Dynamic Analysis of Evasive Modular Malware

Shabnam Aboughadareh[1], Christoph Csallner[1], Mehdi Azarmi[2]

[1]University of Texas at Arlington

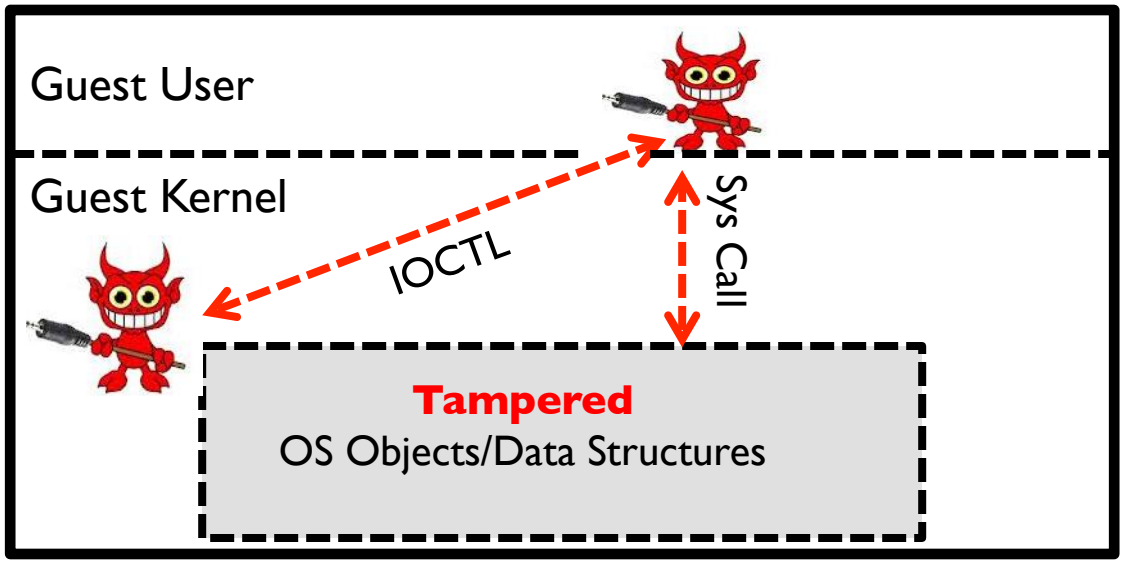
[2]Purdue University

ACSAC 2012

Orlando, Florida

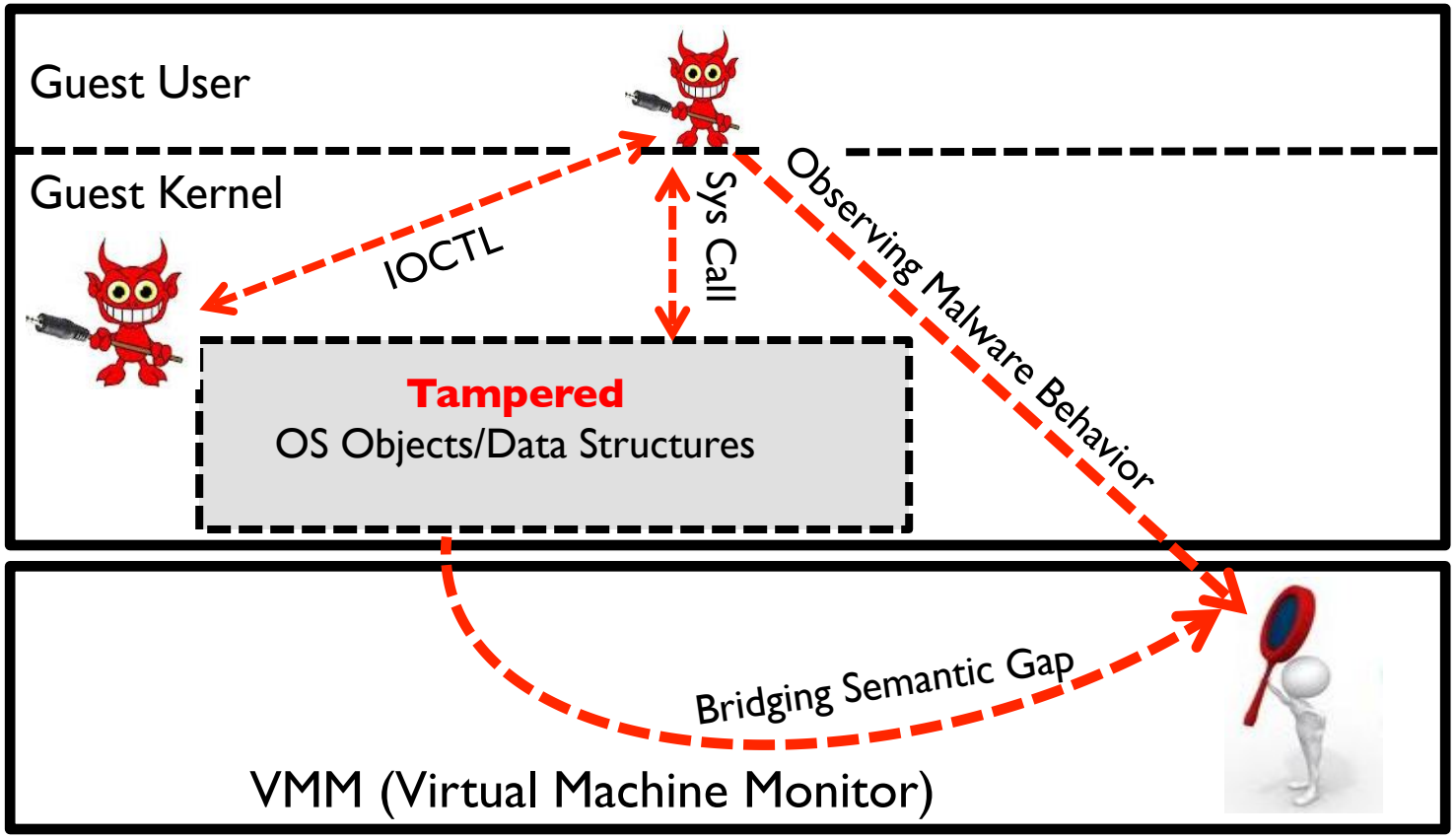
Evasive Modular Malware

- User-mode malware that drops a kernel module (rootkit)
  - Module (rootkit) manipulates OS objects and kernel data structures
- User-mode malware only exposes its malicious behavior **after** the rootkit has successfully manipulated the kernel
- To analyze its behavior, we have to allow evasive modular malware to manipulate kernel data



Evasive Modular Malware Makes Analysis/Tracing Difficult

- Analysis/tracing infrastructure typically assumes integrity of some kernel data structures
- But rootkit may manipulate kernel data that are used by analysis or tracing infrastructure



UNIVERSITY OF TEXAS ARLINGTON

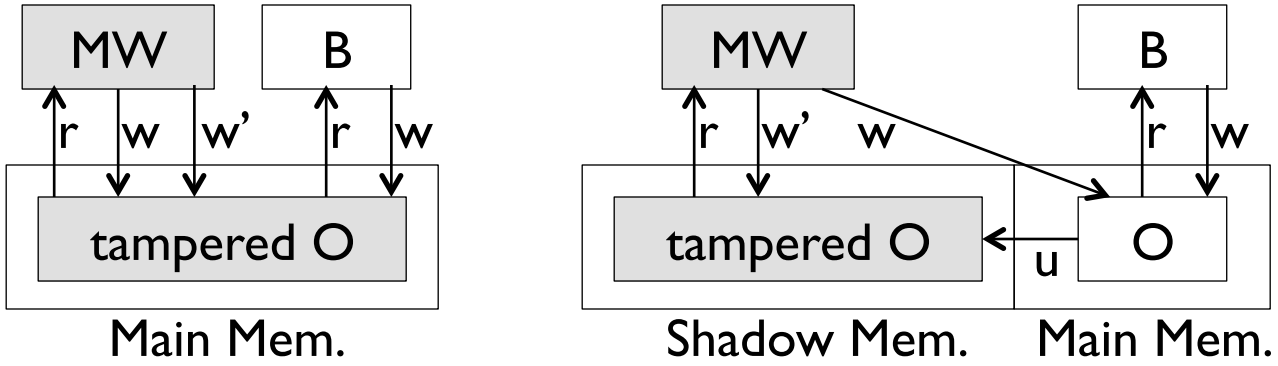
PURDUE UNIVERSITY

Problem Statement

- The assumption of "Kernel Data Integrity" in bridging the semantic gap is not valid/realistic.
- Recent semantic gap solutions in handling DKOM/DKSM attacks are limited. (Virtuoso [SP'11], Space Traveling Across VM [SP'12])
- Analyzing behaviors of evasive modular malware is not possible with current techniques (i.e., NICKLE [RAID'08])
  - They prevent the execution of malware
- Evasive modular malware can compromise different VM-based malware analysis approaches (i.e., Ether [CCS'08], and Temu [VEE'10]).

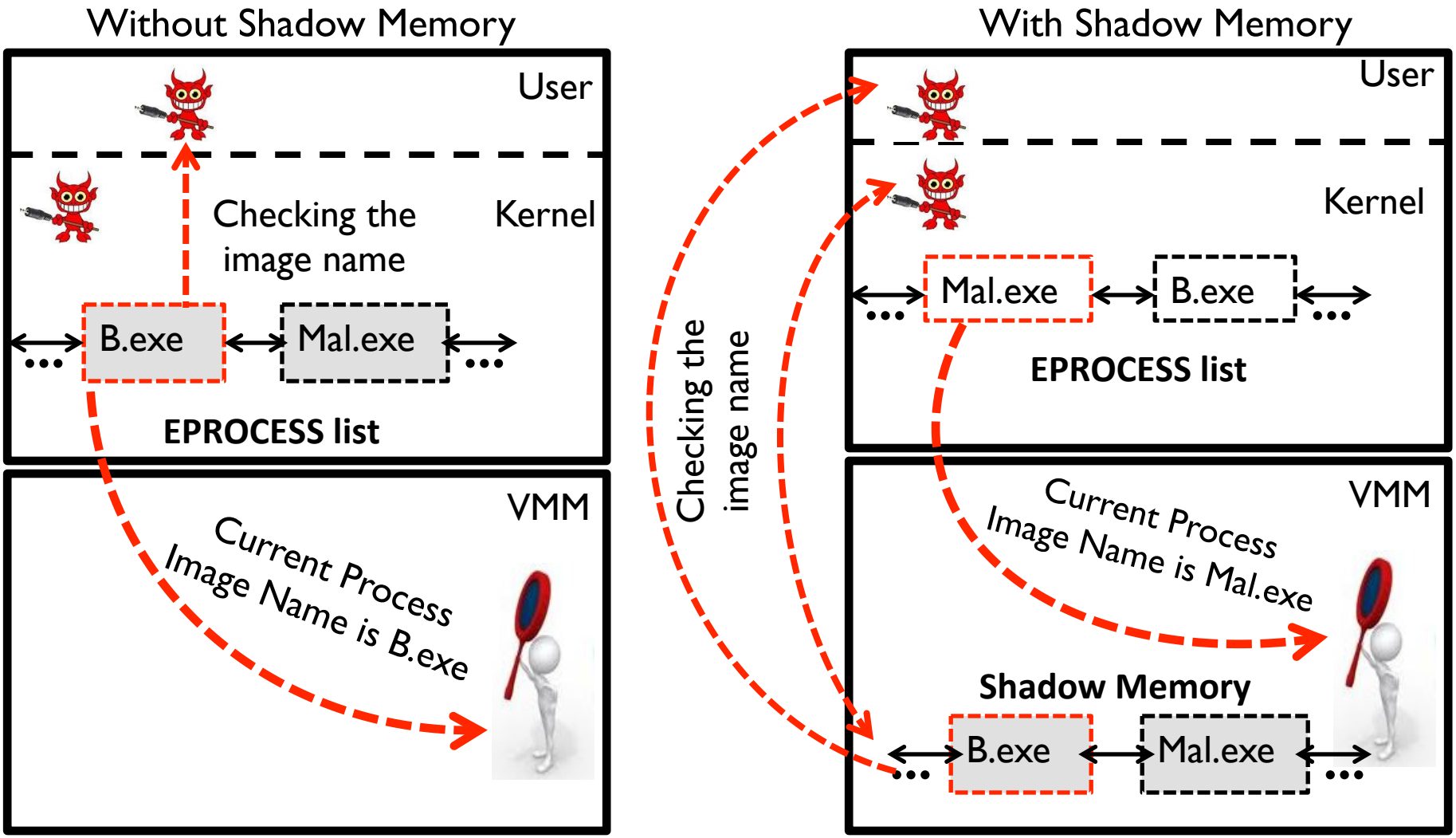
Proposed Approach: Shadow Memory

- Maintain copy of kernel objects / data structures **O** in shadow memory
  - Analyst can customize specific set of kernel objects / data structures
- Redirect reads/writes of malware to shadow memory
- Rootkit can manipulate kernel data without breaking malware analysis



MW = malware in user-mode or kernel-mode
B = benign code in user-mode or kernel-mode
O = kernel object ; r = read; w = authorized write; w' = unauthorized write
u = update the shadow memory with authorized changes in O

Example: DKSM attack in Windows (Direct Kernel Structure Manipulation)



Current malware analysis systems, e.g.: TEMU, Ether

UNIVERSITY OF TEXAS ARLINGTON

PURDUE UNIVERSITY

Initial Results

- DKOM (hiding process/driver objects)
- DKSM (Modifying PID/Image name in EPROCESS)
- Function Pointer Manipulation in Driver Object (ZeroAccess Rootkit)
- Overhead of current prototype is currently within an order of magnitude
- We expect the overhead to become competitive with TEMU (implementation needs more work)

UNIVERSITY OF TEXAS ARLINGTON

PURDUE UNIVERSITY

References

[CCS'08] Dinaburg, Artem, et al. "Ether: malware analysis via hardware virtualization extensions." *Proceedings of the 15th ACM conference on Computer and communications security (CCS)*. ACM, 2008.

[DKSM'10] Bahram, Sina, et al. "DKSM: Subverting virtual machine introspection for fun and profit." *Reliable Distributed Systems, 2010 29th IEEE Symposium on*. IEEE, 2010.

[RAID'08] Riley, Ryan, Xuxian Jiang, and Dongyan Xu. "Guest-transparent prevention of kernel rootkits with vmm-based memory shadowing." *Recent Advances in Intrusion Detection (RAID)*. Springer Berlin/Heidelberg, 2008.

[SP'11] Dolan-Gavitt, B., Leek, T., Zhivich, M., Giffin, J., & Lee, W. (2011, May). Virtuoso: Narrowing the semantic gap in virtual machine introspection. In *Security and Privacy (SP), 2011 IEEE Symposium on* (pp. 297-312). IEEE.

[SP'12] Fu, Yangchun, and Zhiqiang Lin. "Space Traveling across VM: Automatically Bridging the Semantic Gap in Virtual Machine Introspection via Online Kernel Data Redirection." *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012.

[VEE'10] Yin, Heng, and Dawn Song. "Temu: Binary code analysis via whole-system layered annotative execution." *Submitted to:VEE* 10 (2010).

UNIVERSITY OF TEXAS ARLINGTON

PURDUE UNIVERSITY

Backup

- Using ReactOS for bridging the semantic gap
  - Object-level reverse engineering of Windows OS memory
  - Extracting physical addresses belonging to OS objects, data structures and their function pointers