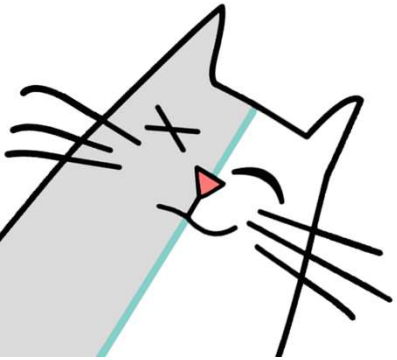
HyperDbg



ACM Conference on Computer and Communications Security (CCS)
November 2022

IPM
INSTITUTE FOR RESEARCH IN FUNDAMENTAL SCIENCES

조선대학교
CHOSUN UNIVERSITY

WPI

Stony Brook University

HyperDbg

CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY

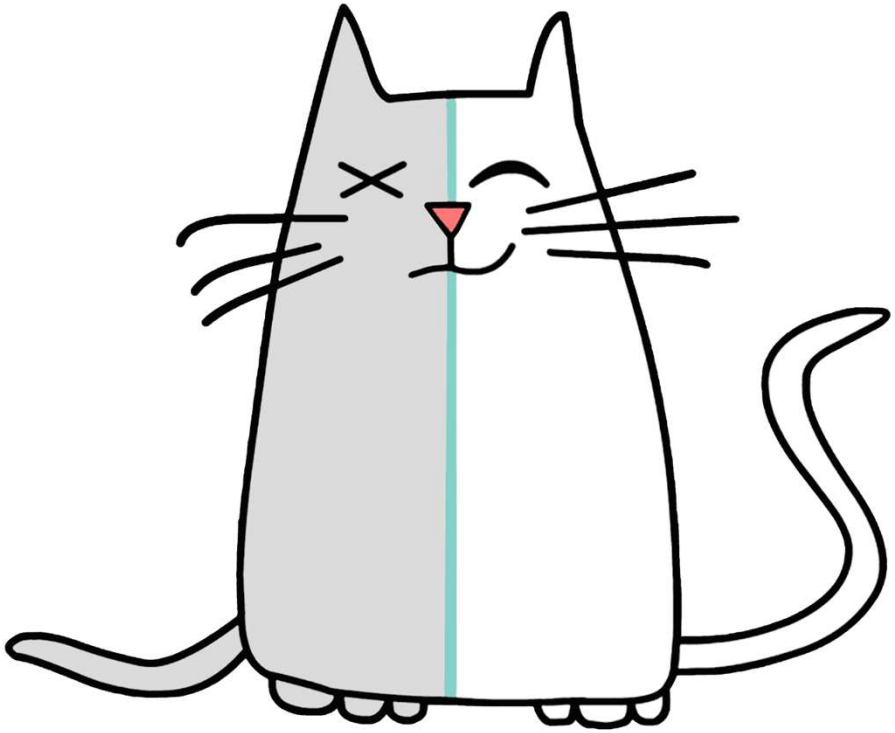
It’s a

Transparent

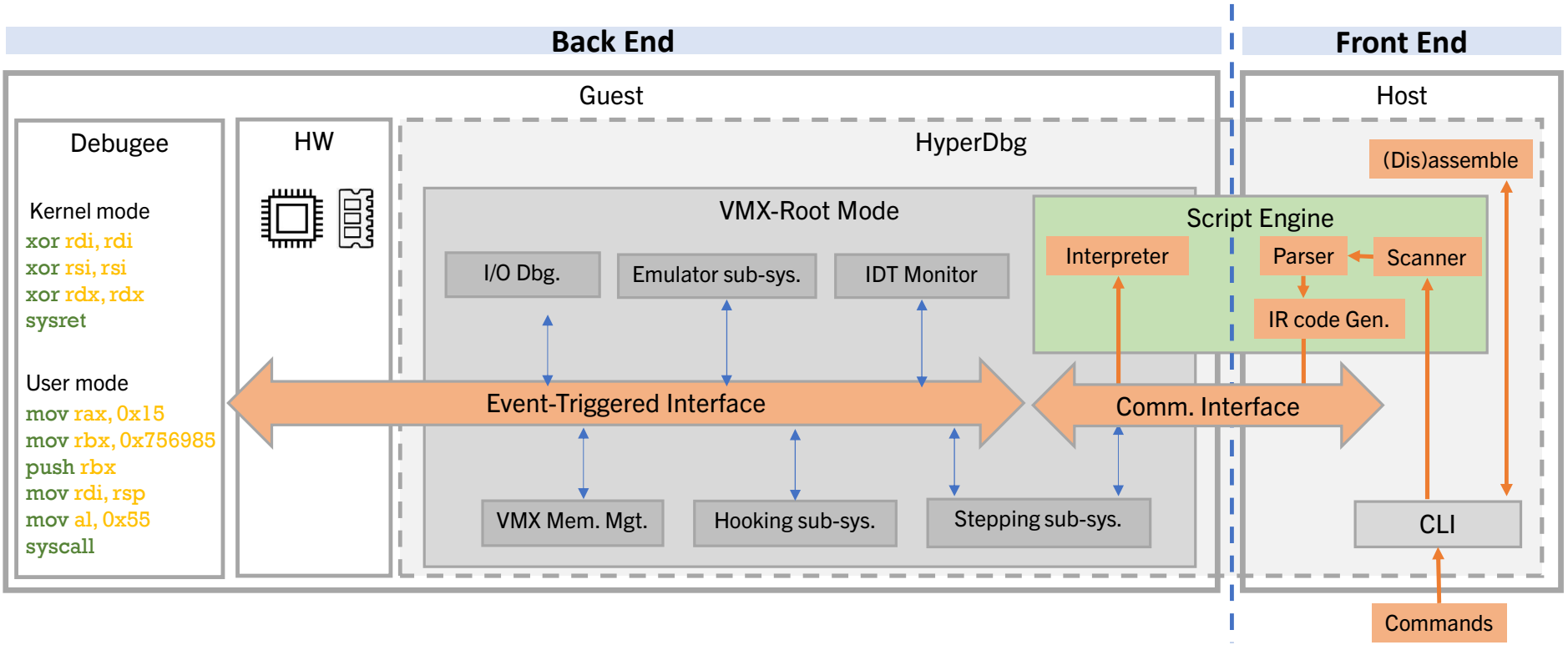
High-Performance

Hypervisor-Assisted

Debugger



	User Mode Debuggers	Kernel Mode Debuggers	Other sub-kernel debugging solutions	HyperDbg
Privilege	Ring > 0	Ring 0	Ring < 0	Ring -1
Transparency	Low	Medium	High	High
Performance	Medium	Medium	Low	High
Practicality	High	High	Low	High
	x64dbg OllyDbg Immunity Debugger	WinDbg GDB	Malt Ether	HyperDbg

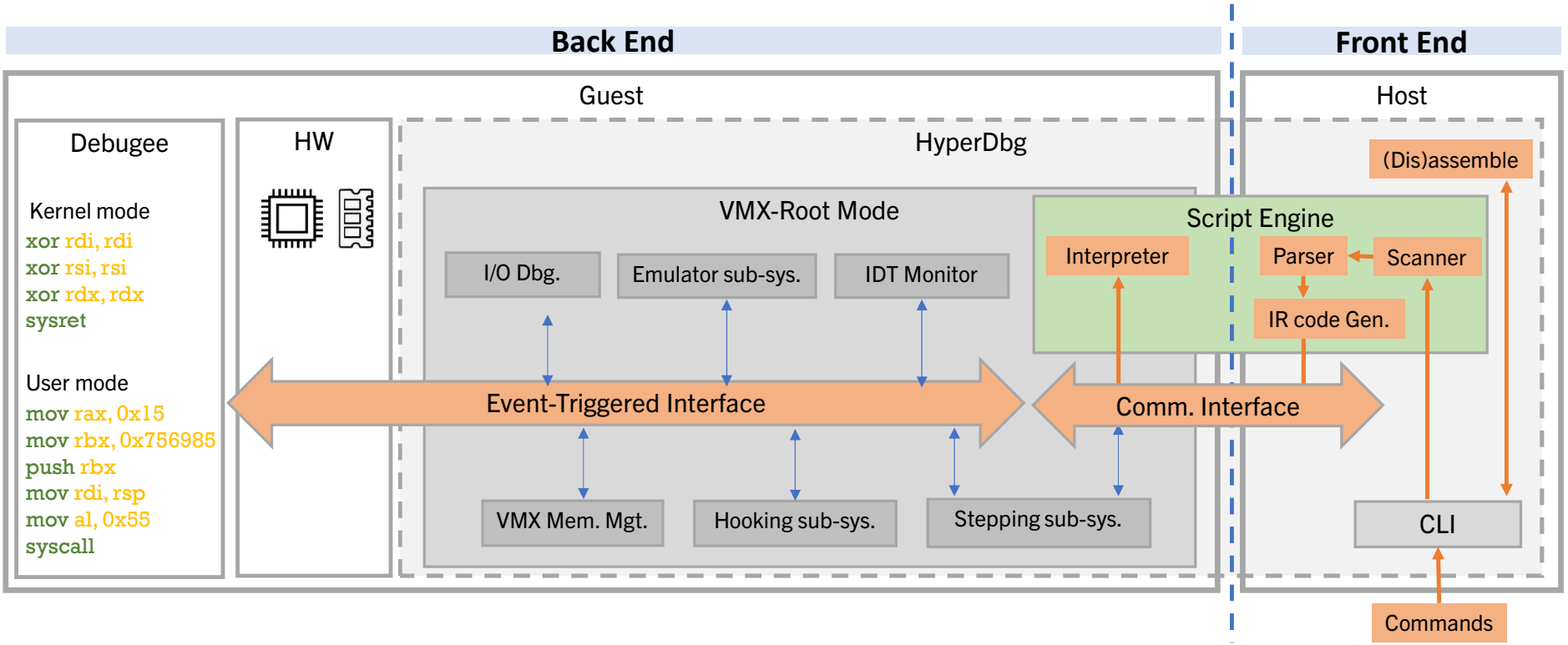


Event: An incident of interest
Syscall, specific memory access, . . .

Condition: Logical expressions constraining event triggers

Action: A set of functionalities following an event
Break, script (view/modify memory, registers), log,. . .

Script Engine on backend
C-like: if, else, for, printf



Performance
CPU Features (e.g., VT-x, EPT)
Script Engine
Efficient low-level implementation

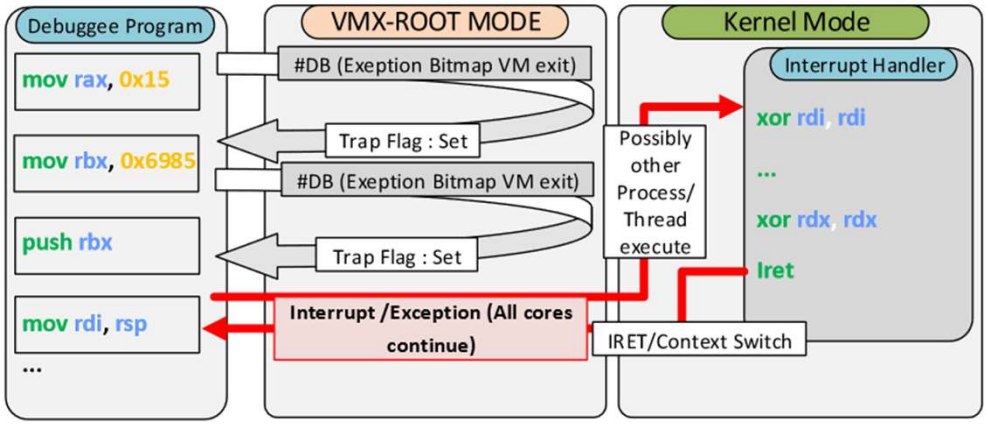
Privilege
VMX-Root Mode

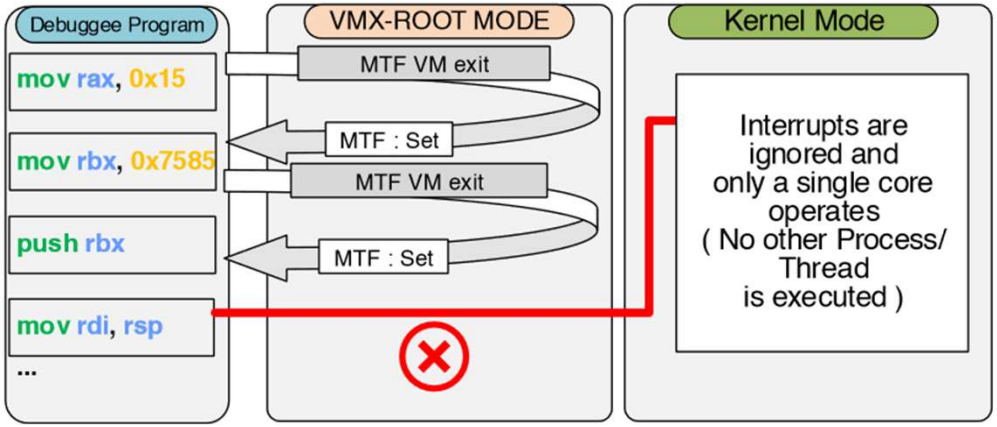
Transparency
Stealth in design

Stepping Subsystem

Guaranteed line by line instrumentation

VMX-Enable Single Core execution /Interrupt Masking





MTF+VM exit (and NMI/PIN Based VM EX Control)

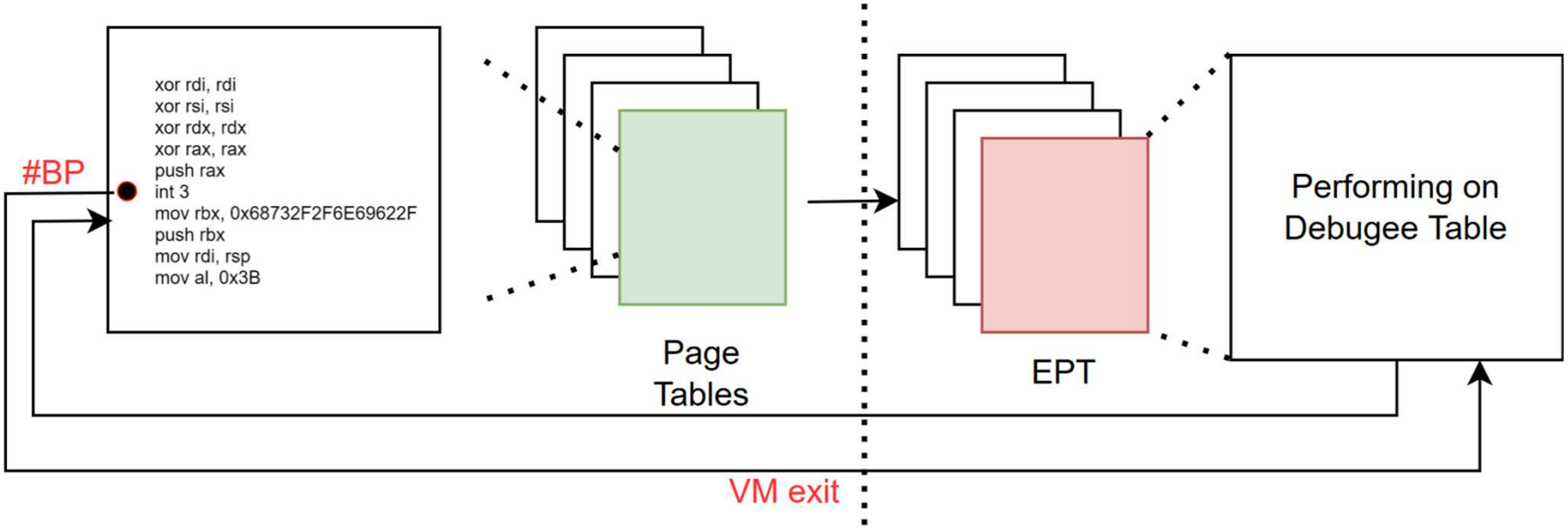
C:\Users\sina\Desktop\HyperDbg\HyperDbg\hyperdbg\build\bin\release\hyperdbg-cli.exe

```
HyperDbg Debugger [version: v0.2.0, build: 20221006]
Please visit https://docs.hyperdbg.org for more information...
HyperDbg is released under the GNU Public License v3 (GPLv3).

HyperDbg>
```

Hooking Subsystem

Transparent Hooking

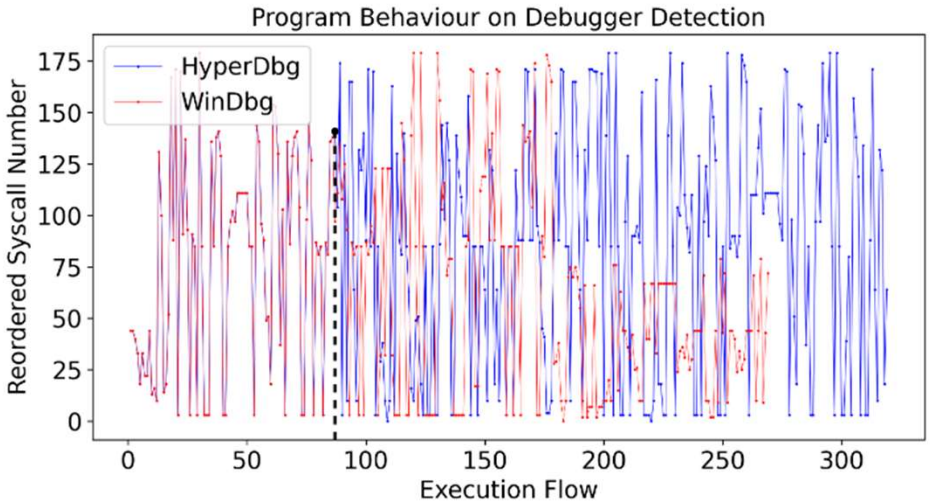
EPT-Hook and Debug Register Emulation



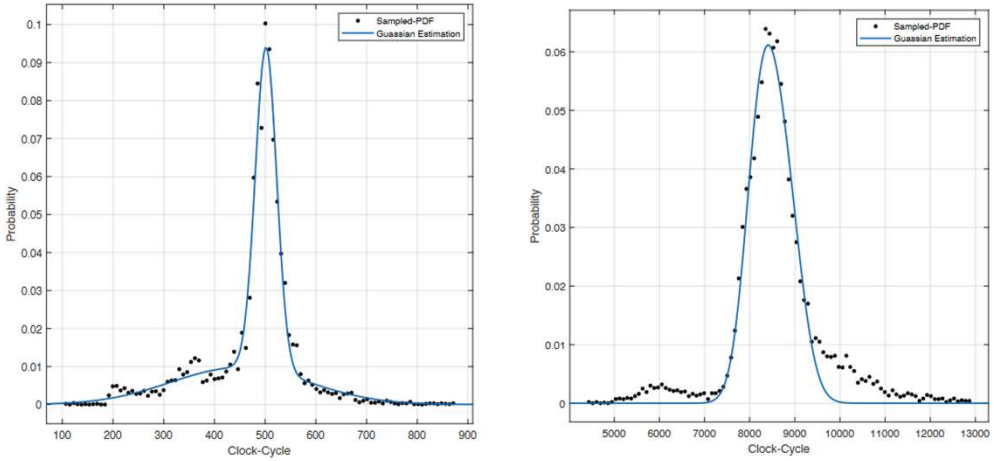
Transparency

Table 1: Anti-Debugging and Anti-VM exercises and mitigation in HyperDbg

Cat.	Methodology	Example of the Meth.	Example	Mitigation in HyperDbg
Anti-Debugging and Fingerprinting Methods	API-Call (System-Call)	GetCurrentProcessId() CreateToolhelp32Snap() Process32Next() NtQueryInfor.Proc.() FindWindow()	[9, 56]	Modify results via EPT-Hook (hiding process)
	PEB Field	IsDebuggerPresent() NtGlobalFlags()	[53]	HyperDbg is not detectable by default
	Heap Structure	HEAP.Flags HEAP.ForceFlags	[44]	HyperDbg is not detectable by default
	#BP Detection	Find BP (0xCC) inst. Read DR (Debug Register)	[57]	!dr to modify and disable unwanted BPs
	Timing Measurement	GetTickCount(), QueryPerf.Counter, GetLocalTime()	[58]	EPT-Hook Modification of results
	Trap-Interrupt	Instruction Prefix, INT 3, 0x2D, Interrupt 0x41	[52]	Set Exception bitmap in VMCS
	Control Flow Manupulation	NtSuspendThread(), NtSetInf.Thread(), CreateThread()	[54]	HyperDbg not detectable by default
Anti-VM/Hypervisor/Emulation	CPU Instructions	CPUID forces a VM-exit certain info in VM	[61]	VM-exit (CPUID result modification)
	Protection Model Instructions	SIDT, SLDT, SGDT STR, SMSW	[9]	VM-exit (emulation and modification)
	Architectural Delta-Timing	RDTSC+CPUID+RDTSC RDTSC(P)+RDTSC(P)	[61]	HyperDbg Trans. Mode (!hide command)
	In/Out Instructions	Magic I/O port in VMware	[24]	VM-exit handled (I/O bitmap)
	Invalid MSR Access	Invalid MSR issues General Protection (#GP)	[95]	Emulate !msrread/!msrwrite command
	Exception Handling	Try-Catch General Protection Excep. (#GP)	[55]	Handled by default Inject routine into user-mode



Timing Transparency



(a) (b)

Figure 5: PDF distribution of timing measurement for deactivated HyperDbg (a), with activated HyperDbg (b)

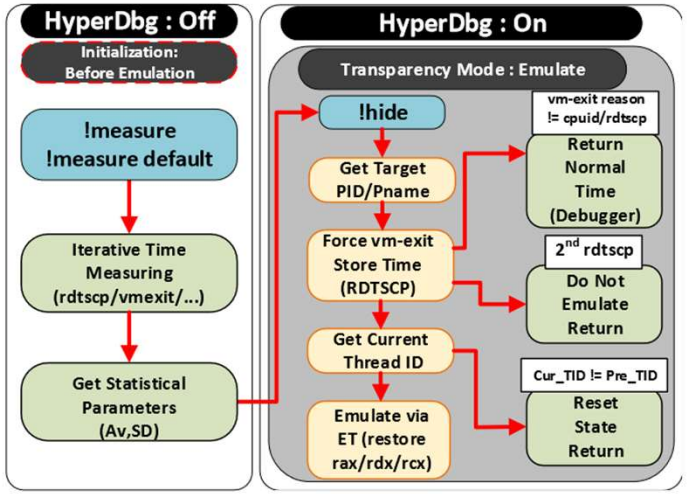
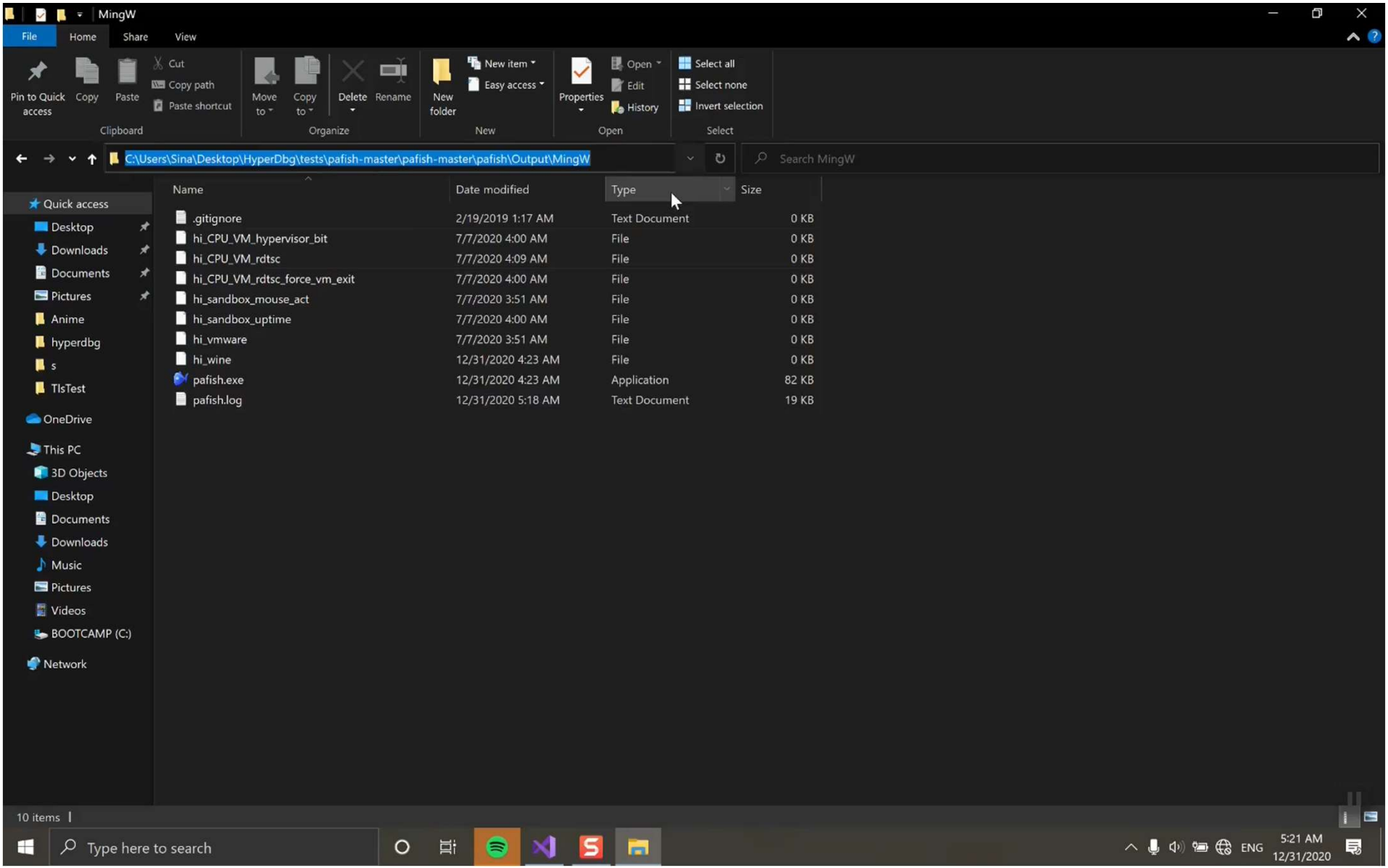


Figure 6: State Diagram Process of *rdtsc/rdtscp* emulation by HyperDbg

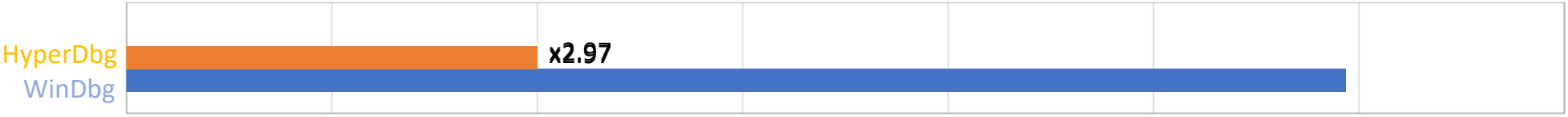
```
rdtscp    ; get the current time clock
cpuid     ;Execute a serialization instruction (VM-exit)
rdtscp    ; Delta Timing
```

Listing 2: The timing measurement code by forcing VM-exit


MingW
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New item
Easy access
New
Properties
Open
Edit
History
Select all
Select none
Invert selection
Select
C:\Users\Sina\Desktop\HyperDbg\tests\pafish-master\pafish-master\pafish\Output\MingW
Search MingW
Quick access
Desktop
Downloads
Documents
Pictures
Anime
hyperdbg
s
TlsTest
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
BOOTCAMP (C:)
Network
Name
Date modified
Type
Size
.gitignore 2/19/2019 1:17 AM Text Document 0 KB
hi_CPU_VM_hypervisor_bit 7/7/2020 4:00 AM File 0 KB
hi_CPU_VM_rdtsc 7/7/2020 4:09 AM File 0 KB
hi_CPU_VM_rdtsc_force_vm_exit 7/7/2020 4:00 AM File 0 KB
hi_sandbox_mouse_act 7/7/2020 3:51 AM File 0 KB
hi_sandbox_uptime 7/7/2020 4:00 AM File 0 KB
hi_vmware 7/7/2020 3:51 AM File 0 KB
hi_wine 12/31/2020 4:23 AM File 0 KB
pafish.exe 12/31/2020 4:23 AM Application 82 KB
pafish.log 12/31/2020 5:18 AM Text Document 19 KB
10 items
Type here to search
ENG
5:21 AM
12/31/2020

Performance

Single stepping time (lower is better)



Conditional Breakpoint executed (higher is better)

Syscall executed (higher is better)

*new 6 - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

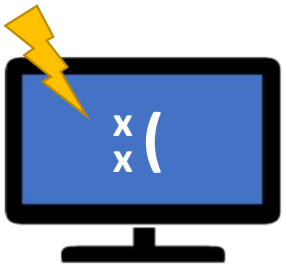
new 1 | new 2 | new 3 | new 4 | new 5 | new 6

```
WinDbg :

bp nt!NtOpenFile ".printf /D \"%mu \\n\", poi(poi(@r8+10)+8);g;"


-------------------------------------------------------------------

HyperDbg :


!epthook nt!NtOpenFile script {

    printf("%ws\n", dq(poi(r8 + 10) + 0x8));

}

-------------------------------------------------------------------
```

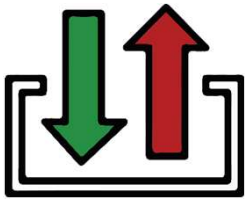
Normal text file | length : 332 lines : 18 | Ln : 18 Col : 72 Pos : 333 | Windows (CR LF) | UTF-8 | INS

68° ENG 4:09 PM 10/8/2022

Evasive Malware Analysis



Kernel Fuzzing



I/O Debugging

HyperDbg in the Wild

Over 125k+ lines of code
Over 2.5 years of development

HyperDbg / HyperDbg lines 128.7k Public

State-of-the-art native debugging tool

hyperdbg.org

GPL-3.0 license

1.9k stars 276 forks

Star Watch

Gerhart
@gerhart_x

Thanks for mention!
Btw, Hyper-V 0-day DoS vulnerability (CVE-2020-0890) was found, when i tested HyperDBG)

CVE-ID	
CVE-2020-0890	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

A denial of service vulnerability exists when Microsoft Hyper-V on a host server fails to properly validate specific malicious data from a user on a guest operating system.To exploit the vulnerability, an attacker who already has a privileged account on a guest operating system, running as a virtual machine, could run a specially crafted application.The security update addresses the vulnerability by resolving the conditions where Hyper-V would fail to handle these requests., aka 'Windows Hyper-V Denial of Service Vulnerability'. This CVE ID is unique from CVE-2020-0904.

Microsoft MSRC | Security Updates Acknowledgements {} Developer

MSRC > Customer Guidance > Security Update Guide > Vulnerabilities > CVE 2020 0890

Windows Hyper-V Denial of Service Vulnerability

CVE-2020-0890

On this page

Security Vulnerability

Released: Sep 8, 2020

Assigning CNA: Microsoft

CVE-2020-0890

HyperDbg Documentation

HyperDbg | Research | Download | Source code | Search... ^K

HyperDbg

GETTING STARTED

- Quick Start
- FAQ
- Build & Install
- Attach to HyperDbg

USING HYPERDBG

- Prerequisites
- User-mode Debugging
- Kernel-mode Debugging

Powered By GitBook

HyperDbg

A hypervisor-assisted debugger designed for analyzing, fuzzing and reversing

What is it?

HyperDbg debugger is an open-source, user-mode, and kernel-mode debugger focusing on using hardware technologies to provide new features to the debuggers' world.

HYPERDBG

A debugger designed for analyzing, fuzzing and reversing

https://docs.hyperdbg.org/

HyperDbg Debugger

Main Page | Related Pages | Namespaces | Classes | Files | Search

- HyperDbg Debugger
  - HyperDbg Debugger
  - Changelog
  - Contributor Covenant Code of Conduct
  - Contribution
  - HOWTO
  - Namespaces
  - Classes
  - Files

HyperDbg Debugger Documentation

Link Website | Link Docs | Link Doxygen | License GPLv3 | Published Papers | HyperDbg Twitter

HyperDbg Debugger

HyperDbg Debugger is an open-source, community-driven, hypervisor-assisted, user-mode and kernel-mode Windows debugger with a focus on using modern hardware technologies. It is a debugger designed for analyzing, fuzzing and reversing.

Follow **HyperDbg** on Twitter to get notified about new releases.

Description

HyperDbg is designed with a focus on using modern hardware technologies to provide new features to the debuggers' world. It operates on top of Windows by virtualizing an already running system using Intel VT-x and Intel PT. This debugger aims not to use any APIs and software debugging mechanisms, but instead, it uses Second Layer Page Table (a.k.a. Extended Page Table or EPT) extensively to monitor both kernel and user executions.

https://doxygen.hyperdbg.org/

HyperDbg Debugger

A debugger designed for analyzing, fuzzing and reversing

1st Edition

Kernel Debugger Design in HyperDbg

Website https://hyperdbg.org

Research https://research.hyperdbg.org

July 4, 2022

this issue by presenting a guaranteed stepping method. As explained, MTF is a feature that works similar to *RFLAGS* but is transparent to the guest. The following listing illustrates the set/unset of MTF in an execution sequence.

```
/* Set the monitor trap flag */
void HvSetMonitorTrapFlag(BOOLEAN Set)
{
 unsigned long CpuBasedVmExecControls = 0;
 // Read the previous flag
 __vmx_vmread(CPU_BASED_VM_EXEC_CONTROL, &CpuBasedVmExecControls);
 if (Set) {
 CpuBasedVmExecControls |= CPU_BASED_MONITOR_TRAP_FLAG;
 }
 else {
 CpuBasedVmExecControls &= ~CPU_BASED_MONITOR_TRAP_FLAG;
 }
 // Set the new value
 __vmx_vmwrite(CPU_BASED_VM_EXEC_CONTROL, CpuBasedVmExecControls);
}
```

Listing 1: MTF Set/Unset in an example execution sequence

By executing each instruction, it is ensured that the specific line of code is passed to the CPU. In order to get the execution after executing an instruction, a *VM-exit* is triggered by setting an MTF which guarantees that only one succeeding instruction will be executed in the guest. In order to do so, HyperDbg continues at only one core and disables interrupts in the same core (ignoring external-interrupts by setting external-interrupts exiting bit in VMCS) to offer a fine-grained stepping. Figure 4 depicts the general procedure in

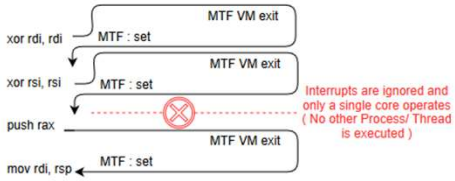


Figure 4: The *i command* Instrumentation Stepping Approach in HyperDbg

https://research.hyperdbg.org/

HyperDbg Debugger

A debugger designed for analyzing, fuzzing and reversing

version 0.1.0.0

VM-exit Transparency in HyperDbg

Website https://hyperdbg.org

Research https://research.hyperdbg.org

July 25, 2022

3.1 Changing Timestamps

In this approach, by the use of the statistical methodology described in the previous section, an accurate timing profiles are constructed and employed in transparency functions. An overview of the approach is shown in Figure 3.

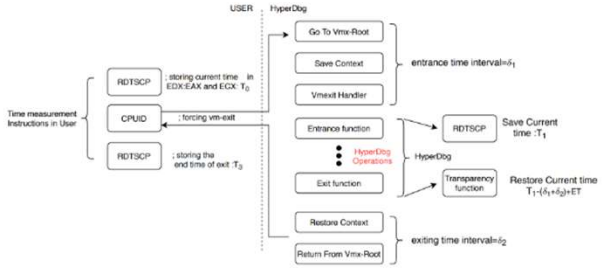


Figure 3: Transparency of HyperDbg by changing IA32_ TIME_ STAMP_COUNTER

Assuming the target time measuring instruction set in Listing 1, according to the Figure 3, in the presence of HyperDbg, CPUID, will force out a *VM-exit*. The *VM-exit*, consist of a *entrance* and *exiting* phases as depicted in the figure. *entrance* and *exiting* time intervals represented by δ_1 and δ_2 respectively, are measured iteratively in order to derive statistical characteristics. These timing are also follow a Gaussian Distribution (or two terms Gaussian Dis.). Note that the values of δ_1 and δ_2 are measured initially before the hiding process itself. To ensure the validity of these measurement for the HyperDbg, a special function in the user-mode is executed by the debugger, storing the timing stamps (T_0) using *Volatile* registers (e.g. EDX, ECX). Moreover, additional hash validation is checked for each time measurement.

After the profiling phase, the hiding process is executed by HyperDbg as shown in the Figure 3. When an analyzer software is activated to detect any low-level interception, the Entrance Function stores the time by executing RDTSCP (T_1), after *VM-exit* entrance. Then, any arbitrary functions in HyperDbg are executed. At the end of the operations in the hypervisor, a Transparency function is called. Here, the IA32_ TIME_ STAMP_COUNTER value is replaced with the following value.

$$Time\_Stamp = T_1 - (GRG(\delta_1) + GRG(\delta_2)) - GRG(Norm) \quad (2)$$

The GRG is the Gaussian Random Generated number by the use of the Marsaglia Polar Method [5] activated with Standard Deviation (σ) and Mean (μ) values captured in the initial statistical test-cases. GRG(Norm) represents the Gaussian estimated elapsed clock-cycles for the timing instructions when HyperDbg is not activated in normal condition.

docs.hyperdbg.org

HyperDbg Documentation

HyperDbg Research Download Source code

Powered By GitBook

Copy link

Edit on GitHub

HyperDbg

A hypervisor-assisted debugger designed for analyzing, fuzzing and reversing

What is it?

HyperDbg debugger is an open-source, user-mode, and kernel-mode debugger focusing on using hardware technologies to provide new features to the debuggers' world.

Thank you for your attention
Questions?

Table 5: A comprehensive comparison of HyperDbg with different debuggers

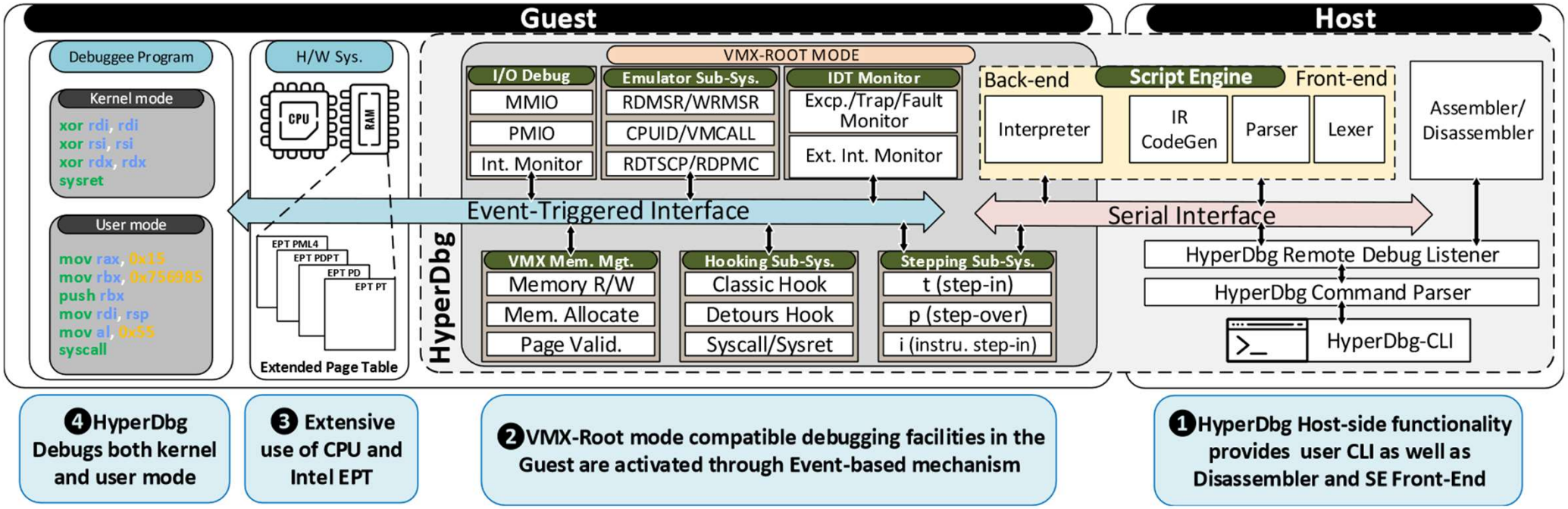
Debugger	Deployment Level	Debugging Mode		Transparency Insurance	Direct Deployment (NO VM/Emulator)	Source Code Available	Hooking	Scripting	Custom Assembly Execution	Notable Features (Currently Working Debuggers)
		User Mode	Kernel Mode							
HyperDbg	Hypervisor (Ring -1)	✓	✓	Hardware Assisted Methods (e.g. TimeStamp Emulation Transparency)	✓	✓	EPT Hidden Hooking	Customized VMX compatible ScriptEngine	✓	Different subsystems Fast script engine I/O debugging
Ghidra[23]	Application (Ring 3)	✓	✗	✗	✗	✓	✗	Python Scripting	✗	Advanced Decompiler Multi-platform Multi-architecture support
WinDbg[19]	Operating System	✓	✓	✗	✓	✗	✗	JavaScript WinDbg Script	✗	Javascript support Advanced scripting language
SoftIce[20]	Operating System	✗	✓	✗	✓	N/A	✗	✗	✗	Local debugging with special GUI
x64dbg[98]	Application (Ring 3)	✓	✗	Software-Based ScyllaHide	✓	✓	✗	Customized Scripting	✗	Flexible and strong GUI Lots of useful features
Ollydbg[76]	Application (Ring 3)	✓	✗	Software-Based ScyllaHide	✓	✗	✗	ODBGScript	✓	Stability
gdb [22]	Operating System	✓	✓	✗	✓	✓	✗	Bash Scripting Automation	✗	Main Linux Debugger Support multiple platforms
Malt[102]	SMM (Ring -2)	N/A	✓	SMM Level Transparency	✓	N/A	N/A	N/A	N/A	N/A
BareBox[50]	Hypervisor (Ring -1)	N/A	✓	Meta OS (Bare Metal) Transparency	✓	N/A	N/A	N/A	N/A	N/A
V2E[100]	Hypervisor (Ring -1)	N/A	✓	Hypervisor Level Transparency	✗	N/A	N/A	N/A	N/A	N/A
Anubis[64]	Hypervisor (Ring -1)	N/A	✓	Limited Software-Based Methods	✗	N/A	N/A	N/A	N/A	N/A
Virt-ICE[82]	Hypervisor (Ring -1)	N/A	✓	Emulating Software-Based Methods	✗	N/A	N/A	N/A	N/A	N/A
HyperDbg (old) (deprecated)	Hypervisor (Ring -1)	✓	✓	N/A	✗	✓	✗	✗	✗	N/A
Ether[27]	Hypervisor (Ring -1)	✓	✓	Hypervisor Level Transparency	✗	N/A	N/A	N/A	N/A	N/A
VAMPiRE[91]	Hypervisor (Ring -1)	N/A	✓	Hypervisor Level Transparency	✓	N/A	N/A	N/A	N/A	N/A
SPIDER[26]	Hypervisor (Ring -1)	N/A	✓	Hypervisor Level Transparency	✗	N/A	N/A	N/A	N/A	N/A
IDAPro[41]	Application (Ring 3)	✓	✗	✗	✓	✗	✗	Built-in Scripting Engine (IDC / Python)	✗	Advanced decompiler Multi-architecture Multi-platform support

Table 1: Anti-Debugging and Anti-VM exercises and mitigation in HyperDbg

Cat.	Methodology	Example of the Meth.	Example	Mitigation in HyperDbg
Anti-Debugging and Fingerprinting Methods	API-Call (System-Call)	GetCurrentProcessId() CreateToolhelp32Snap() Process32Next() NtQueryInfor.Proc.() FindWindow()	[9, 60]	Modify results via EPT-Hook (hiding process)
	PEB Field	IsDebuggerPresent() NtGlobalFlags()	[57]	HyperDbg is not detectable by default
	Heap Structure	HEAP.Flags HEAP.ForceFlags	[48]	HyperDbg is not detectable by default
	#BP Detection	Find BP (0xCC) inst. Read DR (Debug Register)	[61]	!dr to modify and disable unwanted BPs
	Timing Measurement	GetTickCount(), QueryPerf.Counter, GetLocalTime()	[62]	EPT-Hook Modification of results
	Trap-Interrupt	Instruction Prefix, INT 3, 0x2D, Interrupt 0x41	[56]	Set Exception bitmap in VMCS
	Control Flow Manupulation	NtSuspendThread(), NtSetInf.Thread(), CreateThread()	[58]	HyperDbg not detectable by default
Anti-VM/Hypervisor/Emulation	CPU Instructions	CPUID forces a VM-exit certain info in VM	[66]	VM-exit (CPUID result modification)
	Protection Model Instructions	SIDT, SLDT, SGDT STR, SMSW	[9]	VM-exit (emulation and modification)
	Architectural Delta-Timing	RDTSC+CPUID+RDTSC RDTSC(P)+RDTSC(P)	[66]	HyperDbg Trans. Mode (!hide command)
	In/Out Instructions	Magic I/O port in VMware	[25]	VM-exit handled (I/O bitmap)
	Invalid MSR Access	Invalid MSR issues General Protection (#GP)	[103]	Emulate !msrread/!msrwrite command
	Exception Handling	Try-Catch General Protection Excep. (#GP)	[59]	Handled by default Inject routine into user-mode

Table 2: Evaluation and comparison of HyperDbg for integrated software via packers/protectors

Packer/Protector	File Type	WinDbg	x64dbg	Ollydbg	**HyperDbg**	**HyperDbg's Trans. Mode**
ASPack.V2.42	PE32	Error	✓	✓	✓	✓
Enigma.V4.30	PE64	✗	✗	N/A	✗	✓
Net_Crypto.V5	PE32	✗	✗	✗	✓	✓
Obsidium.V1.7	PE64	✗	✗	N/A	✓	✓
Themida.V3.0.4	PE64	✗	✗	N/A	✓	✓
Upx.V3.96	PE64	✓	✓	✓	✓	✓
Vmprotect.V.2.13	PE64	✗	✗	N/A	✗	✓
MEW11.V1.2	PE32	✓	✓	✓	✓	✓
Pecompact.V3.11	PE32	✓	✓	✓	✓	✓
PELock.V2.0	PE32	✗	✗	✗	✓	✓
Petite.V2.4	PE32	Error	✓	✓	✓	✓
TeLock.V0.98	PE32	✓	✓	✓	✓	✓
YodaCrypter.V1.02	PE32	✗	✗	✗	✓	✓



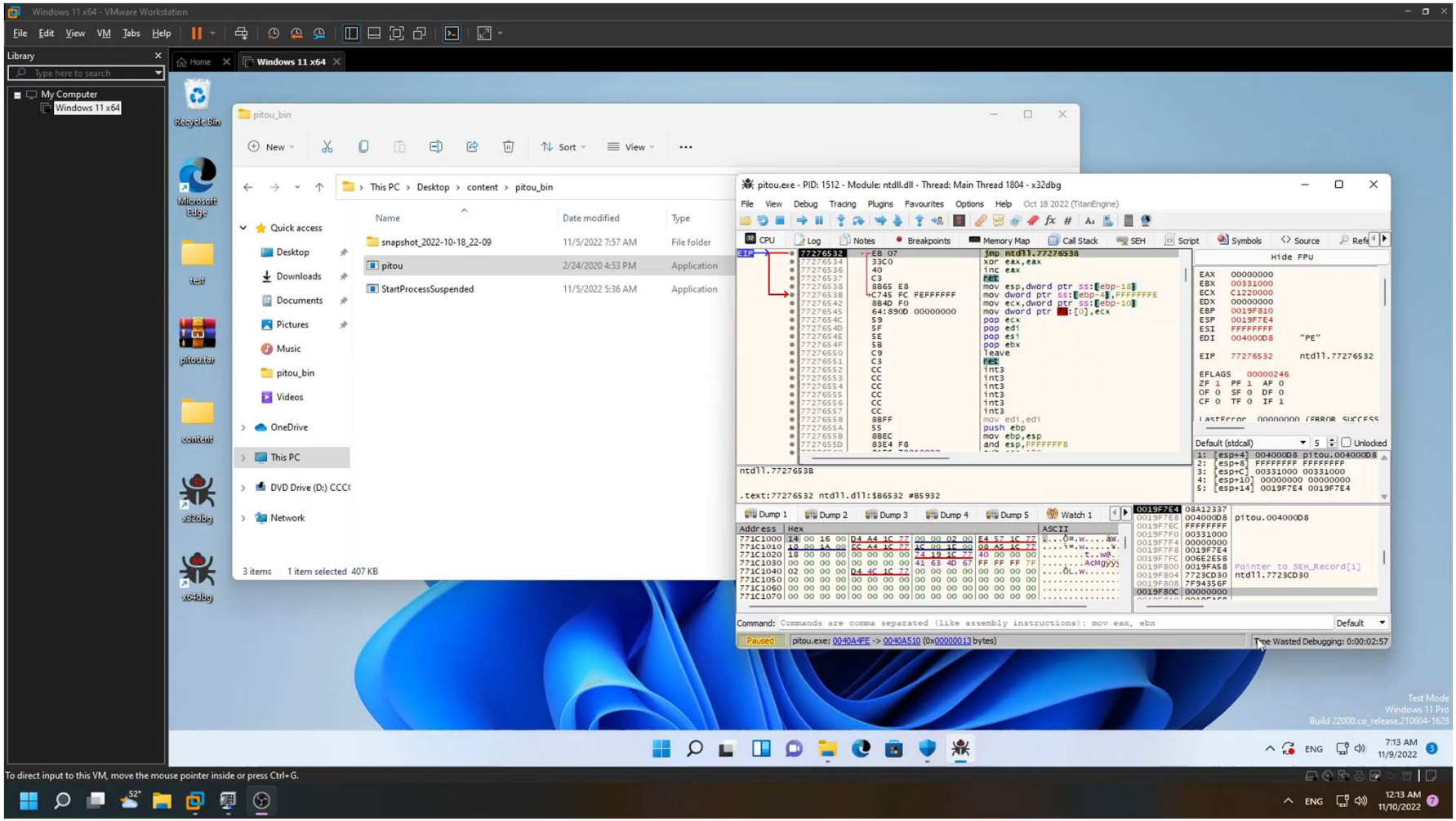
❹ HyperDbg Debugs both kernel and user mode

❸ Extensive use of CPU and Intel EPT

❷ VMX-Root mode compatible debugging facilities in the Guest are activated through Event-based mechanism

❶ HyperDbg Host-side functionality provides user CLI as well as Disassembler and SE Front-End

Pitou


Windows 11 x64 - VMware Workstation
pitou_bin
This PC > Desktop > content > pitou_bin
snapshot_2022-10-18_22-09
pitou
StartProcessSuspended
pitou.exe - PID: 1512 - Module: ntdll.dll - Thread: Main Thread 1804 - x32dbg

Debugger detected… Trying with HyperDbg

```
Select C:\Users\sina\Desktop\HyperDbg\HyperDbg\hyperdbg\build\bin\release\hyperdbg-cli.exe

.debug : debugs a target machine or makes this machine a debuggee.

syntax :        .debug [remote] [serial|namedpipe] [Baudrate (decimal)] [Address (string)]
syntax :        .debug [prepare] [serial] [Baudrate (decimal)] [Address (string)]
syntax :        .debug [close]

                e.g : .debug remote serial 115200 com3
                e.g : .debug remote namedpipe \\.\pipe\HyperDbgPipe
                e.g : .debug prepare serial 115200 com1
                e.g : .debug prepare serial 115200 com2
                e.g : .debug close

valid baud rates (decimal) : 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, 128000, 2
56000
valid COM ports : COM1, COM2, COM3, COM4

HyperDbg> .debug remote namedpipe \\.\pipe\HyperDbgPipe
waiting for debuggee to connect...
connected to debuggee Windows 10 Pro - Client 21H2 (OS Build 22000)
getting symbol details...
interpreting symbols and creating symbol maps
please configure the symbol path (use '.help .sympath' for more information)
press CTRL+C to pause the debuggee
debuggee is running...
```




In honor of all the Iranians who are risking their lives in pursuit of liberty and human rights