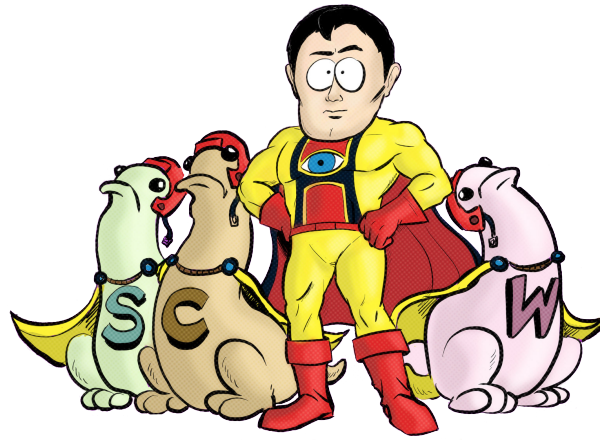




Memoirs of a Hindsight Hero: Detecting Rootkits in OS X

By
Cem Gürkök



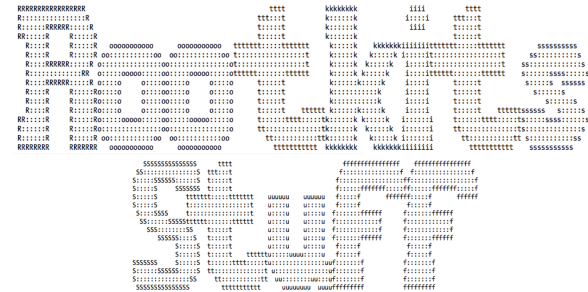
whoami

- A Hindsight Hero?
- InfoSec Researcher and Developer
- 10+ years of experience in the field
- Threat Intel' R & D Manager @Terremark
- 2nd place @ the Volatility Framework Contest 2013 with Windows Kernel Object Security and OS X Rootkit Detection Plugins



nemo@felinemenace.org

- Anti-forensics properties: Paste script into the interpreter without touching disk.
- No modification of standard rootkit vectors, syscall table/IDT etc.
- Safe, low risk of causing system failure and getting caught.
- Easy removal



Wendy's Paris



Don't detect me



Challenge accepted!





What's this all about?

“ There are known knowns; there are things we know that we know.
There are known unknowns; that is to say, there are things that we now know we don't know. ”
But there are also unknown unknowns – there are things we do not know we don't know.

—United States Secretary of Defense, Donald Rumsfeld

Rootkit Method	Type	Detecting Plugin
DTrace Hooks	Known Unknown	check_dtrace
Syscall Table Hooks	Known Unknown	check_hooks
Shadow Syscall Table	Known Unknown	check_hooks
IDT Hooks	Unknown Unknown	check_idt
Call Reference Modification	Known Unknown	check_hooks
Shadow TrustedBSD/ mac_policy_list	Known Unknown	check_hooks

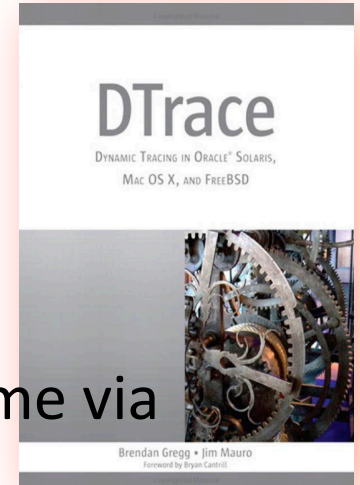


Some Definitions...

- **Syscall Table:** List of functions that permit a userland process to interact with the kernel (BSD level)
- **Mach Trap Table:** Prototypes of traps as seen from userland (Mach level syscalls)
- Function Hooking
 - **Direct:** Replace the function entry with the modified version's address
 - **Inline:** Keep original function entry in place, modify the function itself (e.g. prologue) to execute modified function by inserting trampolines, calls, or other instructions.



What's DTrace?



- Dynamic Tracing Framework [*]
- Built for Solaris, now on OS X and TrustedBSD
- Used for troubleshooting system issues in real time via providers, for example:
 - **syscall**: Let's a user monitor the entry point into the kernel from applications in userland and is not very OS specific
 - **fbt** (function boundary tracing): probes for almost all kernel functions, more useful when monitoring a particular behavior or issue in a specific kernel subsystem
 - **mach_trap**: fires on entry or return of the specified Mach library function
- Used for rootkit detection in the past by Beaucham and Weston [*]

* <http://www.dtracebook.com>

* http://blackhat.com/presentations/bh-usa-08/Beauchamp_Weston/BH_US_08_Beauchamp-Weston_DTrace.pdf



DTrace Artifacts in Memory

- How to detect DTrace activity?
- After some research...
- Artifacts depend on the provider that is used (syscall, fbt, mach_trap etc.)
 - **syscall**: Easy to detect, direct modification/hooks of the Syscall Table
 - **fbt**: Not so straight forward, inline modification of the probed function
 - **mach_trap**: Easy to detect, direct modification/hooks of the Mach Trap Table



DTrace Hooks Detection



```
$ python vol.py mac_check_dtrace -f ~/Downloads/MacMemoryReader/ram_dump-after.mach-o --  
profile=MacLion_10_7_5_AMDx64
```

Volatile Systems Volatility Framework 2.3_beta

Table Name	Index	Address	Symbol	D-Trace Probe
Syscall_Table	344	0xffffffff80005c89e0	_dtrace_systrace_syscall	syscall_probe

```
$ python vol.py mac_check_dtrace -f ~/Downloads/MacMemoryReader/ram_dump-trap.mach-o --  
profile=MacLion_10_7_5_AMDx64
```

Volatile Systems Volatility Framework 2.3_beta

Table Name	Index	Address	Symbol	D-Trace Probe
Trap_Table	46	0xffffffff80285dbc30	_dtrace_machtrace_syscall	mach_trap_probe

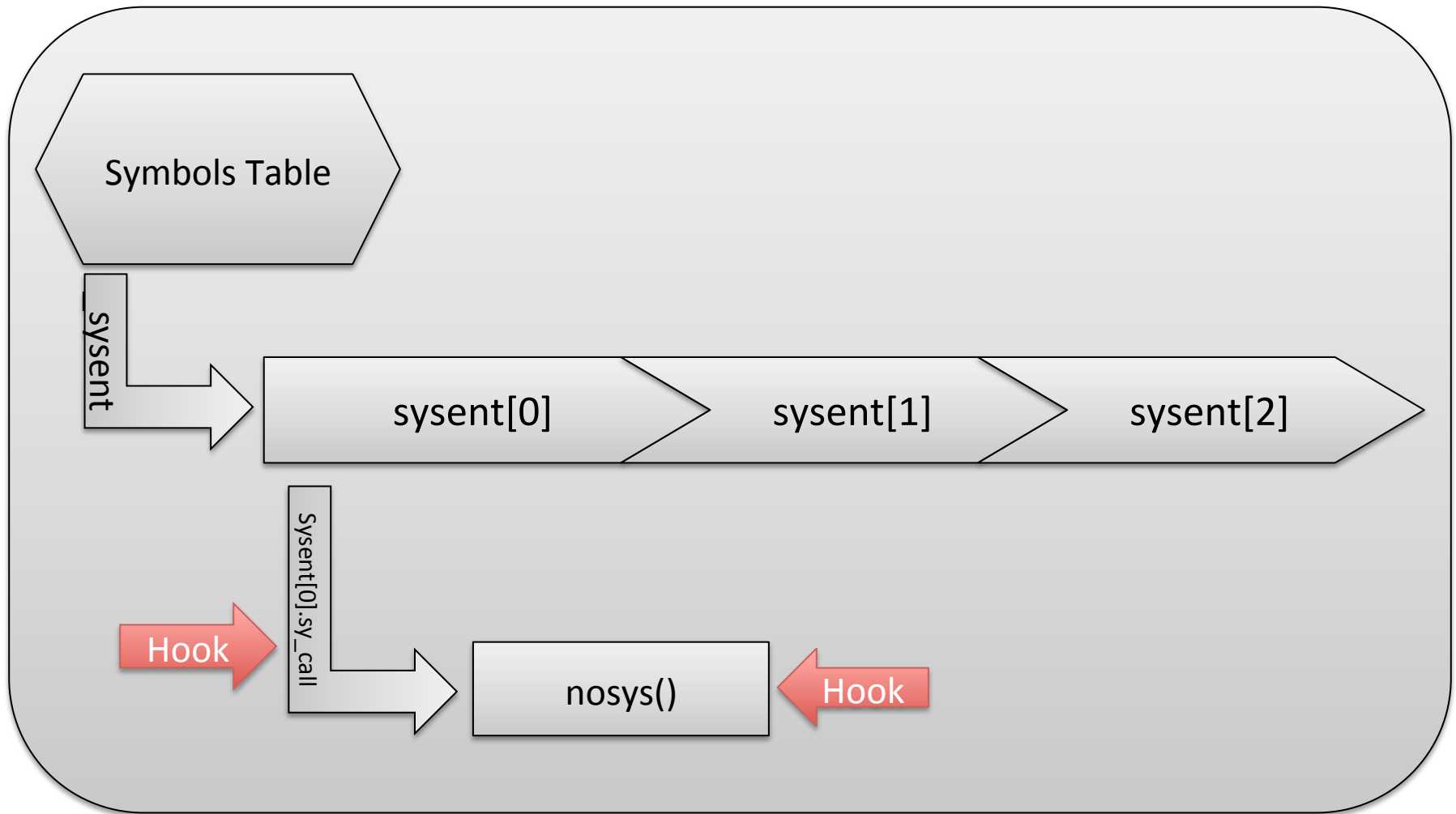
```
$ python vol.py mac_check_dtrace -f ~/Downloads/MacMemoryReader/ram_dump-fbt.mach-o --  
profile=MacLion_10_7_5_AMDx64
```

Volatile Systems Volatility Framework 2.3_beta

Table Name	Index	Address	Symbol	D-Trace Probe
Syscall_Table	344	0xffffffff8000306fb0	_getdirenties64	fbt_probe



Syscall Table Hooks





Detecting Syscall Table Hooks

```
$ python vol.py mac_check_hooks -f ~/Downloads/MacMemoryReader/ram_dump-dtrace.mach-o  
--profile=MacLion_10_7_4_AMDx64
```

Volatile Systems Volatility Framework 2.3_beta

Table Name	Index	Address	Symbol	Inlined	Shadowed	Perms	Hook In
SyscallTable	344	0xffffffff80005c89e0	[HOOKED] _dtrace_systrace_syscall	No	No	-	__kernel__

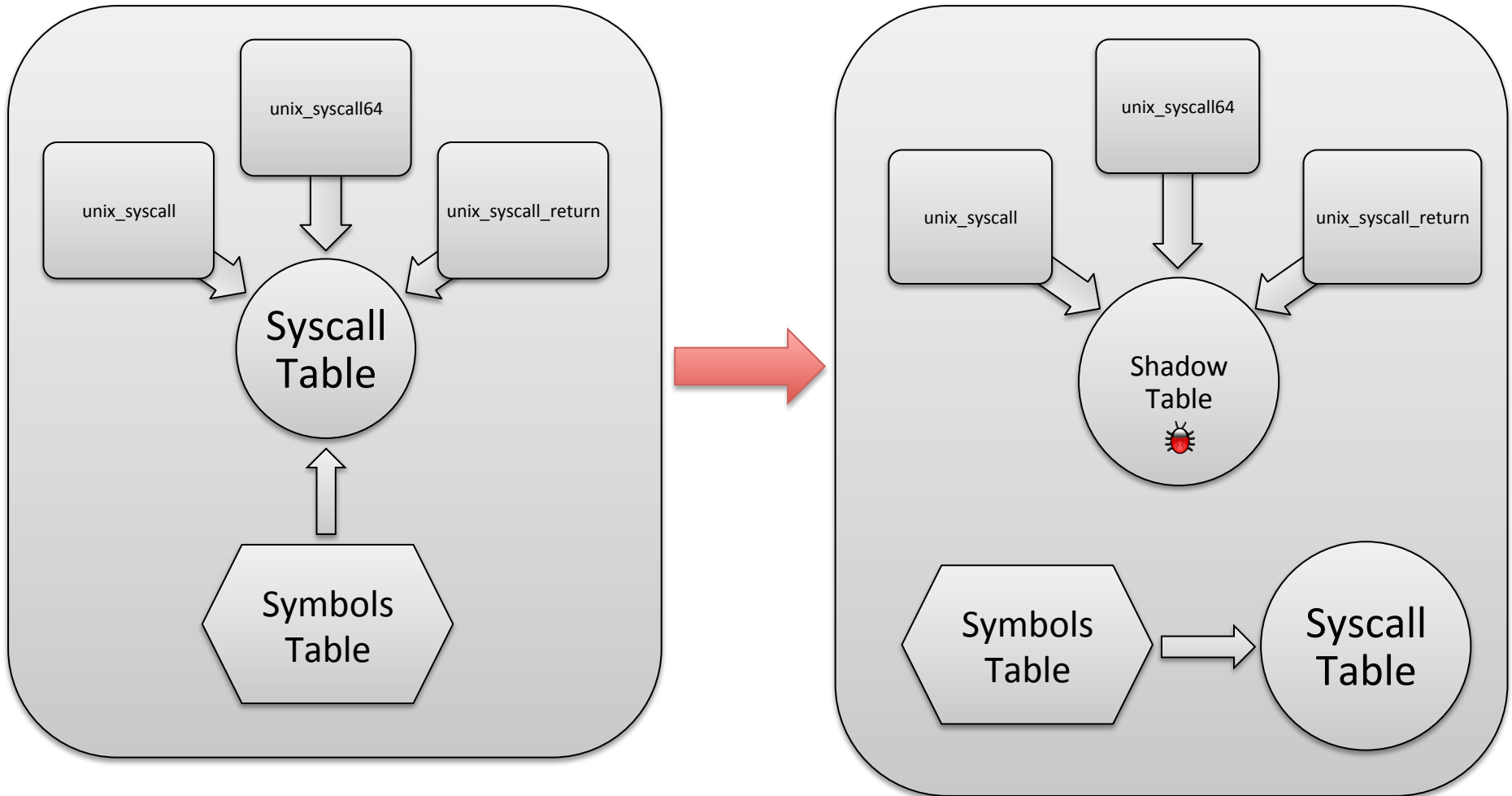
```
$ python vol.py mac_check_hooks -f ~/Downloads/MacMemoryReader/ram_dump-fbt.mach-o --  
profile=MacLion_10_7_5_AMDx64
```

Volatile Systems Volatility Framework 2.3_beta

Table Name	Index	Address	Symbol	Inlined	Shadowed	Perms	Hook In
SyscallTable	344	0xffffffff8000306fb0	_getdirenties64	Yes	No	-	__kernel__



Shadow Syscall Table





Detecting the Shadow Syscall Table

- To detect the Shadow Syscall Table
 1. Check functions known to have references to the syscall table: `unix_syscall_return`, `unix_syscall64`, `unix_syscall`
 2. Disassemble them to find the syscall table references.
 3. Obtain the references in the function and compare to the address in the symbols table.
- All incorporated into the `check_hooks` plugin!



Detecting the Shadow Syscall Table

```
python vol.py mac_check_hooks -f /Volumes/Storage/HITB/ShadowSyscall-  
MountainLion_10_8_3_AMDx64.vmem --profile=MacMountainLion_10_8_3_AMDx64
```

```
sysent table is shadowed at _unix_syscall_return: 0xffffffff800f3e084b ADD R15, [RIP+0x21f87e]  
shadow sysent table is at 0xffffffff7f907b2350  
sysent table is shadowed at _unix_syscall_return: 0xffffffff800f3e0852 CMP R15, [RIP+0x21f877]  
shadow sysent table is at 0xffffffff7f907b2350  
sysent table is shadowed at _unix_syscall_return: 0xffffffff800f3e0996 ADD R15, [RIP+0x21f733]  
shadow sysent table is at 0xffffffff7f907b2350  
sysent table is shadowed at _unix_syscall_return: 0xffffffff800f3e09d3 CMP R15, [RIP+0x21f6f6]  
shadow sysent table is at 0xffffffff7f907b2350  
sysent table is shadowed at _unix_syscall64: 0xffffffff800f3e04ac ADD R13, [RIP+0x21fc1d]  
shadow sysent table is at 0xffffffff7f907b2350  
sysent table is shadowed at _unix_syscall64: 0xffffffff800f3e04b3 CMP R13, [RIP+0x21fc16]  
shadow sysent table is at 0xffffffff7f907b2350  
sysent table is shadowed at _unix_syscall64: 0xffffffff800f3e04e9 ADD R13, [RIP+0x21fbe0]  
shadow sysent table is at 0xffffffff7f907b2350  
sysent table is shadowed at _unix_syscall64: 0xffffffff800f3e059b ADD R13, [RIP+0x21fb2e]  
shadow sysent table is at 0xffffffff7f907b2350  
sysent table is shadowed at _unix_syscall: 0xffffffff800f3e020a ADD RBX, [RIP+0x21febf]  
shadow sysent table is at 0xffffffff7f907b2350  
sysent table is shadowed at _unix_syscall: 0xffffffff800f3e0216 CMP RBX, [RIP+0x21feb3]  
shadow sysent table is at 0xffffffff7f907b2350  
sysent table is shadowed at _unix_syscall: 0xffffffff800f3e0246 ADD RBX, [RIP+0x21fe83]  
shadow sysent table is at 0xffffffff7f907b2350
```

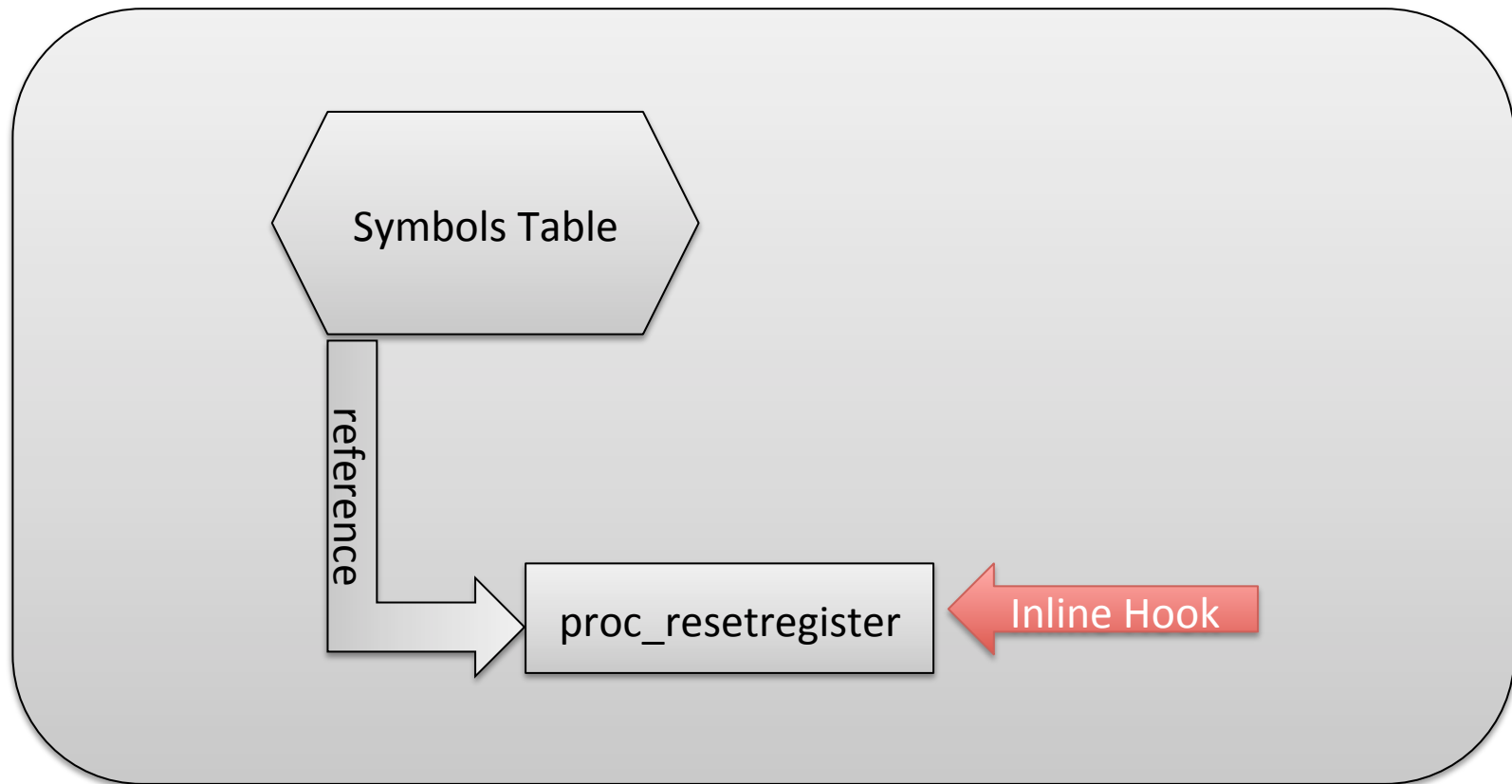



Symbols Table Hooks

- Functions are exposed by the kernel and kexts in their symbols tables.
- These functions can also be hooked using the techniques that have been described (direct or inline).
- To check the functions, need to obtain the list of symbols.
- Then check for modifications that cause the execution to continue in an external kext/module.



Symbols Table Hooks





Hooking

Symbols Table Functions

- Hydra [*], a kext that intercepts a process's creation
- Inline hooks `proc_resetregister`, a function in the kernel symbols
- The destination of the hook is in the 'put.as.hydra' kext
- Download, and compile Hydra with Xcode
 - Copy compiled kext into `/System/Library/Extensions/`
 - `sudo chown -R root:wheel /System/Library/Extensions/hydra.kext`
 - `sudo chmod -R 755 /System/Library/Extensions/hydra.kext`
 - `sudo kextload /System/Library/Extensions/hydra.kext`
- Used the `check_hooks` plugin to find the hook

* "Hydra," Pedro Vilaca, <https://github.com/gdbinit/hydra>



Detecting Symbols Table Hooks

```
$ python vol.py mac_check_hooks -f ~/Documents/Virtual\ Machines/Mac\ OS\ X\ 10.8\ 64-bit-DEMO.vmx/564d438d-cc29-2121-3dd6-ac473e701f8d.vmem --  
profile=MacMountainLion_10_8_5_AMDx64 -K
```

Volatility Foundation Volatility Framework 2.3

Table Name	Index	Address	Symbol	Inlined	Shadowed	Perms	Hook In
SymbolsTable	-	0xffffffff8000b68fe0	proc_resetregister	Yes	No	-	put.as.hydra

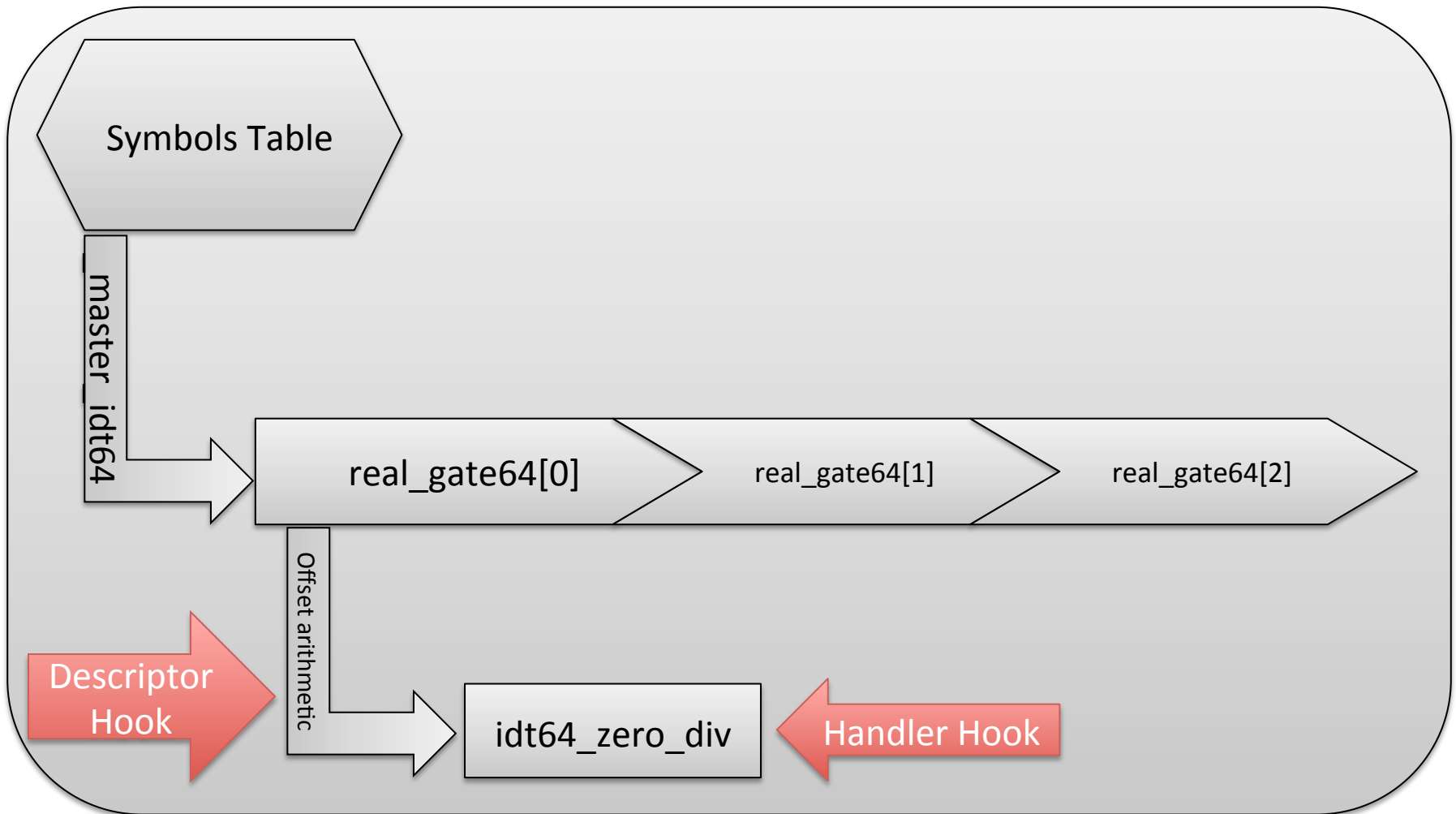


Hooking the IDT

- Interrupt descriptor table (IDT)
- Associates each interrupt or exception identifier (handler) with a descriptor (vector).
- Descriptors have the instructions for the associated event.
- An interrupt is usually defined as an event that alters the sequence of instructions executed by a processor.
- IDT entries: Interrupt Gates, Task Gates and Trap Gates...
- Why hook the IDT?
 - Because it gives us ring 0 or root access!



Hooking the IDT





Hooking the IDT Descriptor

- Hooked `idt64_zero_div` and redirected to `idt64_stack_fault`
- Used both hooking methods

```
testers-Mac:~ tester$ ./div0  
Floating point exception: 8
```



```
testers-Mac:~ tester$ ./div0  
Illegal instruction: 4_
```



Detecting the Descriptor Hook

```
$ python vol.py mac_check_idt -f /Volumes/Storage/HITB/IDTDescriptorHook-  
MountainLion_10_8_3_AMDx64.vmem --profile=MacMountainLion_10_8_3_AMDx64
```

Volatile Systems Volatility Framework 2.3_beta									
CPU#	Index	Address	Symbol	Ring	Selector	Module		Hooked	Inlined
0	0	0xffffffff7f8dfba6e5	UNKNOWN	0	KERNEL64_CS	com.vmware.kext.vmhgfs		Yes	Yes
0	1	0xffffffff800c8cd030	_idt64_debug	0	KERNEL64_CS	__kernel__		No	No
0	2	0xffffffff800c8cac40	_intr_0x02	0	KERNEL64_CS	__kernel__		No	No
0	3	0xffffffff800c8cac60	_idt64_int3	3	KERNEL64_CS	__kernel__		No	No
0	4	0xffffffff800c8cac80	_idt64_int0	3	KERNEL64_CS	__kernel__		No	No
0	5	0xffffffff800c8caca0	_idt64_bounds	3	KERNEL64_CS	__kernel__		No	No
0	6	0xffffffff800c8cacc0	_idt64_invop	0	KERNEL64_CS	__kernel__		No	No
0	7	0xffffffff800c8cace0	_idt64_nofpu	0	KERNEL64_CS	__kernel__		No	No
0	8	0xffffffff800c8cd0e0	_idt64_double_fault	0	KERNEL64_CS	__kernel__		No	No
0	9	0xffffffff800c8cad00	_idt64_fpu_over	0	KERNEL64_CS	__kernel__		No	No
0	10	0xffffffff800c8cad20	_idt64_inv_tss	0	KERNEL64_CS	__kernel__		No	No
0	11	0xffffffff800c8cd160	_idt64_segnp	0	KERNEL64_CS	__kernel__		No	No
0	12	0xffffffff800c8cd140	_idt64_stack_fault	0	KERNEL64_CS	__kernel__		No	No



Detecting the Handler Hook

```
$ python vol.py mac_check_idt -f /Volumes/Storage/HITB/IDTHandlerHook-  
MountainLion_10_8_3_AMDx64.vmem --profile=MacMountainLion_10_8_3_AMDx64
```

Volatile Systems Volatility Framework 2.3_beta									
CPU#	Index	Address	Symbol	Ring	Selector	Module		Hooked	Inlined
0	0	0xffffffff802a4cac20	_idt64_zero_div	0	KERNEL64_CS	__kernel__		No	Yes
0	1	0xffffffff802a4cd000	_idt64_debug	0	KERNEL64_CS	__kernel__		No	No
0	2	0xffffffff802a4cac40	_intr_0x02	0	KERNEL64_CS	__kernel__		No	No
0	3	0xffffffff802a4cac60	_idt64_int3	3	KERNEL64_CS	__kernel__		No	No
0	4	0xffffffff802a4cac80	_idt64_into	3	KERNEL64_CS	__kernel__		No	No
0	5	0xffffffff802a4caca0	_idt64_bounds	3	KERNEL64_CS	__kernel__		No	No
0	6	0xffffffff802a4cacc0	_idt64_invop	0	KERNEL64_CS	__kernel__		No	No
0	7	0xffffffff802a4cace0	_idt64_nofpu	0	KERNEL64_CS	__kernel__		No	No
0	8	0xffffffff802a4cd0e0	_idt64_double_fault	0	KERNEL64_CS	__kernel__		No	No
0	9	0xffffffff802a4cad00	_idt64_fpu_over	0	KERNEL64_CS	__kernel__		No	No
0	10	0xffffffff802a4cad20	_idt64_inv_tss	0	KERNEL64_CS	__kernel__		No	No
0	11	0xffffffff802a4cd160	_idt64_segnp	0	KERNEL64_CS	__kernel__		No	No
0	12	0xffffffff802a4cd140	_idt64_stack_fault	0	KERNEL64_CS	__kernel__		No	No



Breaking Volatility?!?



- fG! once more claims to break Volatility by:
 - Modifying Call References
 - Shadow TrustedBSD/
mac_policy_list
 - Hiding from Memory Acquisition... Irrelevant!





Call Reference Modification

- Modified **ps_read_file** function
- Calls **vnode_pagein**
- Redirected call to an address in the kext
com.vmware.kext.vmhgfs
- Tool? Volatility!

```
0xffffffff8002e049b0 55          PUSH RBP
0xffffffff8002e049b1 4889e5      MOV RBP, RSP
0xffffffff8002e049b4 4156      PUSH R14
0xffffffff8002e049b6 53          PUSH RBX
0xffffffff8002e049b7 4883ec10   SUB RSP, 0x10
0xffffffff8002e049bb 4d89ce      MOV R14, R9
0xffffffff8002e049be 4489c0      MOV EAX, R8D
0xffffffff8002e049c1 c1e80c     SHR EAX, 0xc
0xffffffff8002e049c4 488d1d75396b00 LEA RBX, [RIP+0x6b3975]
0xffffffff8002e049cb ff0483     INC DWORD [RBX+RAX*4]
0xffffffff8002e049ce 034f18     ADD ECX, [RDI+0x18]
0xffffffff8002e049d1 488b3f     MOV RDI, [RDI]
0xffffffff8002e049d4 448b4d10   MOV R9D, [RBP+0x10]
0xffffffff8002e049d8 48c7042400000000 MOV QWORD [RSP], 0x0
0xffffffff8002e049e0 e85b313c00 CALL 0xffffffff80031c7b40
...
```



```
0xffffffff8002e049b0 55          PUSH RBP
0xffffffff8002e049b1 4889e5      MOV RBP, RSP
0xffffffff8002e049b4 4156      PUSH R14
0xffffffff8002e049b6 53          PUSH RBX
0xffffffff8002e049b7 4883ec10   SUB RSP, 0x10
0xffffffff8002e049bb 4d89ce      MOV R14, R9
0xffffffff8002e049be 4489c0      MOV EAX, R8D
0xffffffff8002e049c1 c1e80c     SHR EAX, 0xc
0xffffffff8002e049c4 488d1d75396b00 LEA RBX, [RIP+0x6b3975]
0xffffffff8002e049cb ff0483     INC DWORD [RBX+RAX*4]
0xffffffff8002e049ce 034f18     ADD ECX, [RDI+0x18]
0xffffffff8002e049d1 488b3f     MOV RDI, [RDI]
0xffffffff8002e049d4 448b4d10   MOV R9D, [RBP+0x10]
0xffffffff8002e049d8 48c7042400000000 MOV QWORD [RSP], 0x0
0xffffffff8002e049e0 e8008d8481 CALL 0xffffffff7f8464d6e5
...
```



Detecting Call Reference Modification

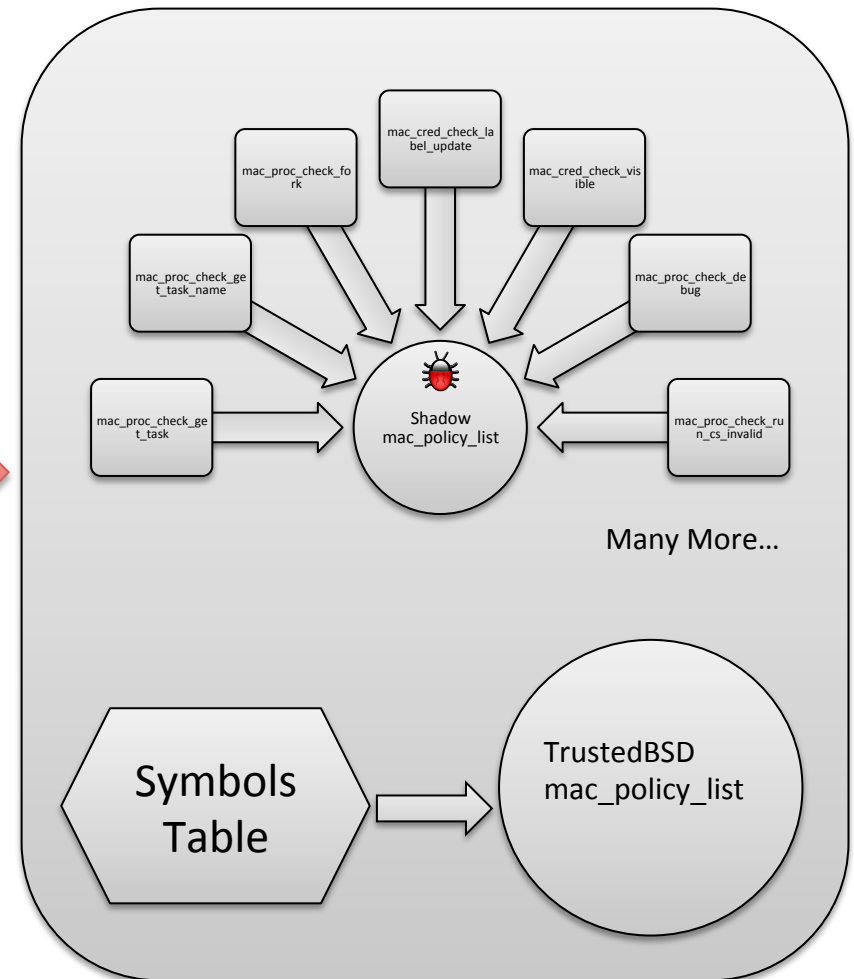
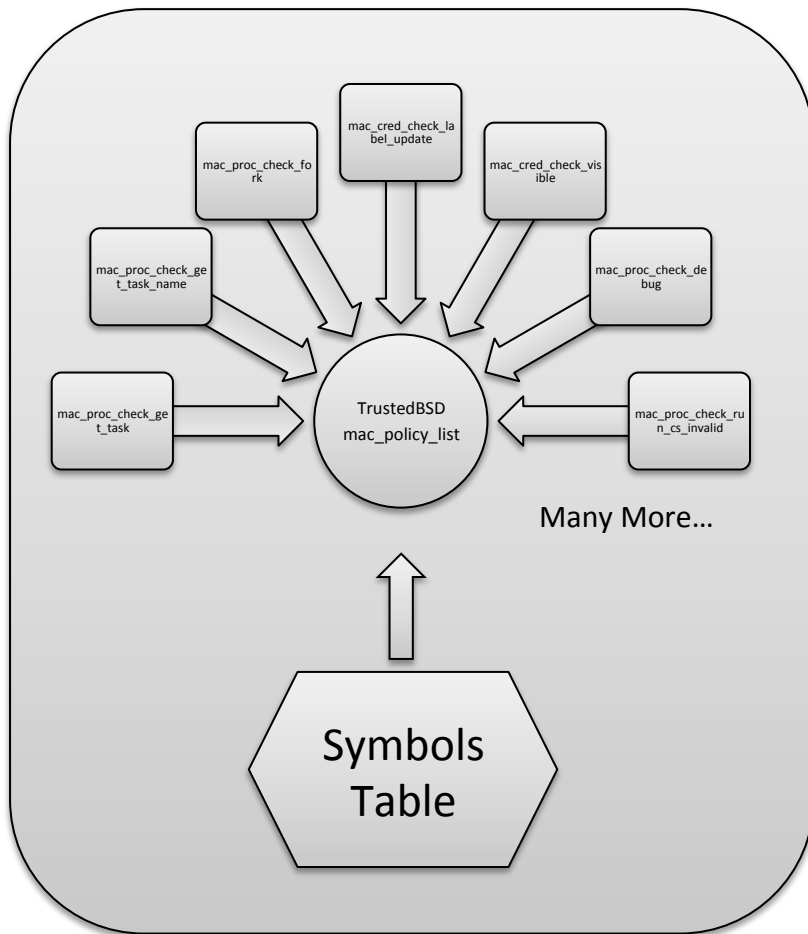
```
$ python vol.py mac_check_hooks -f ~/Documents/Virtual\ Machines/Mac\ OS\ X\ 10.8\ 64-bit-DEMO.vmx/564d438d-cc29-2121-3dd6-ac473e701f8d.vmem --  
profile=MacMountainLion_10_8_5_AMDx64 -K
```

Volatility Foundation Volatility Framework 2.3.1

Table Name	Index	Address	Symbol	Inlined	Shadowed	Perms	Hook In
SymbolsTable	-	0xffffffff8002e049b0	_ps_read_file	Yes	No	-	com.vmware.kext.vmhgfs



Shadow TrustedBSD or mac_policy_list





Detecting Shadow TrustedBSD

- All functions for Mach Trap Table include the macro `MAC_CHECK`, so not as easy as Shadow Symbols table
- Need to scan all Trap Table functions for proper referencing (for Rex scan only `mac_proc_check_get_task`)
- Rex could have used the **`mac_policy_list.entries`** member to modify and shadow instead, also detected by `check_hooks`!

```
$ python vol.py mac_check_hooks -f ~/Desktop/OMFW-2013/564d438d-cc29-2121-3dd6-ac473e701f8d.vmem --profile=MacMountainLion_10_8_5_AMDx64
```

```
Volatility Foundation Volatility Framework 2.3.1
```

Table Name	Index	Address	Symbol	Inlined	Shadowed	Perms	Hook In
------------	-------	---------	--------	---------	----------	-------	---------

```
-----  
mac_policy_address is shadowed! Original Address: 0xffffffff8024af4d28, Shadow Address: 0xffffffff7fa5c4d6e5, Modification  
at: 0xffffffff802488ee34
```



Conclusion

- DTrace is part of OS X and readily available
- Can be used to detect and create rootkits
- Syscalls and other system functions/structures are easy targets for rootkits
- Memory analysis with Volatility reveals rootkit artifacts
- Detection methods trivially wrapped into a plugin for automation
- If there is no detection mechanism, write a Volatility plugin!



Questions?

Thank you!

- Blog: siliconblade.blogspot.com
- Code: github.com/siliconblade/
- Twitter: @CGurkok
- E-mail: [cemgurkok <at/> gmail.com](mailto:cemgurkok@gmail.com)

