsoft

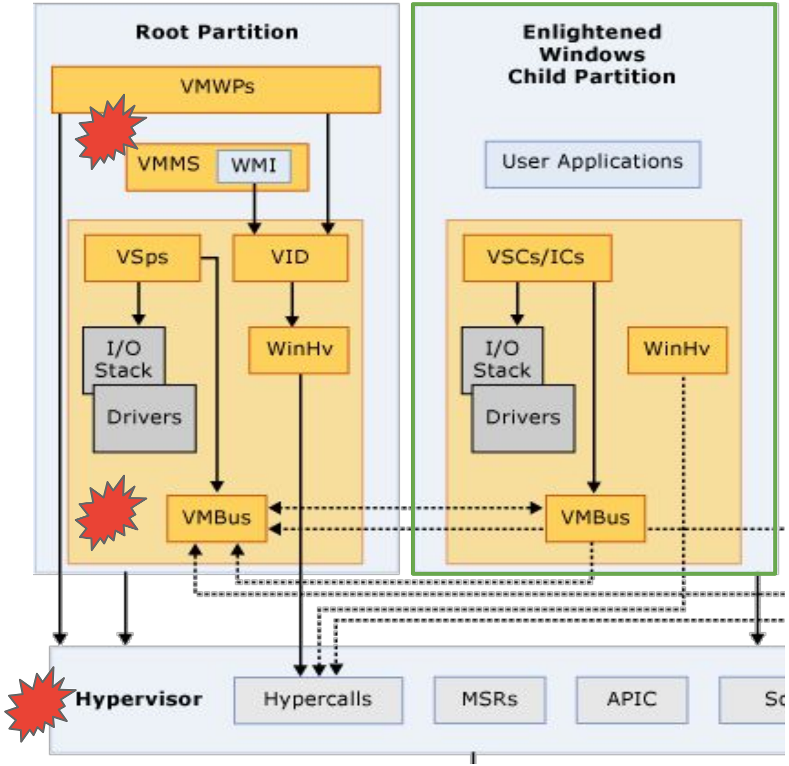
Google

What is a “Guest to Host Escape”?

- Gaining **code execution on** one of the **hypervisor layers from a virtual machine**

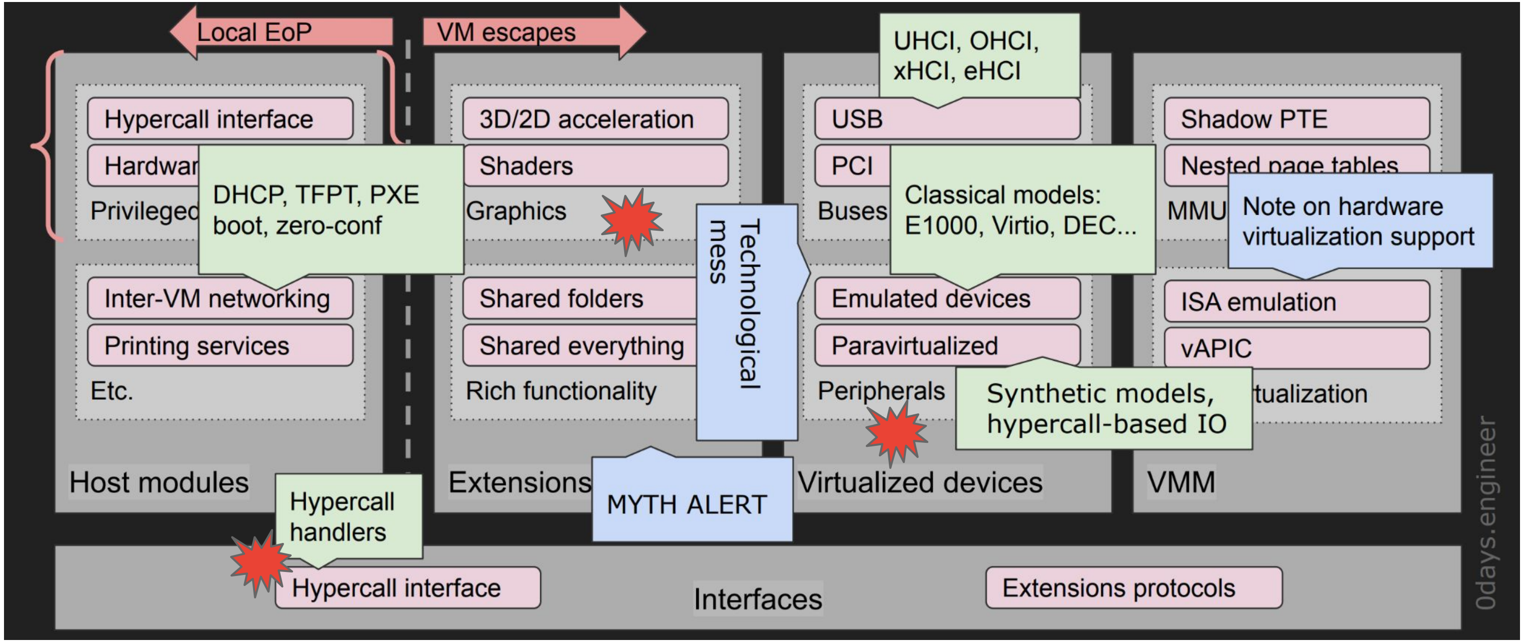
- On Hyper-V: ambiguous
    - Hypervisor layer
    - Root partition (kernel / userland)

- Other type of attack: **host denial of service**

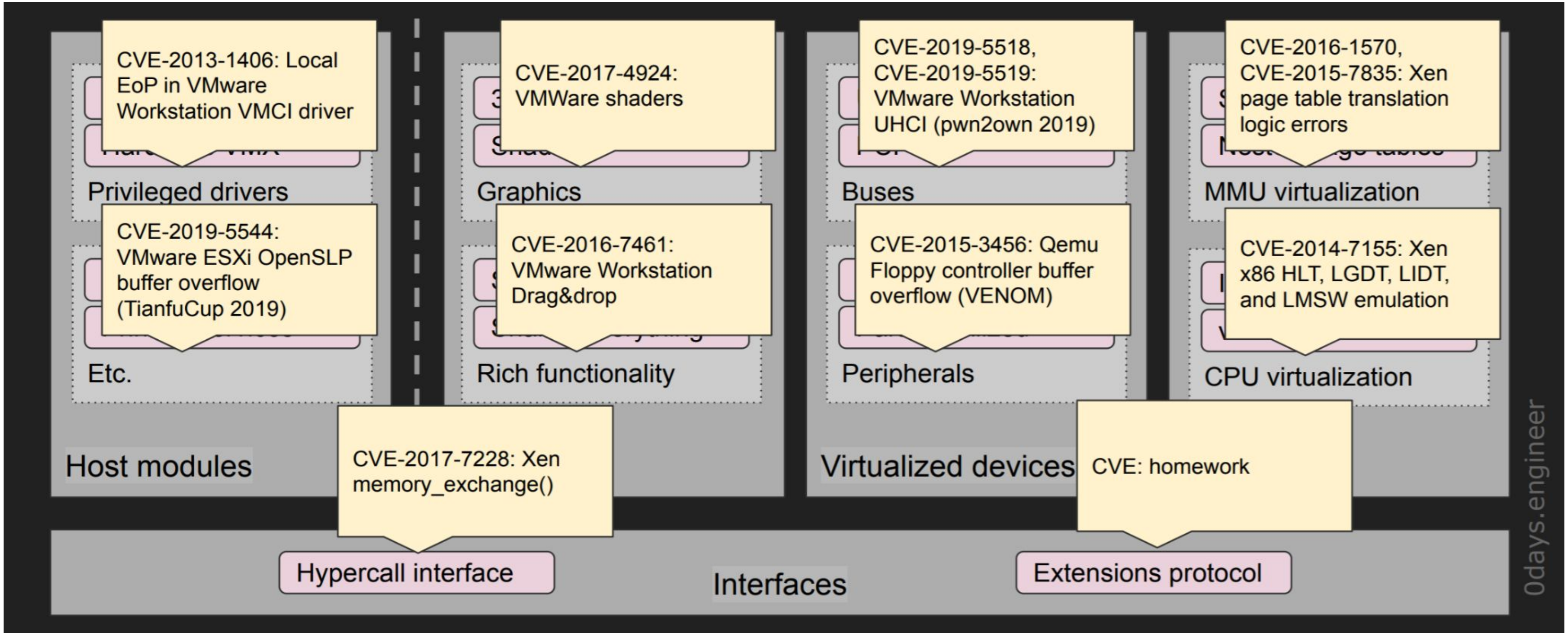


The Hypervisors' Attack Surface

- As defined by Alisa Esage (Zer0Con 2020):



… and in practice



Hyper-V Attack Surface

- Hypervisor
  - Hypercall handlers
  - Faults
  - Instruction emulation
  - Intercepts
  - Register access (MSRs...)
- Root partition kernel attack surface
  - VMBus
- Root partition userland attack surface
  - Emulated devices
  - Integration components

- … and this list is not exhaustive
- MSRC: first-steps-in-hyper-v-research

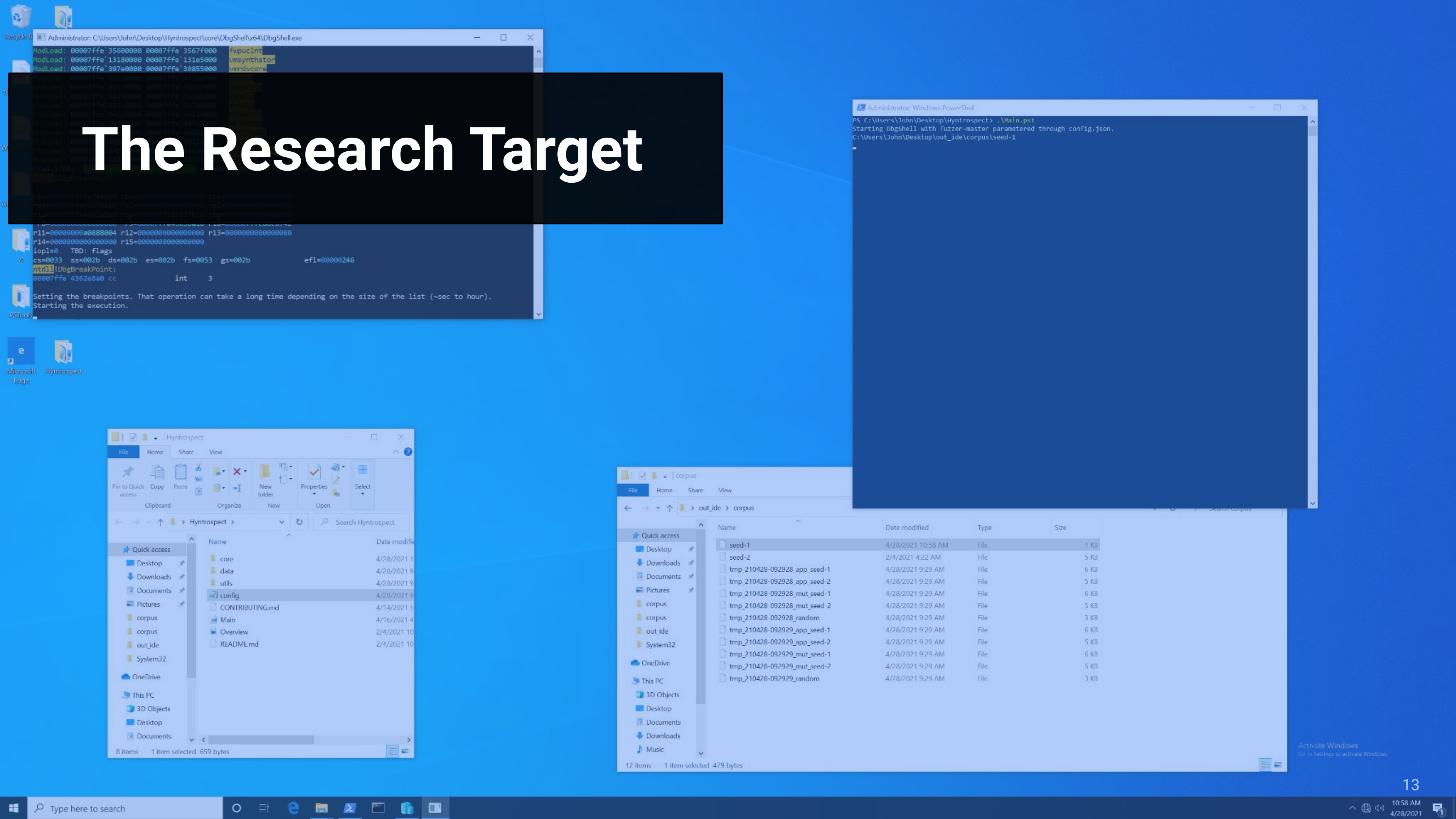
State of the Art on the Research

- MSRC and Microsoft publish on Hyper-V
  - Blog posts to help vulnerability researchers
    - e.g. First Steps in Hyper-V research
  - Posts on Hyper-V components
  - Several presentations at conferences on vulnerabilities found internally
    - e.g. Breaking VSM by Attacking SecureKernel at BlackHatUSA 2020
  - Symbols provided for some key components

- Active external contributors
  - @gerhart_x and his dedicated blog
  - @alisaesage
  - Presentation by Damien Aumaitre on whvp at SSTIC 2020

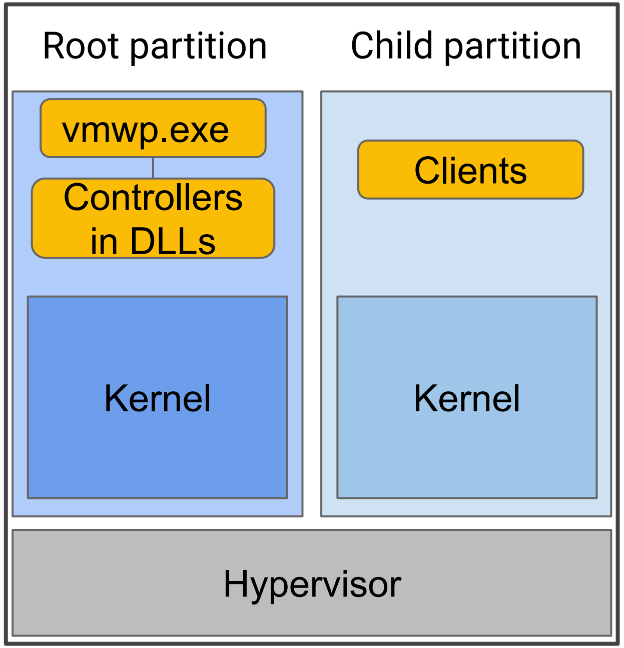
- And many more (a list can be found on GitHub/gerhart01/Hyper-V-Internals)

The Research Target



The **Emulated Devices Controllers**

- Examples: floppy disks, IDE, PS2
- Called “virtual devices” or “VDEVs” at Microsoft
- Emulation of hardware controller by the hypervisor
    - Real hardware controllers use and access control
    - Resources shared
    - Guest operating systems unmodified
- Implemented for Hyper-V generation 1 VMs
    - Azure mostly uses this generation
- Userland of the root partition
- In DLLs loaded by the worker process

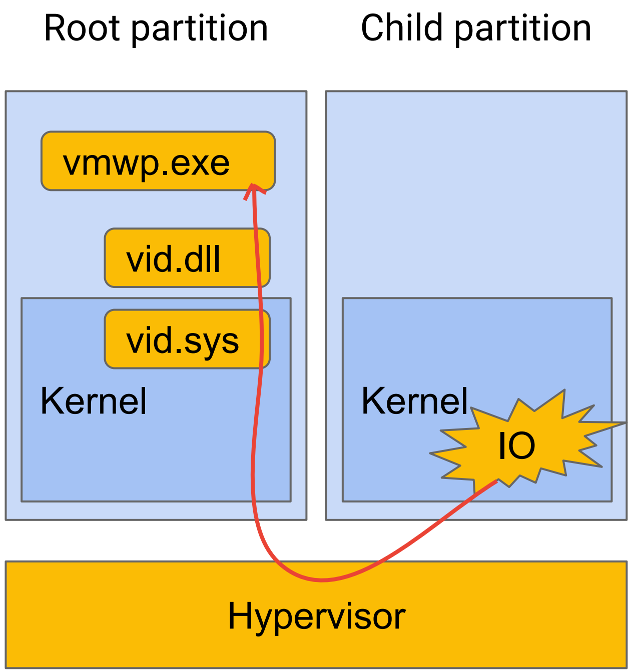


Why Choosing the Emulated Devices?

- Complex (state machines)
  - For example: enabling / disabling ports, updating a status register, waiting for a command
- Several bugs on several hypervisors
- Azure mostly uses Generation 1 VMs
- Hyper-V is developed in C++
- Potential “guest to root partition” escapes

Life of a Request

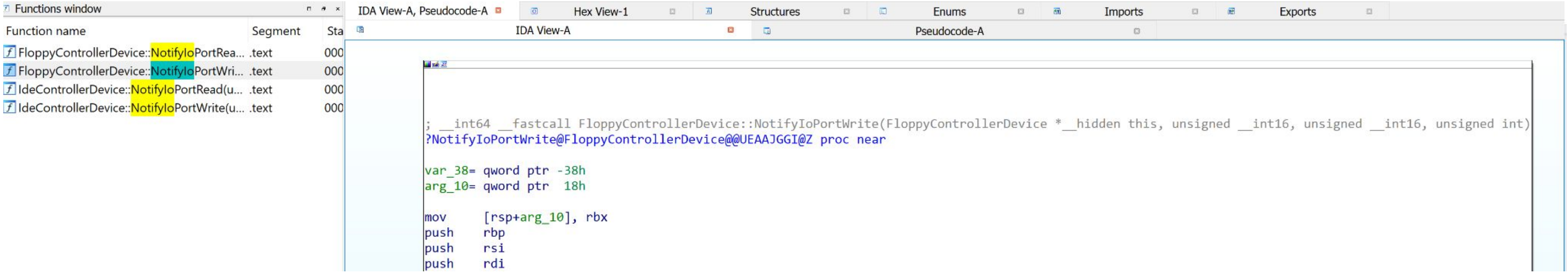
- Communication through IO ports
  - CPU instructions: IN / OUT
  - “IN EAX, DX”: input from I/O port in DX into EAX
  - “OUT DX, EAX”: output in EAX to I/O port address in DX
  - More details and versions in Intel manuals
- Communication through the hypervisor, the VID, and callbacks
- More on MSRC blogpost “Attacking the VM worker process”
- Some VDEVs are more complex with MMIO handling for instance, or the use of the VMBus



Google

Some Reverse Engineering

- DLL implementing the controllers
- Typical IO handlers
  - $Device::NotifyIOPortRead
  - $Device::NotifyIOPortWrite
- + : Symbols available, no particular difficulty (no obfuscation...)
- - : Reversing C++ and its indirect calls
- Example with VmEmulatedStorage.dll

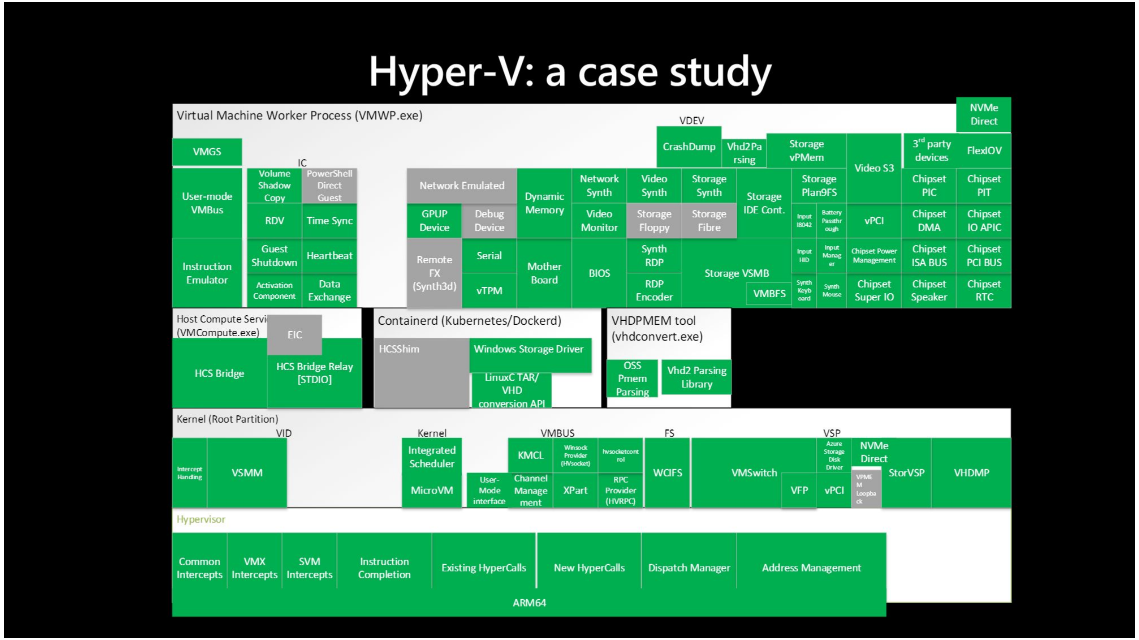


Hyntrospect: A Fuzzer for the Emulated Devices

https://github.com/googleprojectzero/Hyntrospect

Inspiration

- libFuzzer: coverage-guided approach
- Microsoft publication on their coverage (Keeping Windows Secure - Bluehat IL 2019)
- CVE-2018-0959 + Dedicated MSRC blogpost



How to do the same, closed source?

Google

Existing Tools for Windows Binaries Fuzzing

- Gathering Coverage
  - DynamoRIO
  - Intel Pin
  - Intel PT (though this is not a tool like previous two)
  - Mesos
  - QDBI for Windows
  - TinyInst
- Fuzzers
  - WinAFL + DynamoRIO
  - Jackalope
  - whvp
- Memory Corruption Detection
  - PageHeap

So Why Another Toolchain?

- The target is a DLL
  - This disqualifies all the fuzzers that only apply to executables
- Emulating only the relevant functions is hard
  - All the VM context would be needed
- vmwp binary and the DLL cannot be restarted with instrumentation
  - That would mean restarting the whole VM for each run
- The runtime operations are specific
  - Injecting / mocking the injection of IOs
- Some tools were developed during Hyntrospect development
- Managing all the blocks with a minimal set of languages is hard
- The fuzzer will be ported to similar use cases
  - vSMB, or with some architectural changes the network stack...

Scope

- Windows guest VM
- Intel CPU
- Generation 1VMs
- Binaries (DLLs/EXEs) in the userland of the root partition

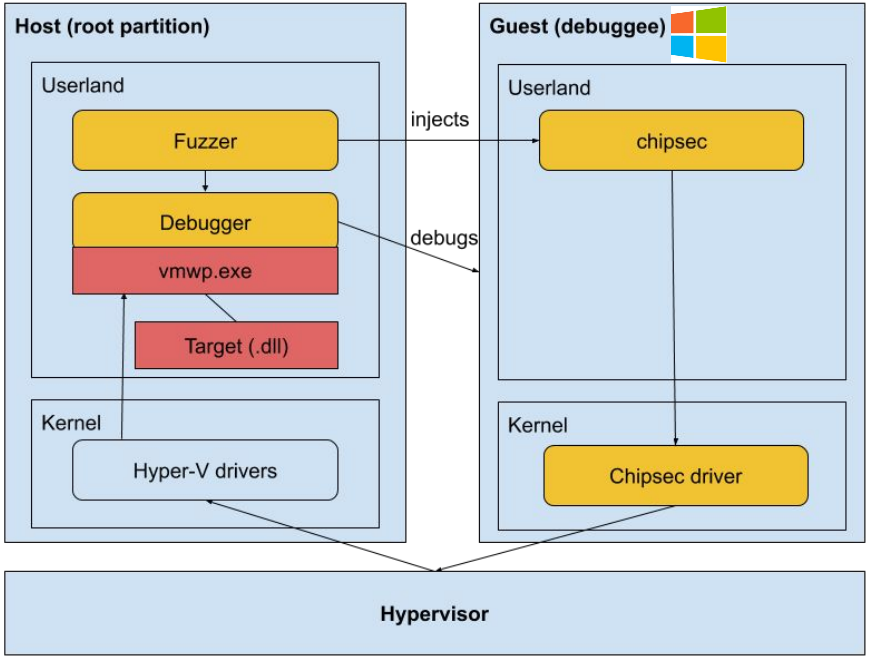
Design Choices at a Glance

Emulation vs execution	Execution of a VM through a debugger (DbgShell) at runtime
Coverage	Tracked with the int3 technique described by @5aelo for TrapFuzz / @gamozolabs mesos
Memory corruption detection	Pageheap (gflags)
Type of bugs	Memory corruption State machine logic errors [Use after free] ~~Race conditions~~

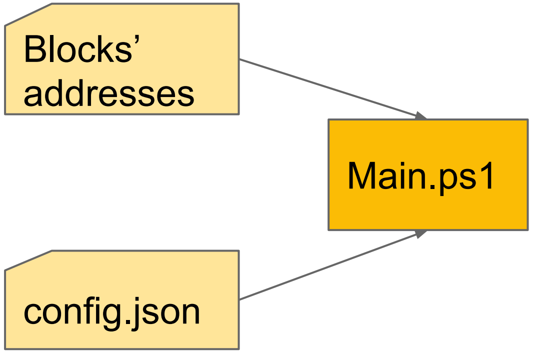
Design Choices at a Glance (2)

Environment reset	Hyper-V checkpoints = snapshots
Mutation strategy	Custom
Language	PowerShell (except for the IDA scripts)
External dependencies	DbgShell, CHIPSEC, [pageheap], [LightHouse], [IDA]

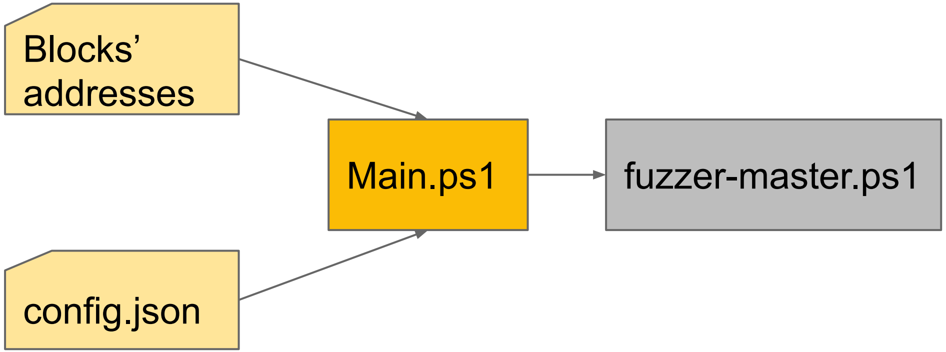
Overview of Hyntrospect



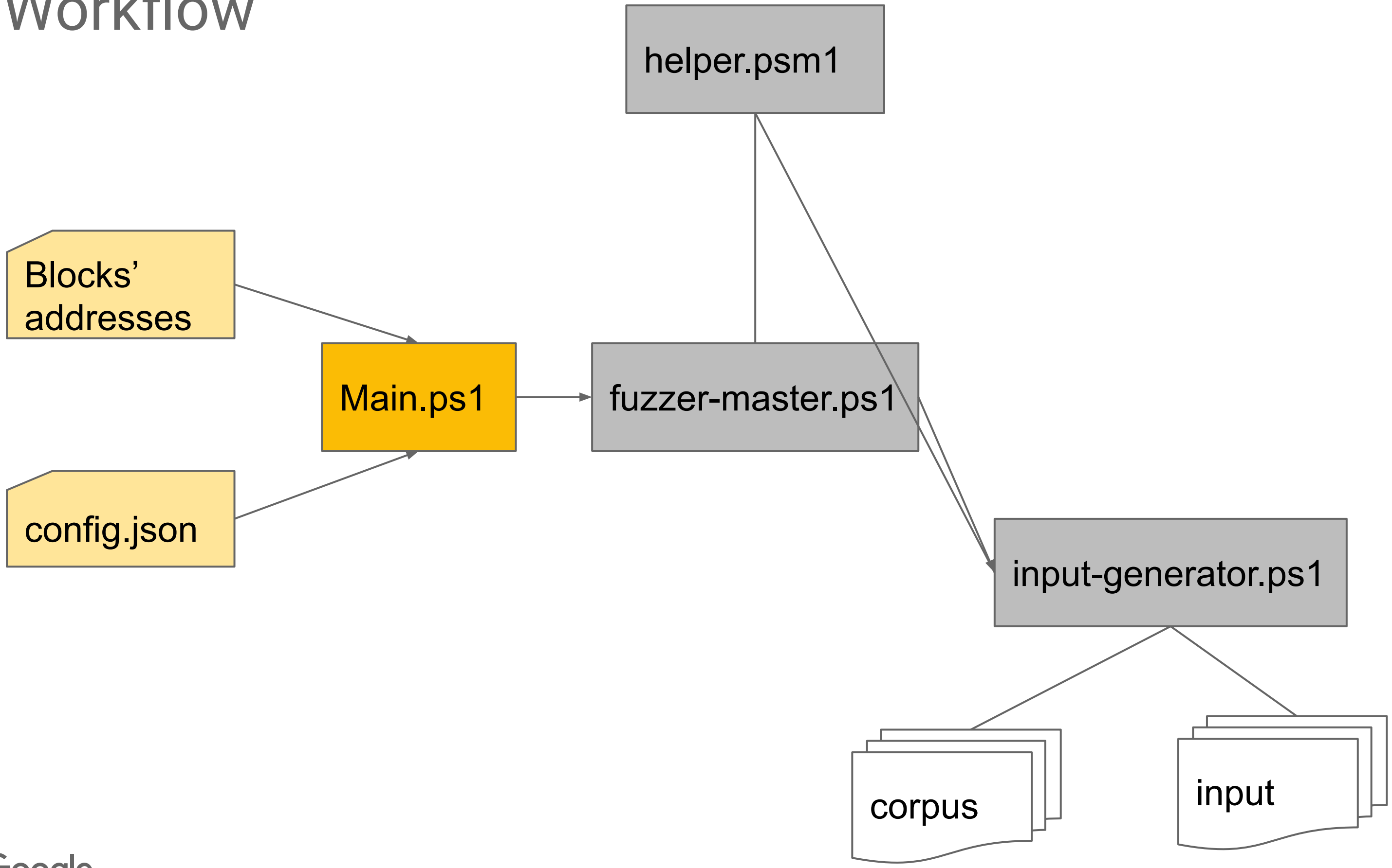
Workflow



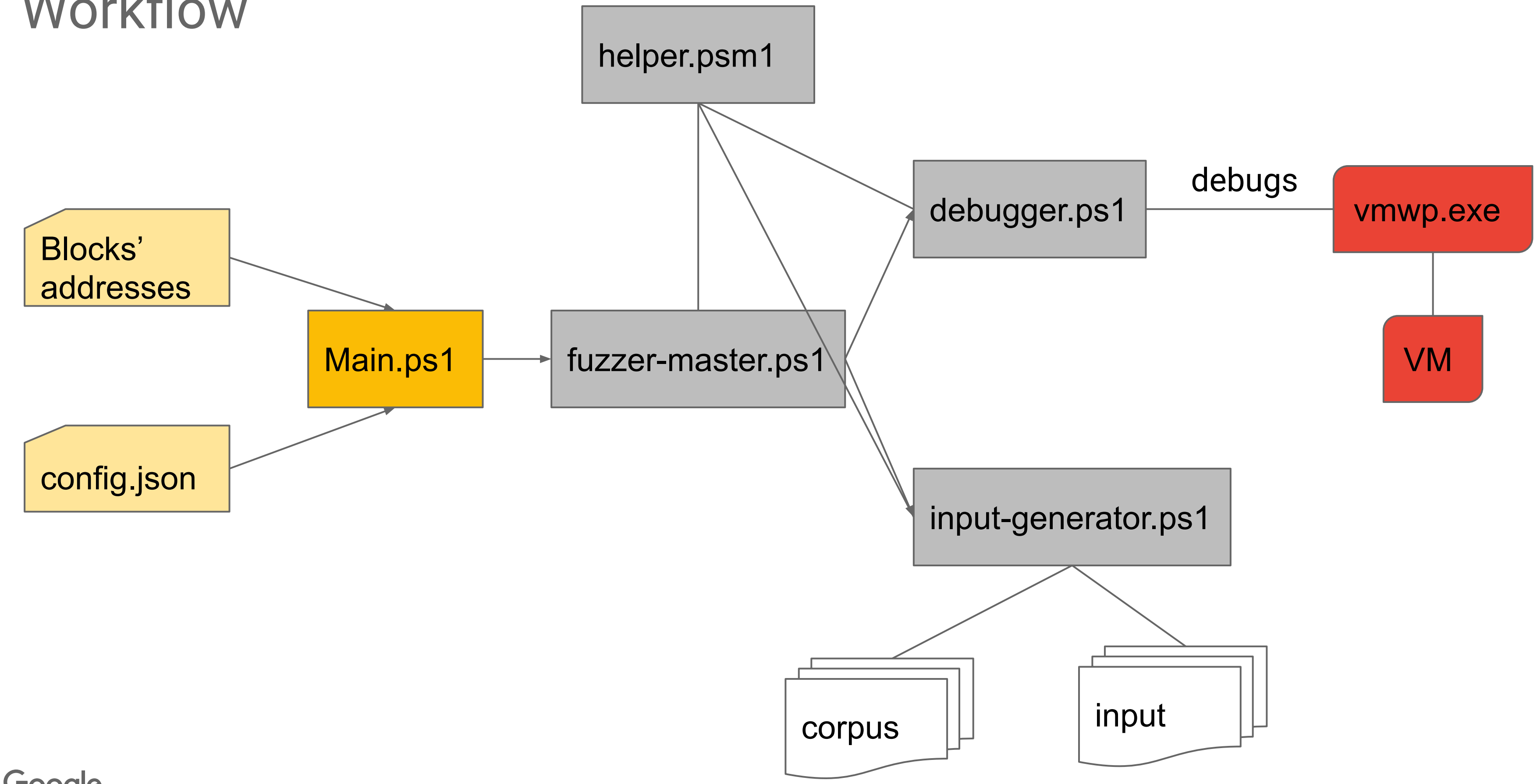
Workflow



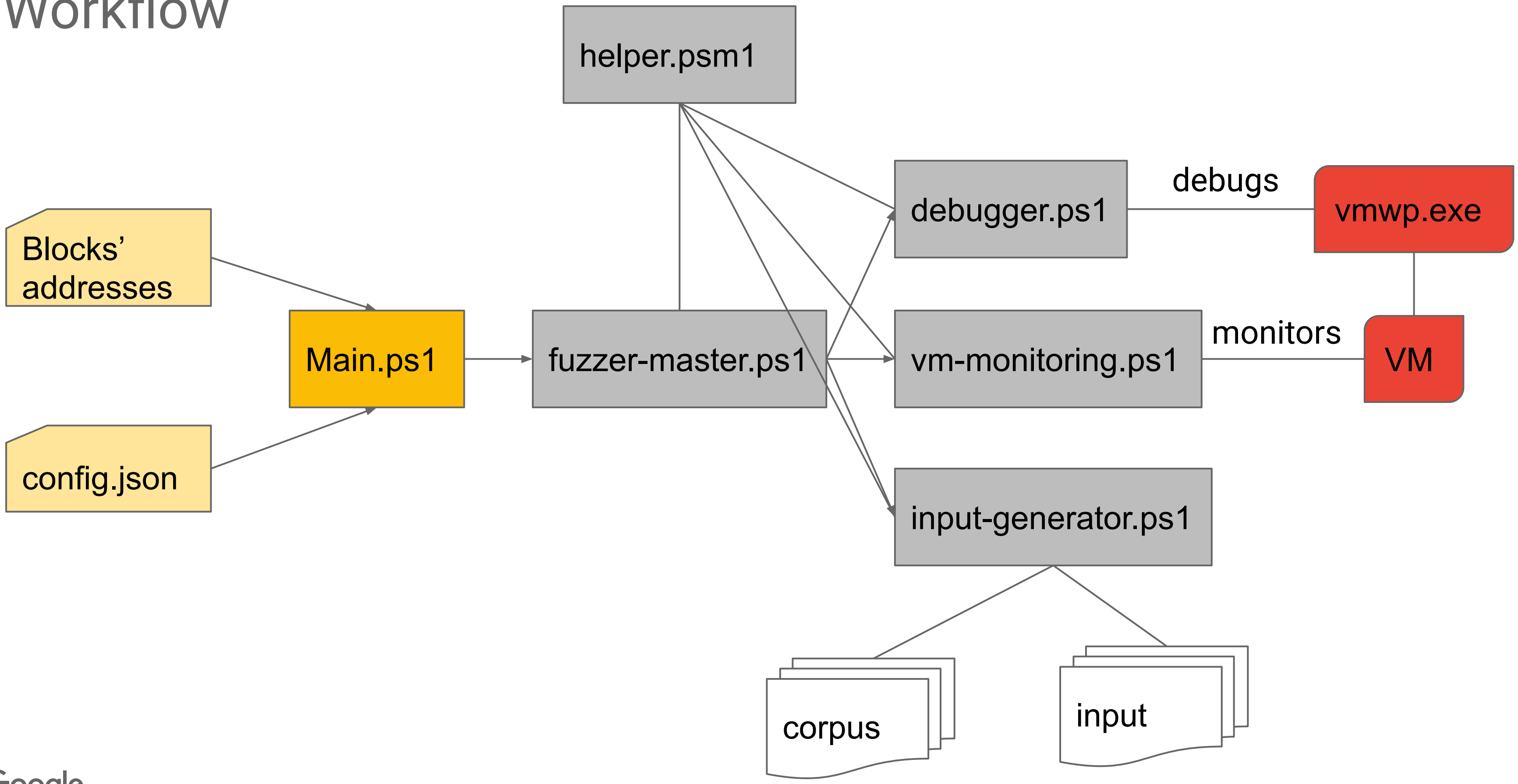
Workflow


helper.psm1
Blocks' addresses
Main.ps1
fuzzer-master.ps1
config.json
input-generator.ps1
corpus
input

Workflow

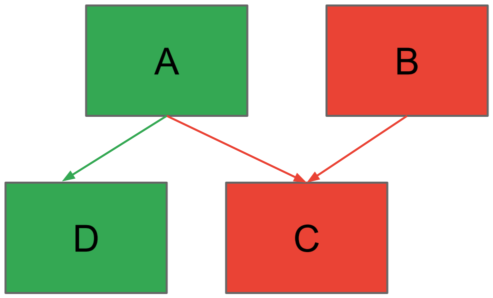

helper.psm1
debugs
debugger.ps1
vmwp.exe
Blocks'
addresses
Main.ps1
fuzzer-master.ps1
VM
config.json
input-generator.ps1
corpus
input

Workflow


helper.psm1
debugs
Blocks'
addresses
debugger.ps1
vmwp.exe
Main.ps1
fuzzer-master.ps1
vm-monitoring.ps1
monitors
VM
config.json
input-generator.ps1
corpus
input

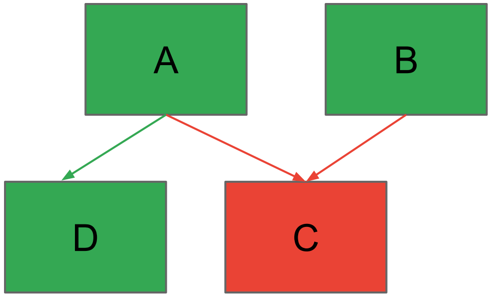
Coverage collection and guidance

- Block coverage



Coverage collection and guidance

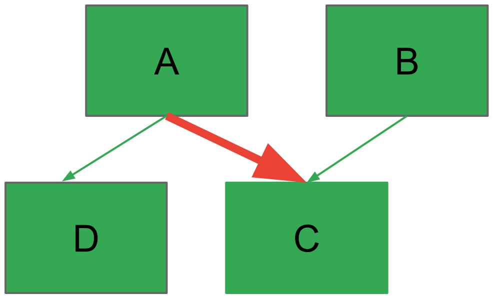
- Block coverage



Google

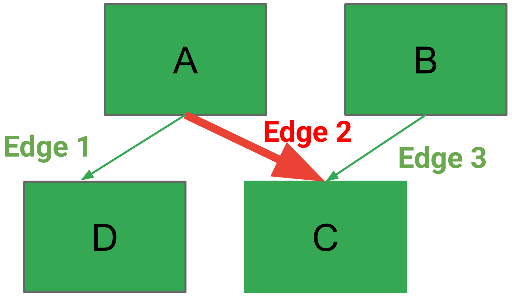
Coverage collection and guidance

- Block coverage



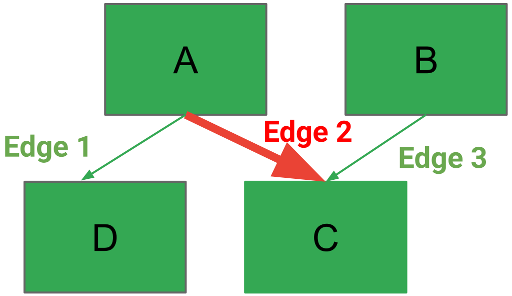
Coverage collection and guidance

- Block coverage
  - Versus edge coverage: easier to implement but does not promote rare paths



Coverage collection and guidance

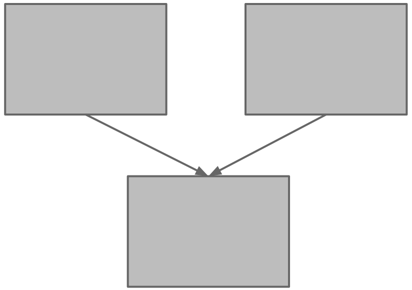
- Block coverage
  - Versus edge coverage: easier to implement but does not promote rare paths
  - No counter



Coverage collection and guidance

- Block coverage
  - Versus edge coverage: easier to implement but does not promote rare paths
  - No counter

- int3 technique
  - Pre-compute the list of targeted blocks' addresses
  - Set int3 at the beginning of each block
  - Each int3 reached = coverage increase
  - The int3 is removed, input file handled, execution resumes
  - Faster over time



Generation of the Input File

- Record of seeds at the beginning [optional]
  - Record of legitimate traffic
- Corpus of “interesting files”
  - Corpus files = permanent residents
  - Input files = temporary residents to be tested

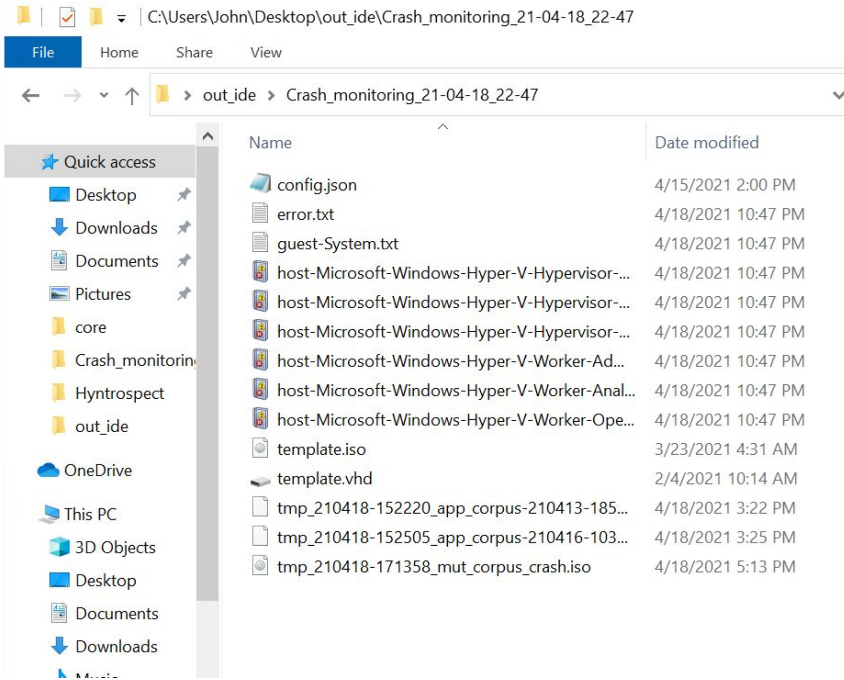
- Coverage increase -> truncated input file added to the corpus
  - Will influence future runs
- 3 strategies: mutate, append, generate randomly

- Format of input files
  - Byte 0 % 2 -> IN / OUT operation
  - Byte 1 % (number of ports) -> selected IO port
  - Byte 2 % 3 -> length
  - If OUT and based on length -> value

Crash Qualification

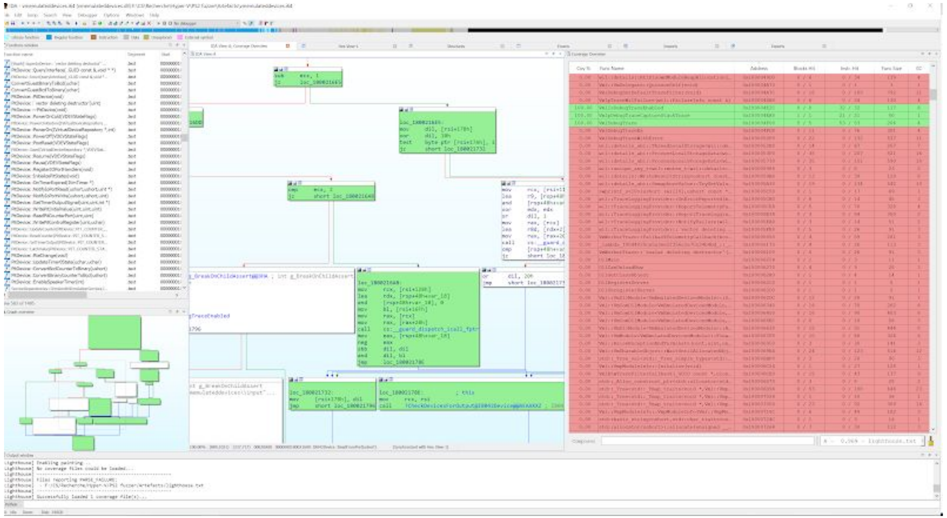
- 2 levels of monitoring:
  debugger level + monitoring process
  - Tip: the monitoring process can track the VM uptime
    -> avoid while(true)
    -> avoid missing quick status change
    (up-down-up again) … as if it was blinking

- Crash folder created with logs and artefacts to re-run the case

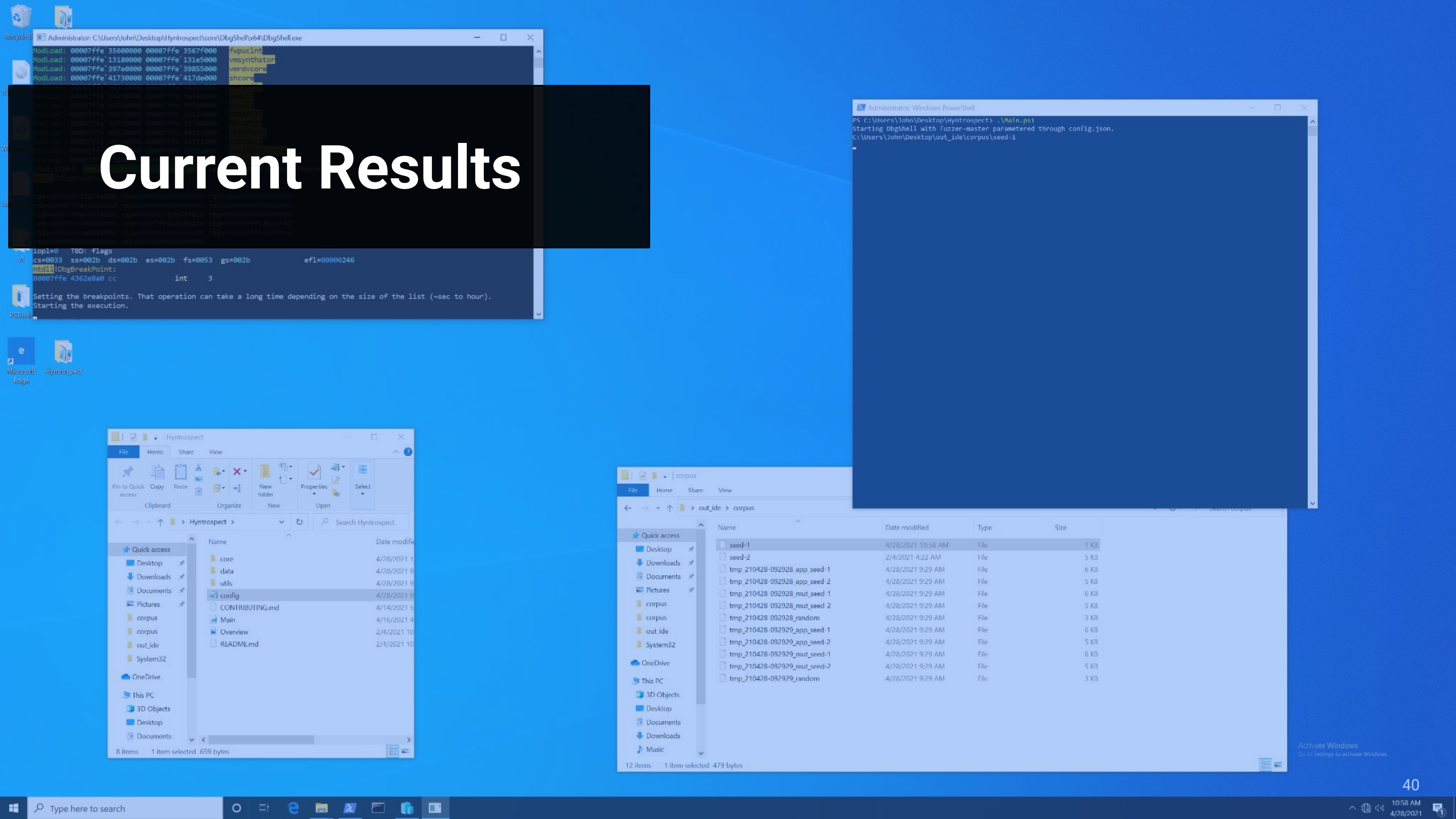


Coverage visualization in IDA

- Optionally, using a helper and IDA+LightHouse



Current Results



Local Runs

- First targets: i8042 (PS/2), videoS3, floppy, IDE
  - Example: I8042 device with IO ports 0x60, 0x61, 0x62, 0x64
- Local setup: dedicated workstation with 32 GB RAM and Intel Core i9 CPU
  - 8 GB per VM, 1 or 2 vCPUs

- Speed limitation
  - Main factor: number of breakpoints
  - Time to set them / update them in DbgShell
  - Not linear

Number of breakpoints	**Time to set up the breakpoints in DbgShell at each iteration**
150	immediate
500	6 seconds
1000	20 seconds
2000	1 minute 15 seconds

- Next goal: port the fuzzer to GCP

Coverage (3 days run)

vmemulateddevices.dll	Current coverage
VideoS3Device	42.7%
i8042Device	40%

VmEmulatedStorage.dll	Current coverage
FloppyControllerDevice	43.3%
IdeControllerDevice	28.8%

- Start / init / stop functions not called
    - Attaching to a running VM
- Debug strings blocks skipped

Guest VM Crash Found

- On i8042 device
- Reproducible
- BSOD of the VM with different error messages at each run
  - SYSTEM_SERVICE_EXCEPTION (0x3b)
  - PFN_LIST_CORRUPT (4e)
  - ATTEMPTED_WRITE_TO_READONLY_MEMORY (0xbe)
  - KERNEL_SECURITY_CHECK_FAILURE (0x139)
- Memory corruption error

Some More Investigation

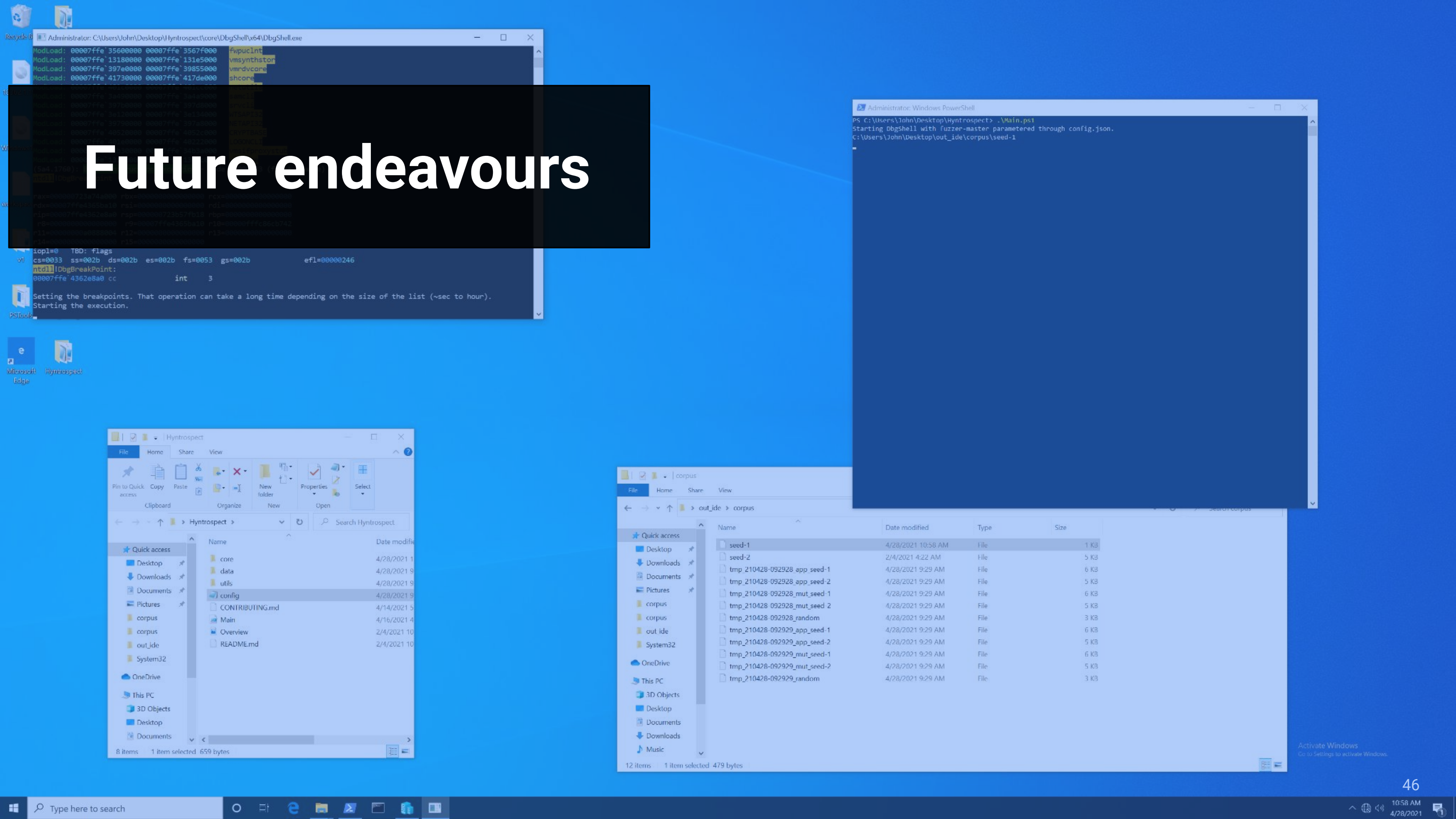
- Narrowed down the case
  - Sequence of 2 OUT operations
- State machine, path accessible in 2 steps
- PciBusDevice::HandleA20GateChange
  - Legacy A20 device
  - Updates the memory mapping on the host
  - … but the guest keeps the same mapping
- Question: possible compromission of VBS?

Follow-up

- In practice, impossible to exploit
- Not a security bug
- Shared with MSRC

- Validates the behavior of the fuzzer, crash handling and reproduction scripts

Future endeavours



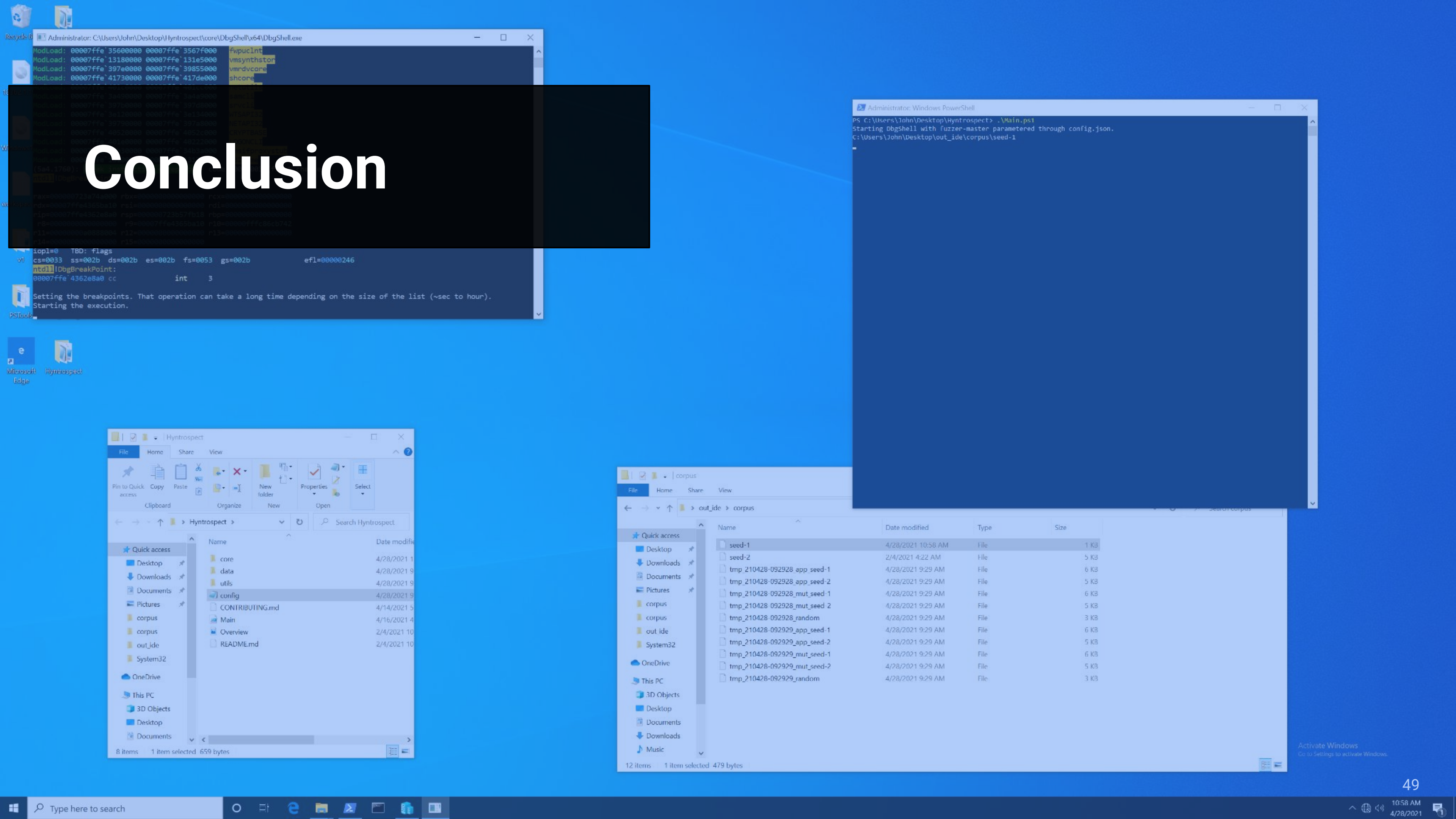
Design Limitations

- Restricted to the userland of the root partition
  - Limits the attack surface as parts of the virtualized stack are in the root partition kernel and hypervisor
- Not optimized for speed
  - More expensive in Cloud as more cycles are needed

Future Work

- Development of the fuzzer internals
  - Mutation strategy
  - Userland vs kernel
  - Speed-related updates: minimal debugger?
- Porting to GCP
  - Port to new devices
  - Run faster and longer
- Adapting to other root partition targets
  - Keeping the frame and “basic blocks”
  - Changing the commands and input consumption

Conclusion



Administrator: C:\Users\John\Desktop\Hyntrospect\core\DbgShell\x64\DbgShell.exe
ModLoad: 00007ffe`35600000 00007ffe`3567f000 fwpuclnt
ModLoad: 00007ffe`13180000 00007ffe`131e5000 vmsynthstor
ModLoad: 00007ffe`397e0000 00007ffe`39855000 vmrdvcore
ModLoad: 00007ffe`41730000 00007ffe`417de000 shcore
iopl=0 TBD: flags
cs=0033 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00000246
ntdll!DbgBreakPoint:
00007ffe`4362e8a0 cc int 3
Setting the breakpoints. That operation can take a long time depending on the size of the list (~sec to hour).
Starting the execution.
Administrator: Windows PowerShell
PS C:\Users\John\Desktop\Hyntrospect> .\Main.ps1
Starting DbgShell with fuzzer-master parameterized through config.json.
C:\Users\John\Desktop\out_ide\corpus\seed-1
Hyntrospect
core
data
utils
config
CONTRIBUTING.md
Main
Overview
README.md
8 items 1 item selected 659 bytes
corpus
seed-1
seed-2
tmp_210428-092928_app_seed-1
tmp_210428-092928_app_seed-2
tmp_210428-092928_mut_seed-1
tmp_210428-092928_mut_seed-2
tmp_210428-092928_random
tmp_210428-092929_app_seed-1
tmp_210428-092929_app_seed-2
tmp_210428-092929_mut_seed-1
tmp_210428-092929_mut_seed-2
tmp_210428-092929_random
12 items 1 item selected 479 bytes
Type here to search
10:58 AM
4/28/2021

https://github.com/googleprojectzero/Hyntrospect