



Livegrid™



Stealthy Rootkit

How bad guy fools live memory forensics?

Tsukasa Ooi <li@livegrid.org>
Livegrid Incorporated, Lead Analyst



Related Topics

- Live Memory Forensics
- Anti-forensics
- Rootkits



What is “Live Memory Forensics?”

- Forensics based on memory contents of running machine
 - Finding “Evidence”
 - Searching malware
- Normally, we use physical memory dump acquired by dedicated tools
 - Contents of physical memory



Live Memory Forensics Process

1. Acquire Physical Memory with Dedicated Tool
 - a. Map physical memory range
 - b. Copy memory mapped contents
 - c. Save contents
2. Put memory dump into Forensic Software
3. Analyze with Forensic Software
 - a. Recover System contents from dump
 - b. Analyze recovered system



What is “Anti-forensics?” (1)

- Way to prevent/fake forensics
- Not only Attackers!
 - “Bad guy” who wants to get rid of investigation
CAN INSTALL rootkit on his OWN machine
to fool investigator.



What is “Anti-forensics?” (2)

- EXISTING “Anti-Forensic Way” can have Great Meaning...
 - Reliability of Forensics is lowered...
 - No longer needed to “hack”
 - Can lose in court even if you have “evidence”
- Reliability of Forensics is Very Important



Live Memory Forensics Process

1. Acquire Physical Memory with Dedicated Tool
 - a. Map physical memory range
 - b. Copy memory range to host
We want to exploit this!
 - c. Save contents
2. Put memory dump into Forensic Software
3. Analyze with Forensic Software
 - a. Recover System contents from dump
 - b. Analyze recovered system



Livegrid™



Result was...

WHAT ROOTKIT CAN DO?



Like “Mythbusters” ...

- EnCase **BUSTED**
- WinEN **BUSTED**
- FastDump **BUSTED**
- dd, Forensic Acquisition Tools, win32dd, Memory DD... **BUSTED**
- Windows Hibernation (hiberfil.sys) **BUSTED**
- Microsoft Windows Crashdump (*.dmp) **BUSTED**



Livegrid™



Fact (1):

**INVISIBLE FROM ALMOST
ALL FORENSIC SOFTWARE**



Why rootkit hidden from these?

- Subverting Page Table(s)
 - Shadow Paging
- Weakness of current (dedicated) forensic tools

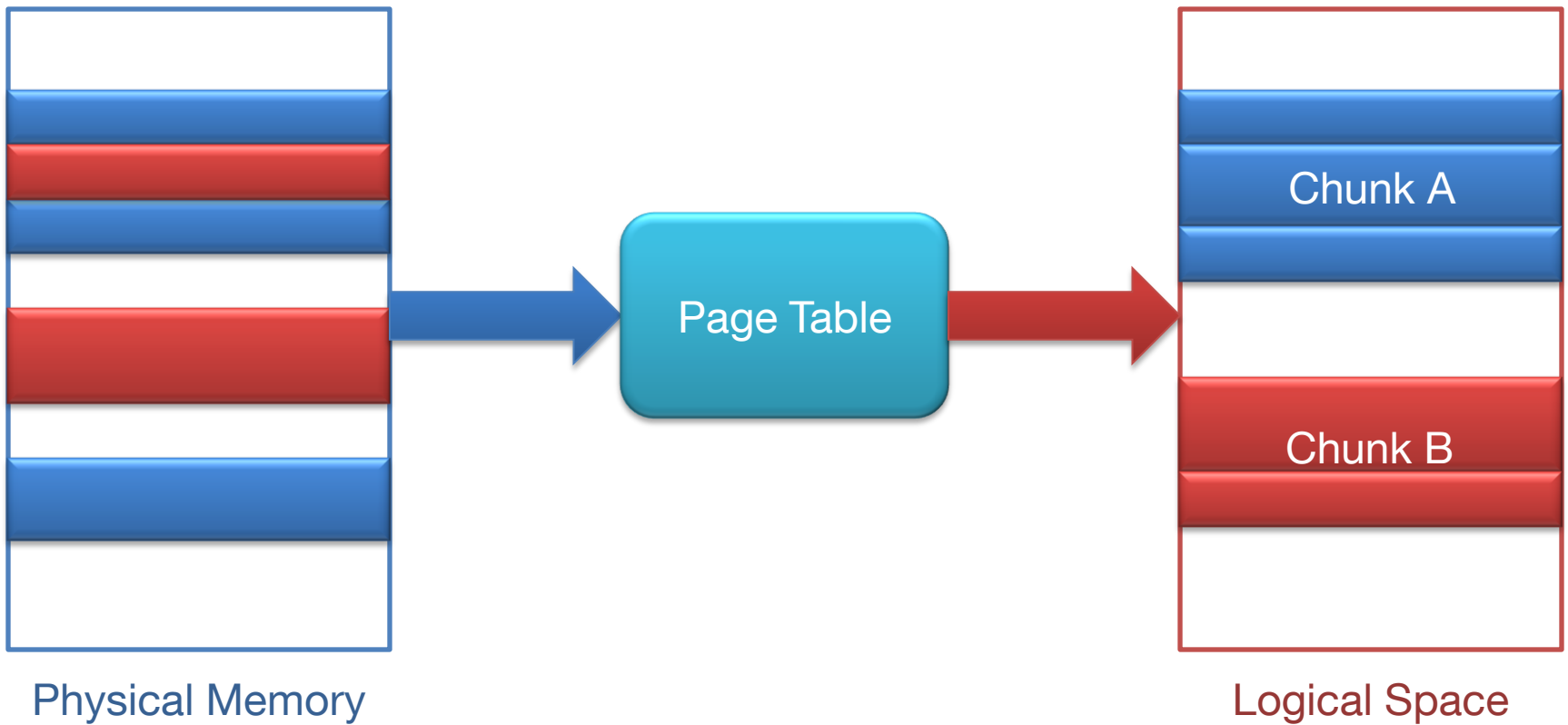


What is Page Table? (1)

- A Mechanism to separate Physical memory layout and Logical structure
- Table to translate Linear Address (that CPU use) to the physical one
- Manage Logical memory space with divided chunk (called “page”), normally 4KiB block

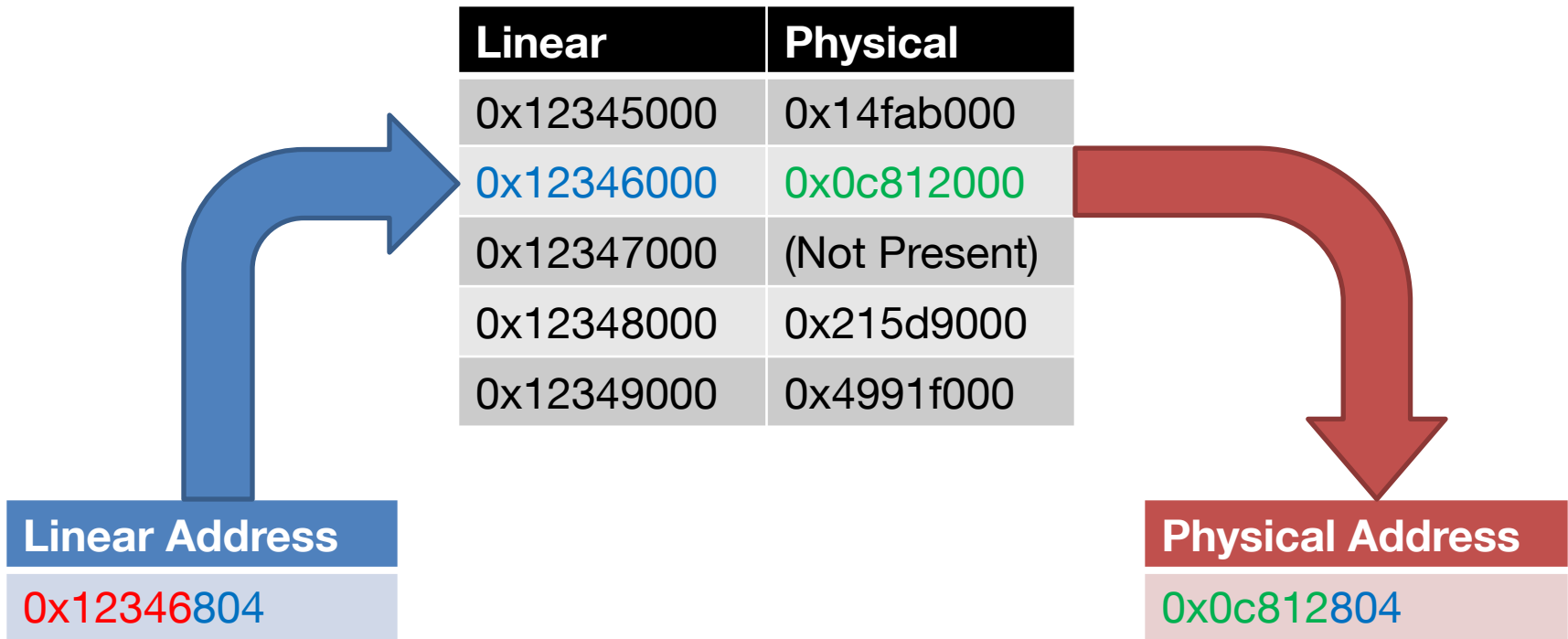


What is Page Table? (2)



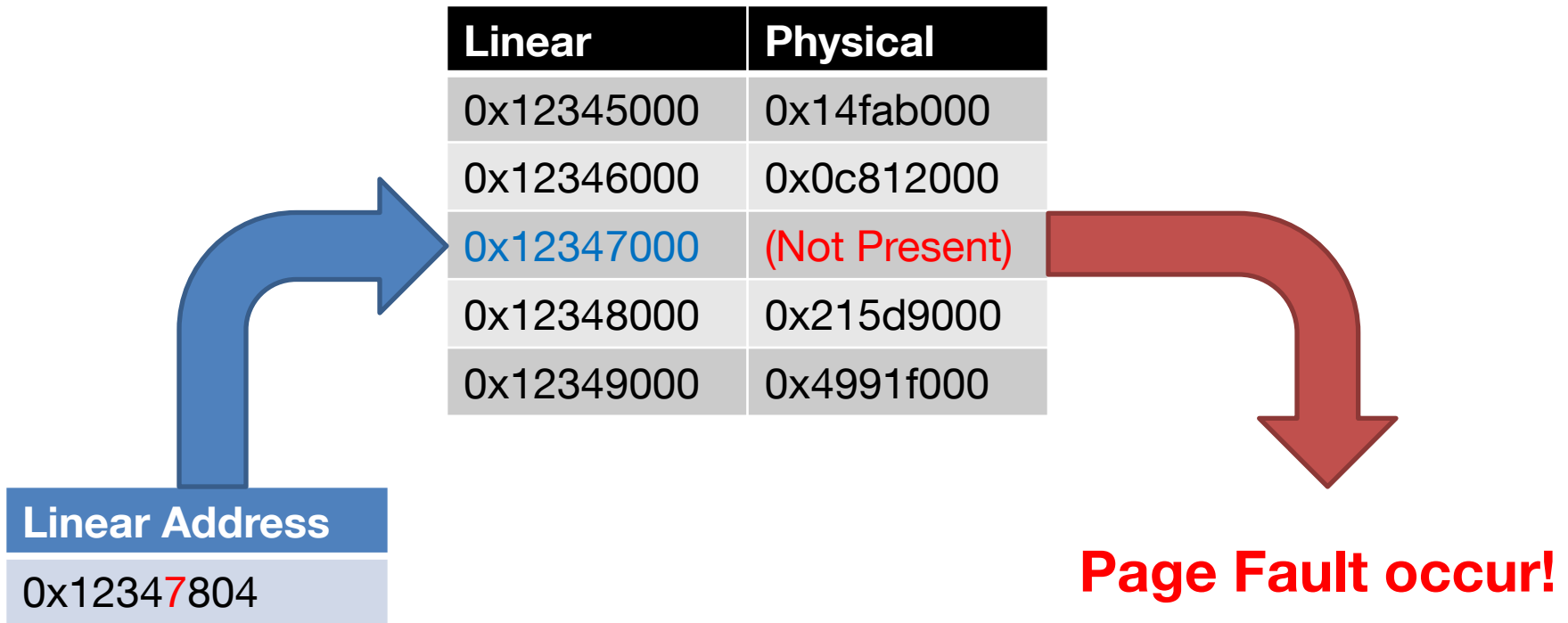


What is Page Table? (3)





What is Page Table? (4)





What is Page Table? (5)

- Page Fault (exception)
 - Occur when...
 - There is no Physical Address present associated to Linear address
 - Access Violating Page's access control
 - Page Fault Handler specified by IDT (Interrupt Descriptor Table) handles this exception
- Access Control of Page
 - Physical Address of Page is Present (P bit)
 - Is Page Writable (R/W bit)
 - Is Page access Prohibited by application (U/S bit)
 - Is Page execution Prohibited (NX bit)



What is Page Table? (6)

- System Registers related to Page Table
 - CR3
Specify Physical Address of Page Table
 - CR2
When Page Fault occur, Linear Address that caused Page Fault is stored



What is Page Table? (7)

- Page Table is very important structure to manage memory!
- Also Useful for rootkits...



Livegrid™



Way to do this:

TAKE CONTROL OF PAGE TABLE



Shadow Paging (1)

- Originally, This is one of the way to virtualize memory of guest OS
- Make a copy of Page Table and Reflect one Shadow Page Table change from original Page Table's one
- Normally, Shadow Paging cannot be implemented without CPU support of virtualization (such as Intel VT-x, AMD-V...)



“NORMAL” Shadow Paging (1)

- Write-protect Page Table of guest OS
- Detect Page Table write of guest OS by Page Fault and do correspond changes to “SHADOW” Page Table

Linear	Physical
0x12345000	0x14fab000
0x12346000	0x0c812000
0x12347000	0x44ab1000 → 0x31442000
0x12348000	0x215d9000
0x12349000	0x4991f000



Linear	Physical (Virtualized)	Physical (Real)
0x12345000	0x14fab000	0x24f03000
0x12346000	0x0c812000	0x12580000
0x12347000	0x44ab1000 → 0x31442000	0x1dabc000 → 0x4d3f2000
0x12348000	0x215d9000	0x379df000
0x12349000	0x4991f000	0x0732f000

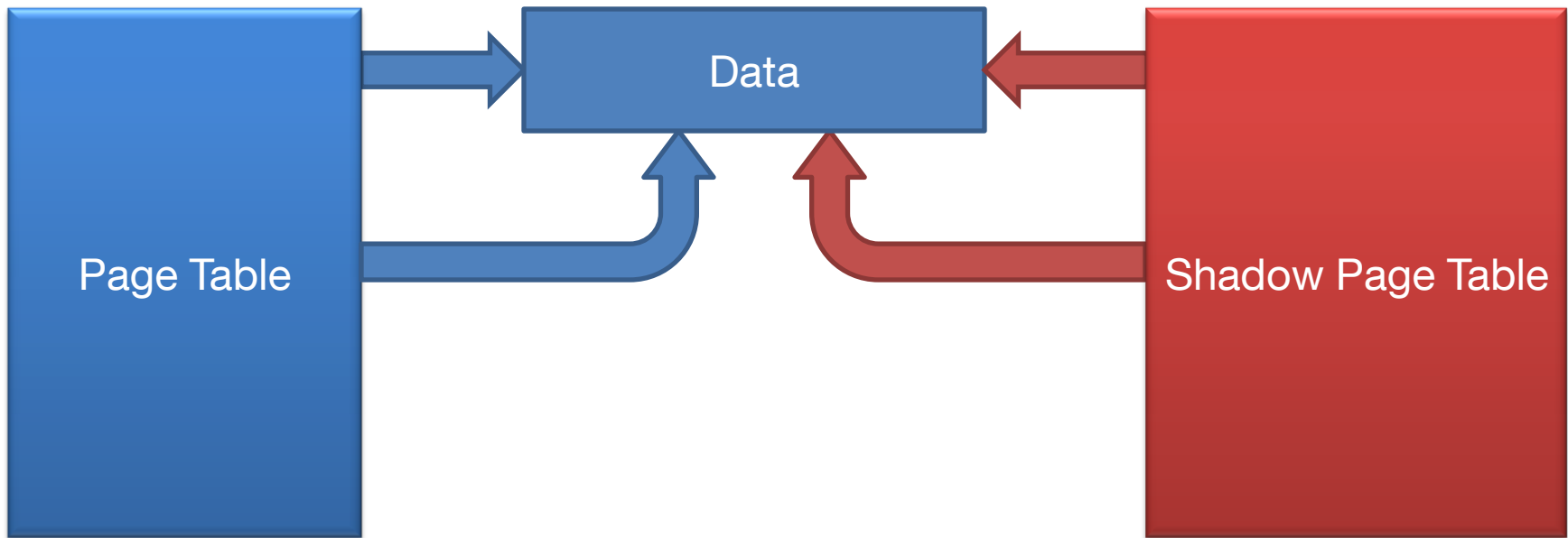
(Virtualized) Page Table

Shadow (Real) Page Table



“NORMAL” Shadow Paging (2)

- Normally, two Page Table Entries points same data are “SAME”





Shadow Paging (2)

- Despite of this, Shadow Paging can be implemented if some conditions are fulfilled
 - CR0.WP bit is always set
 - All Changes of CR3 registers are done by known point of operating system
 - Page Table Handler is hooked
 - (Other conditions are abridged)
- In normal condition, most of operating system fulfills these conditions (Windows, Linux...)



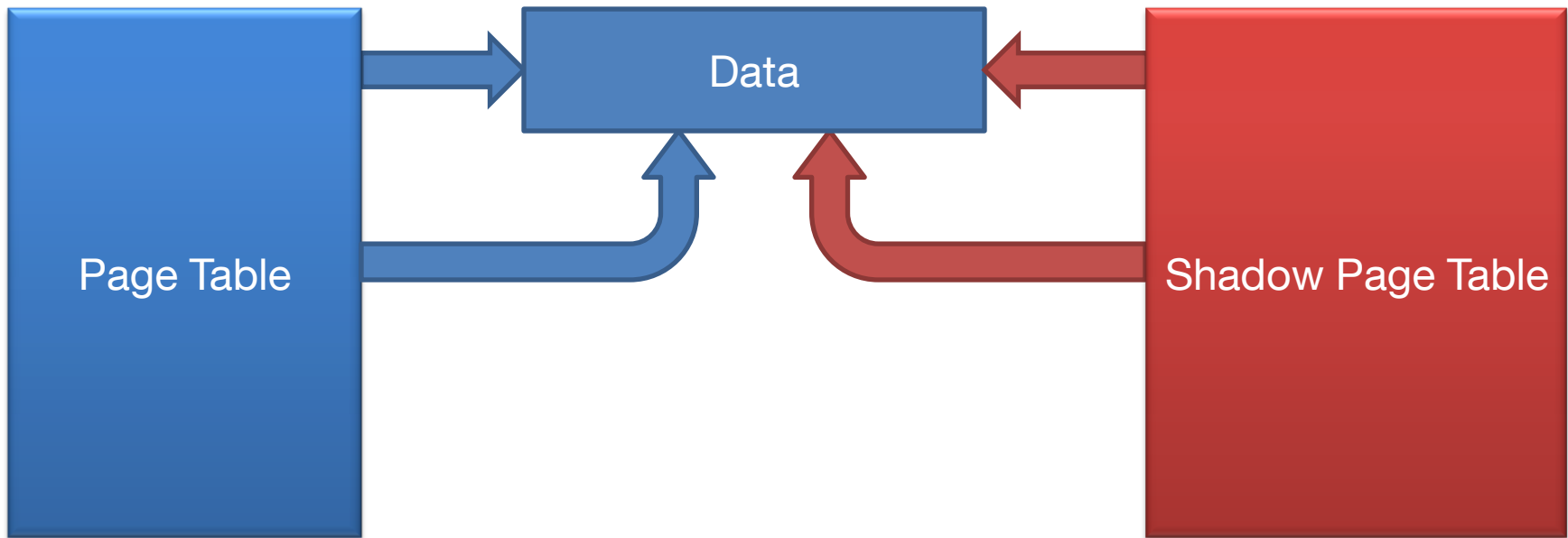
Taking control of Page Table!

1. Set own (rootkit's) Page Fault Handler
2. Make copy of Page Table (for rootkit) and Do Shadow Paging
3. Impersonate (or camouflage) address range of re-mapped memory



“NORMAL” Shadow Paging (2)

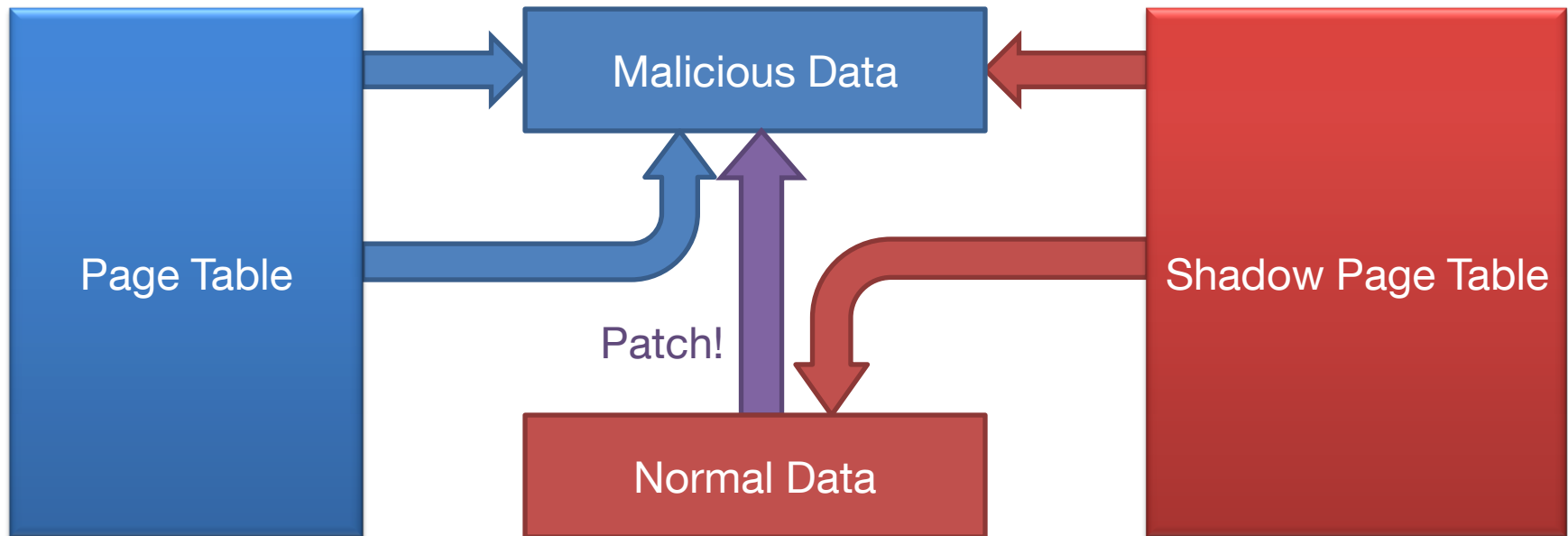
- Normally, two Page Table Entries points same data are “SAME”





“MALICIOUS” Shadow Paging (1)

- In “Malicious” Shadow Paging, modify mapping of certain (malicious) memory ranges



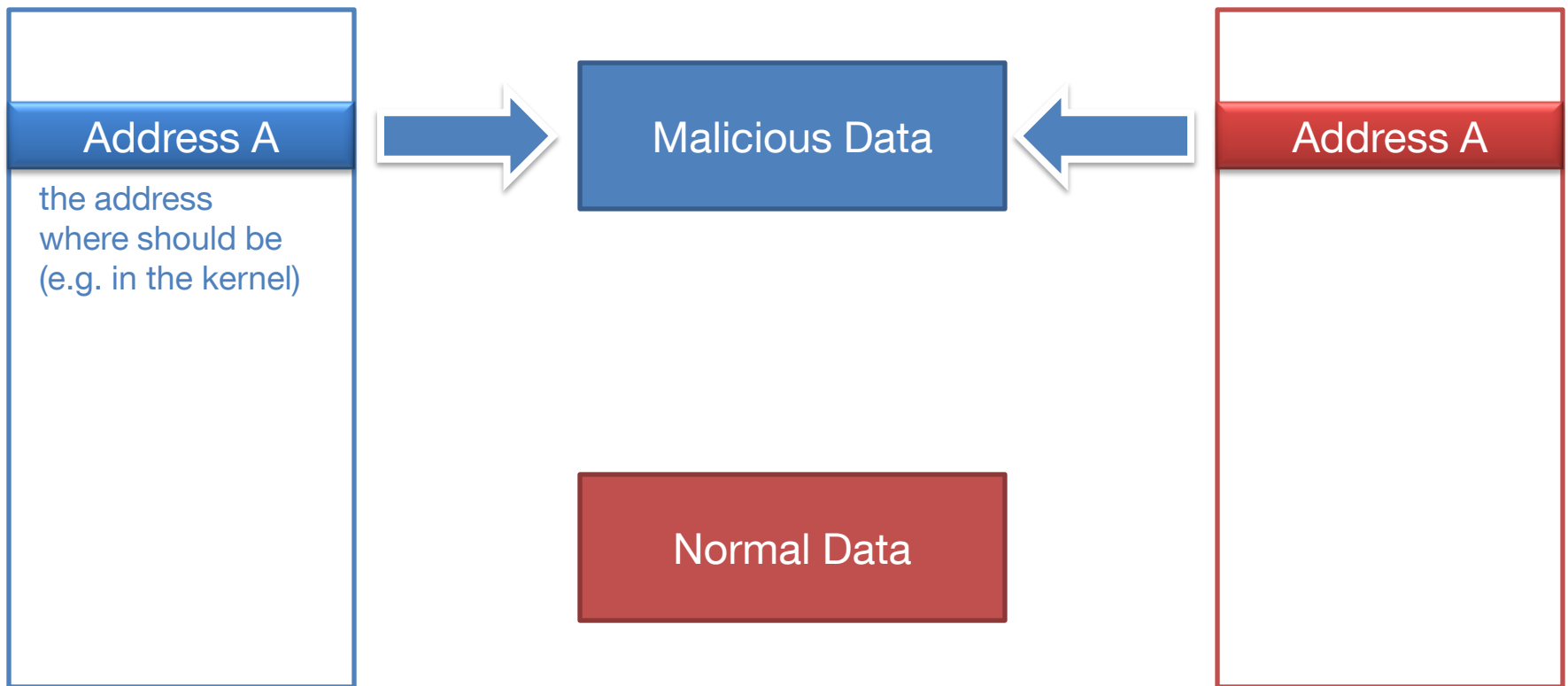


“MALICIOUS” Shadow Paging (2)

- Watch Page Table write
- If Page Table is written, check if...
 - Physical Address is malicious (modified) data/code
 - Corresponding Linear Address is NOT a address where should be
- If these two conditions are fulfilled, rootkit alter this Page Table Entry mapping

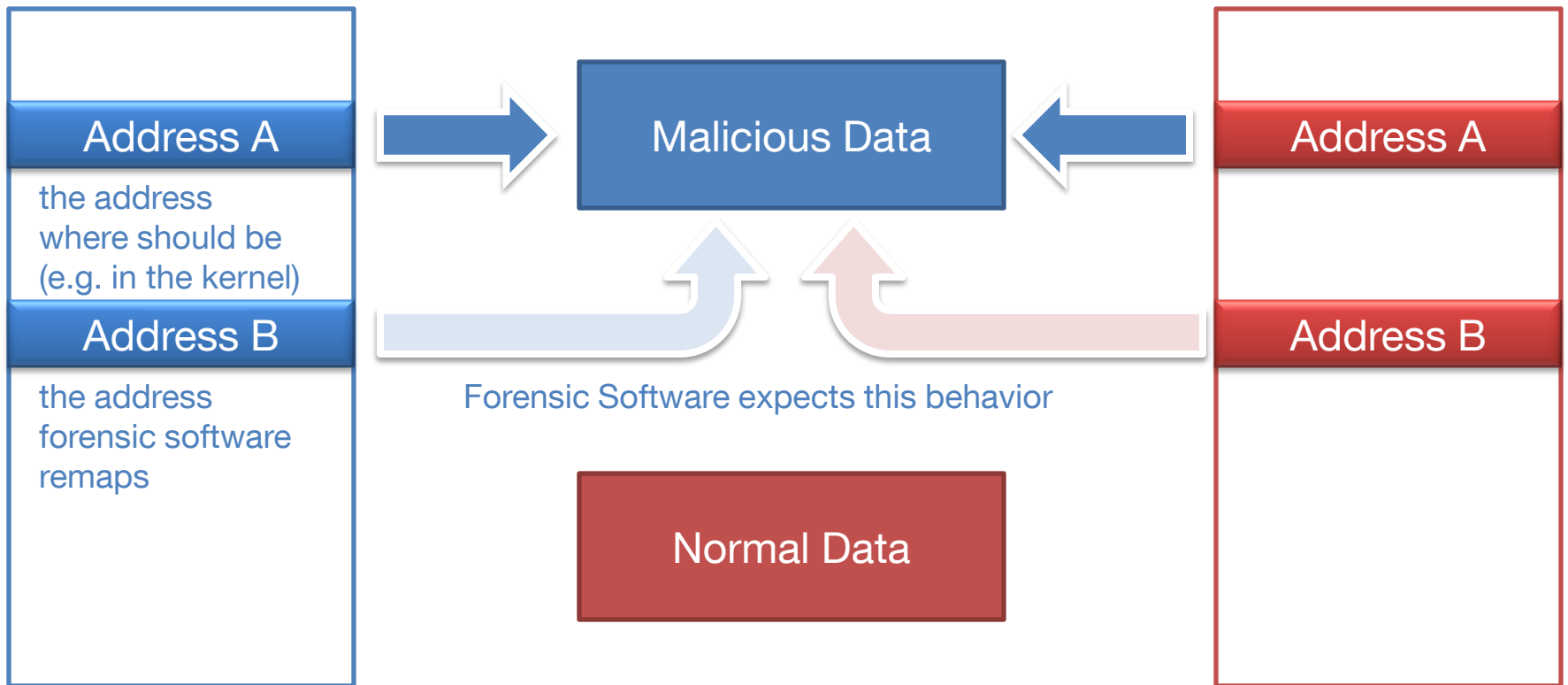


“MALICIOUS” Shadow Paging (3)



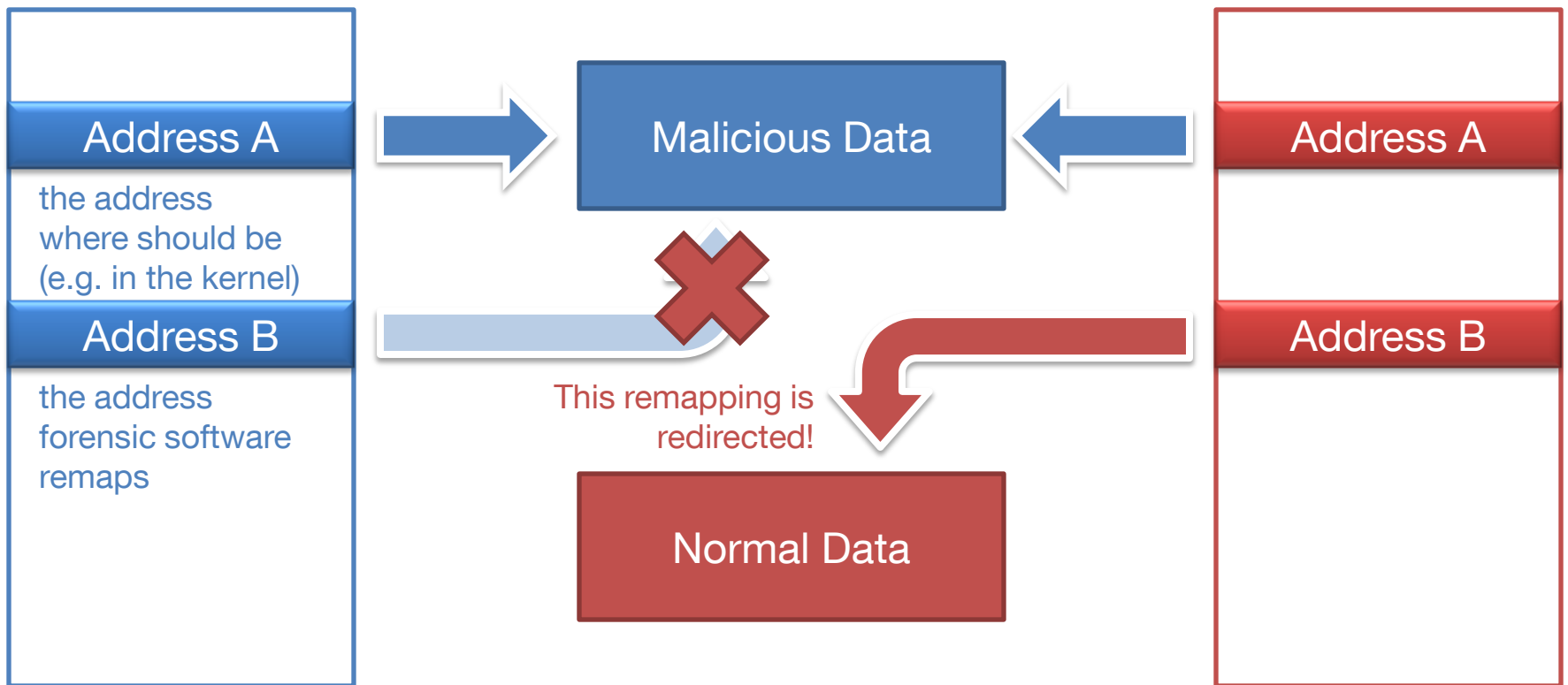


“MALICIOUS” Shadow Paging (3)





“MALICIOUS” Shadow Paging (3)





“MALICIOUS” Shadow Paging (4)

- These remapping like previous example is **WHAT FORENSIC SOFTWARE DOES**
 - When making Hibernation File and Crashdump File, Same remapping will occur
- As a result, this rootkit is invisible from almost all forensic software and Windows-made memory dump
- Rootkit can fake “nearly everything”



Weakness of Forensic Software (1)

- Doing physical memory acquisition
UNDER THE CONTEXT THAT
ROOTKIT CAN EXIST
 - Page Fault can happen on mapping
 - Trusting operating system features
(that may be tainted)
- Not only this...



Weakness of Forensic Software (2)

- Many software still use
¥Device¥PhysicalMemory!
 - Anti-forensic implementation is announced 3 years ago!
(DDefy : http://www.slideshare.net/amiable_indian/antiforensic-rootkits)
 - Many forensic software (such as EnCase, FastDump) uses this device to acquire physical memory
- This device can be “single-point-of-failure”
 - Many forensic software (not much as Shadow Paging based rootkit) can be faked if we just hook this device



Livegrid™



Fact (2):

**DON'T USE FORENSICS
FOR ROOTKIT DETECTION**



Way to Workaround (1)

- To detect rootkit, use local detection software (such as GMER, Rootkit Unhooker...)
 - Actually, this rootkit is not so difficult to detect locally.
 - Using these software, we can detect changes done by rootkit
- Don't trust forensic input when rootkit infection is possible



Way to Workaround (2)

- But this is only symptomatic way
 - Forensic input is a “dump” if machine is tainted by rootkit
- Can we improve implementation of forensic software?

...YES



Livegrid™



Anti-Anti-Forensics:

ACQUIRE CORRECT MEMORY AGAINST ROOTKITS



Livegrid™



Remember previous slides...



Previous Slide : Shadow Paging (2)

- Despite of this, Shadow Paging can be implemented if some conditions are fulfilled
 - CR0.WP bit is always set
 - All Changes of CR3 registers are done by known point of operating system
 - Page Table Handler is hooked
 - (Other conditions are abridged)
- In normal condition, most of operating system fulfills these conditions (Windows, Linux...)

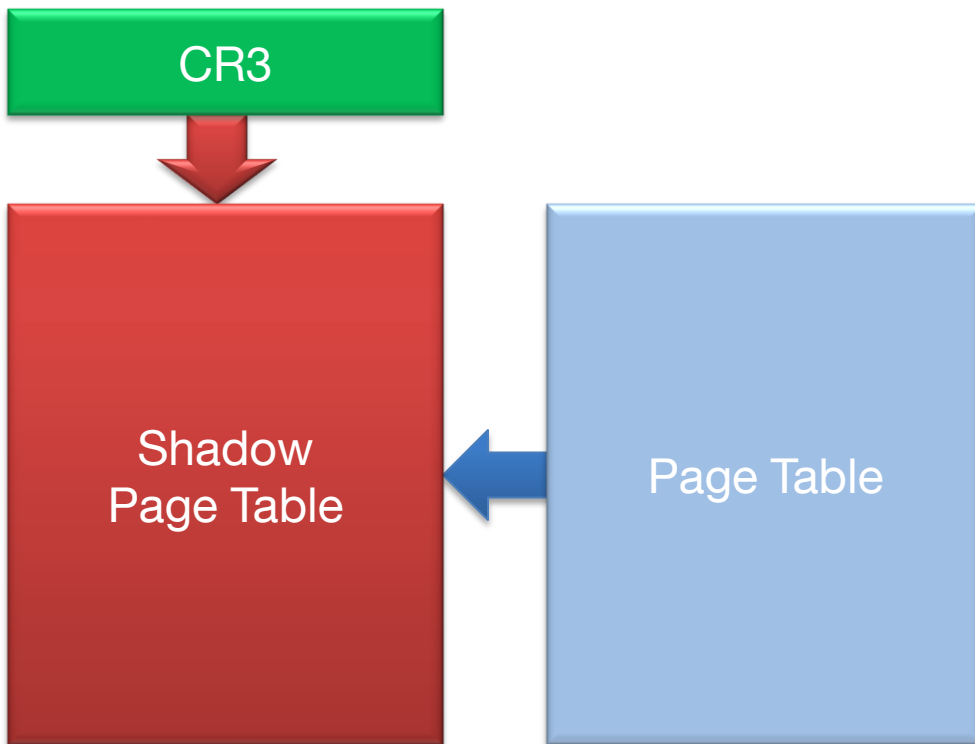


Way to Acquire correct memory contents (1)

- Destroy Context of Shadow Paging
 - Reset CR0.WP bit
 - Change CR3 register to own Page Table (CR3 is pointer to Page Table)
 - Set own Page Fault Handler
- These are easy to implement and effective

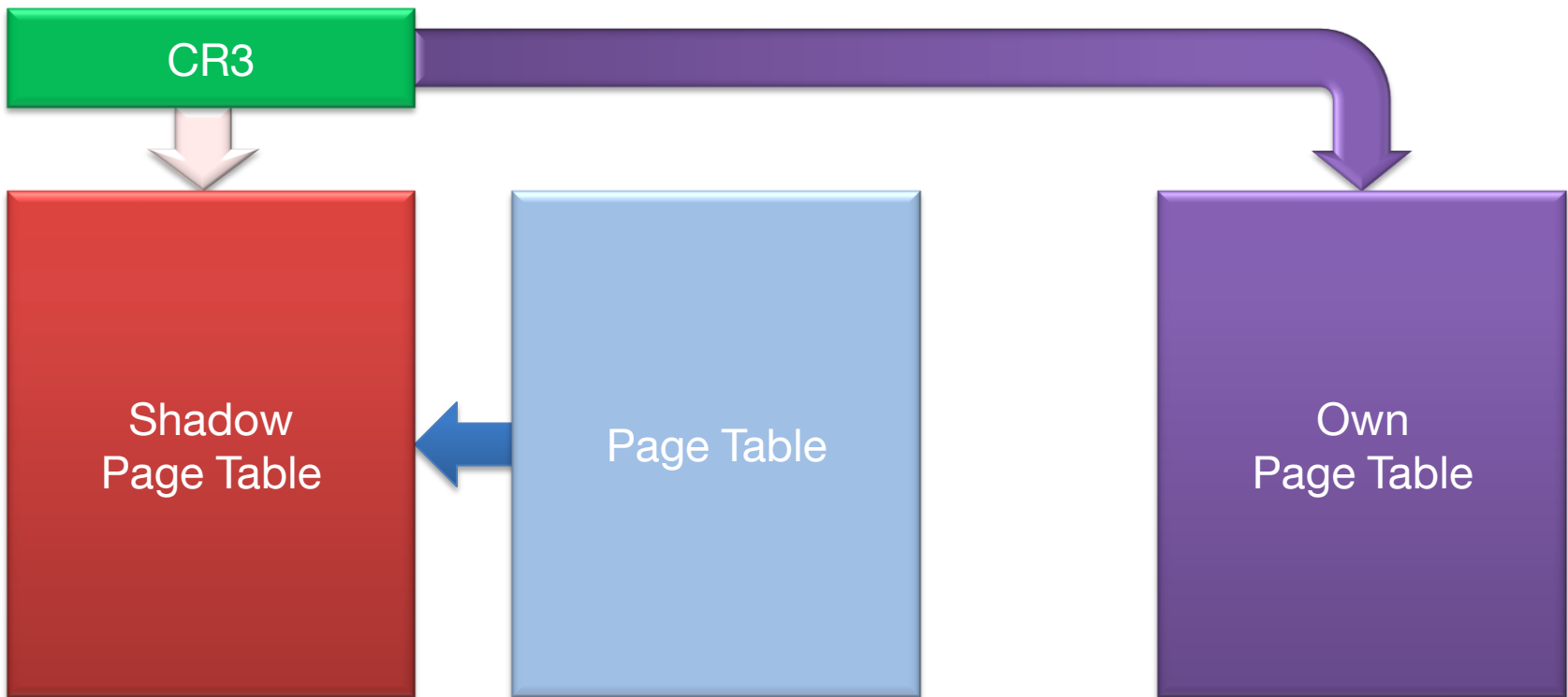


Way to Acquire correct memory contents (2)





Way to Acquire correct memory contents (2)





Way to Acquire correct memory content (3)

- To hook all interrupts (including exceptions), modify interrupt vector by LIDT instruction
- Make own Page Table (for acquisition)
- Set CR3 to own Page Table and acquire physical memory
- They are still problems, but we can acquire correct contents of memory



Problems left [But no need to deal with] (1)

- In rootkit context, there are no way to get correct Physical Address associated to Linear Address
 - In other words, there are no way to validate if own Page Table points valid memory range
 - rootkit still can prevent acquisition of memory
 - For instance, rootkit can force reset computer if forensic software tries to acquire memory



Problems left [But no need to deal with] (2)

- Even though, faking physical memory content without any doubt is nearly impossible
 - rootkit actually can reset computer but... it makes user doubt!
 - some driver compatibility is lost



Livegrid™



Also problems for these rootkits

CONSIDERATIONS OF THESE ROOTKITS



Considerations (1)

- Proof-of-Concept was built for x86+Windows, but this anti-forensic technique is cross-platform.
 - Another operating systems running on x86
 - Linux, Mac OS X...
 - x86_64 architecture
 - Other CPUs supporting paging



Considerations (2)

- This rootkit cannot be detected from Hibernation File
 - In other words, rootkit is (temporarily) removed if you hibernate the computer.
 - Reinfection code is needed to prevent that behavior, but it also raises possibilities of detection



Considerations (3)

- Implementation for SMP is very, very difficult
 - State of Page Table can be “unstable”
therefore many code is needed to handle these state
- No SMP implementation in public proof-of-concept source code



Considerations (4)

- Forensic input unaffected:
 - Improved Forensic Implementation
 - State File of Virtual Machines
 - vmem file in VMware
 - Acquisition using Hardware
 - 1394memimage (FireWire / IEEE1394)



Livegrid™



About all of this:

CONCLUSION



Conclusion

- Current live memory forensics is “a broken concept”
- If you want to detect rootkits, use local rootkit detection software (GMER, Rootkit Unhooker...) even you use live memory forensics
- There are ways to counter with rootkits
- It is not time sensitive, but future forensic implementation must be updated



Livegrid™



Have any questions?

THANK YOU.

Tsukasa Ooi <li@livegrid.org>
Livegrid Incorporated, Lead Analyst



Technical Articles and Sources

- ... will be available November, 2009
- at <http://a4lg.com/>