Characterizing Kernel Malware Behavior with Kernel Data Access Patterns

Junghwan Rhee, Zhiqiang Lin, Dongyan Xu
Department of Computer Science and CERIAS
Purdue University

Operating System Security

Linux kernel vulnerability coughs up superuser rights

By Ryan Naraine | October 21, 2010, 7:06am PDT

Summary

The open-source Linux operating system contains a serious

The open-source Linux operating s
serious security flaw that can be ex
superuser rights on a target syster

The vulnerability, in the Linux imple

New Windows zero-day flaw bypasses UAC

Hi there! If you're new here, you might want to subscribe to the RSS feed for updates.

November 25, 2010 | Comments (25)

in Microsoft Windows was disclosed today. The

Windows kernel vulnerability adds to Microsoft's woes

Hi there! If you're new here, you migh
updates.

by Paul Ducklin on January 21, 2010 | Be the

FILED UNDER: Vulnerability

Microsoft are under the pump
fighting vulnerabilities at
moment. Just six-and-h
after blogging that the C
Aurora Internet Explorer
be ready the next day, t
blogged about a publish

New Windows kernel mode flaw points to future attack vectors

By Peter Bright | Last updated 6 months ago

A new Windows flaw that allows all current, supported versions of Windows to be crashed was published on Friday by Israeli researcher Gil Dabah. The bug allows a local user to cause a system to suffer a blue-screen of death crash. In principle, this may also allow attackers to run code of their choosing with kernel privileges,

the flaw.

s many key Windows features like
ment's handling of the system
stem can be made to corrupt the
question did not run in kernel mode;
phics substantially faster.

News

Microsoft says rootkit caused Windows blue screens

Users may have to reinstall Windows to eradicate the malware

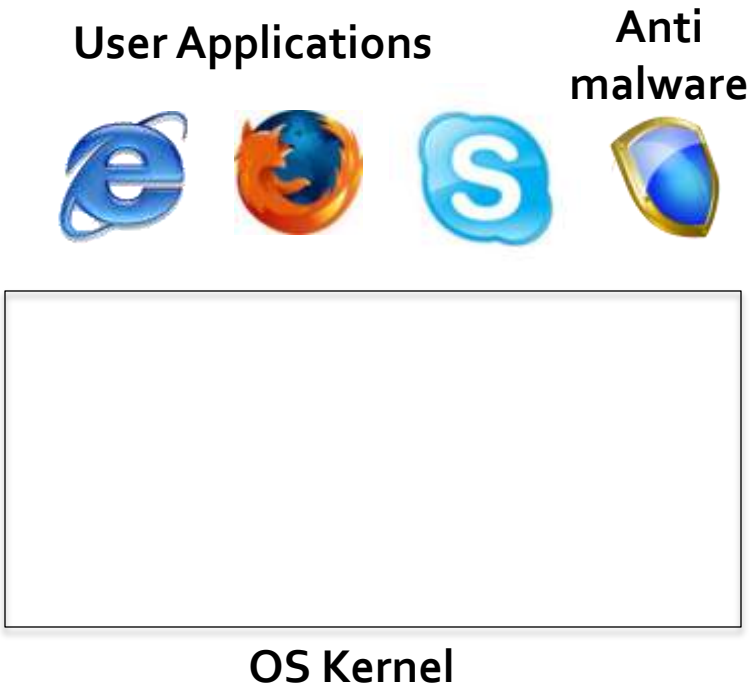
By Gregg Keizer

February 18, 2010 06:49 AM ET

Comments (4) Recommended (25) Like

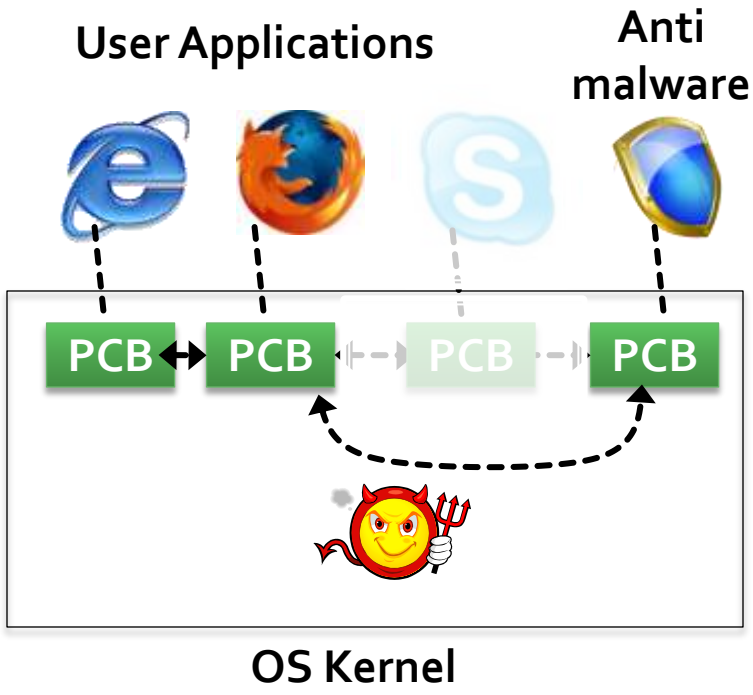
Kernel-Level Malware

- Kernel rootkits
  - Goals: Hide malicious activities
  - Create backdoors or hidden services
  - Attack vectors: modules, direct access to kernel memory



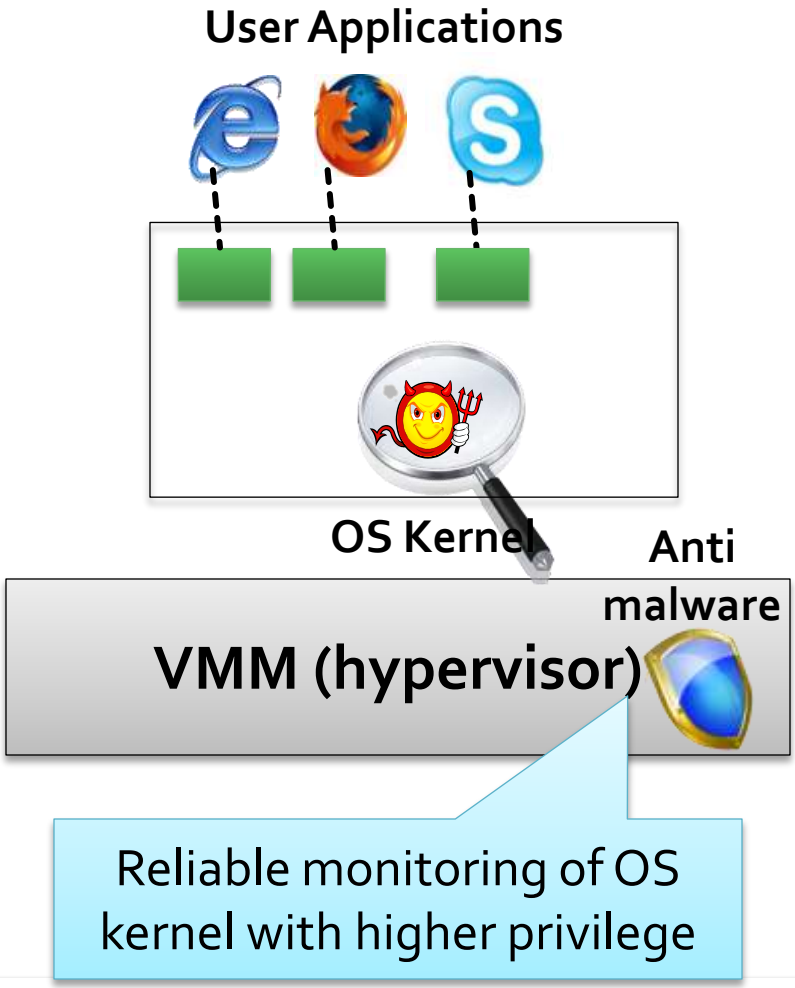
Kernel-Level Malware

- Kernel rootkits
  - Goals: Hide malicious activities
  - Create backdoors or hidden services
  - Attack vectors: modules, direct access to kernel memory



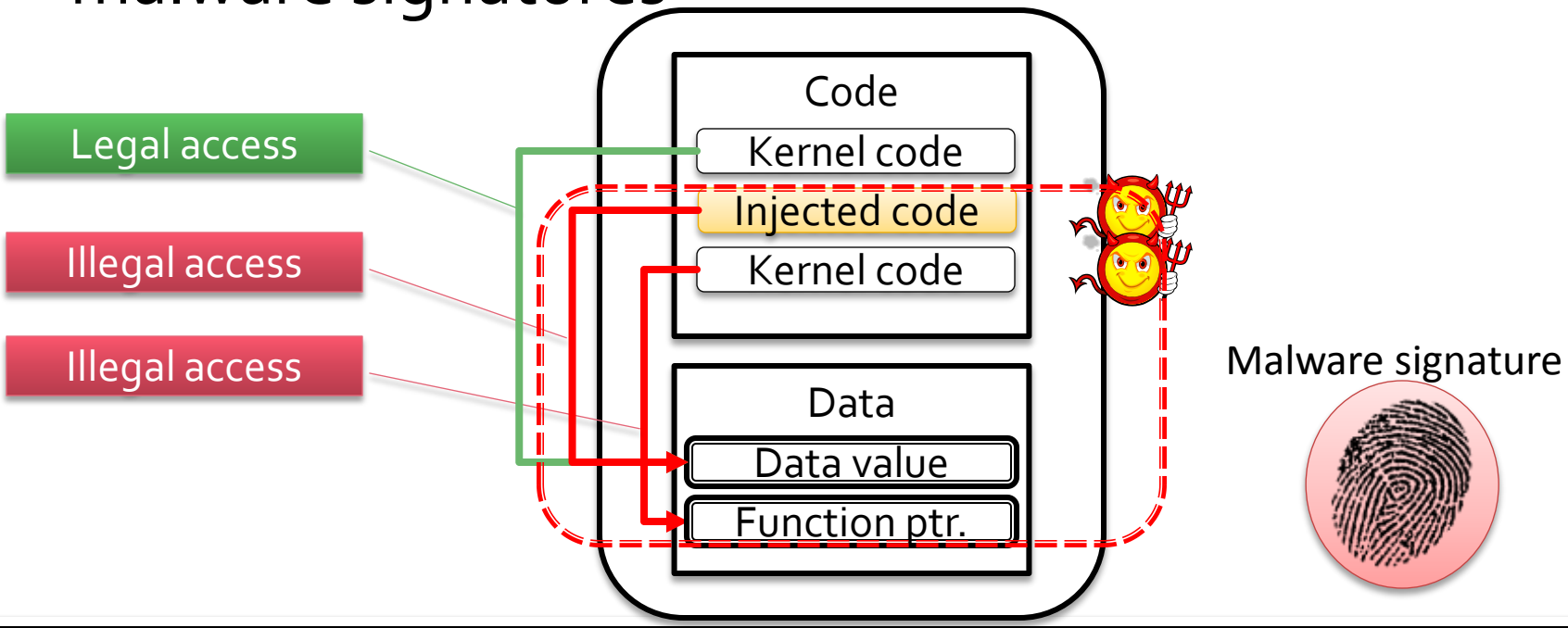
Kernel-Level Malware

- Kernel rootkits
  - Goals: Hide malicious activities
  - Create backdoors or hidden services
  - Attack vectors: modules, direct access to kernel memory
- How is kernel malware detected?

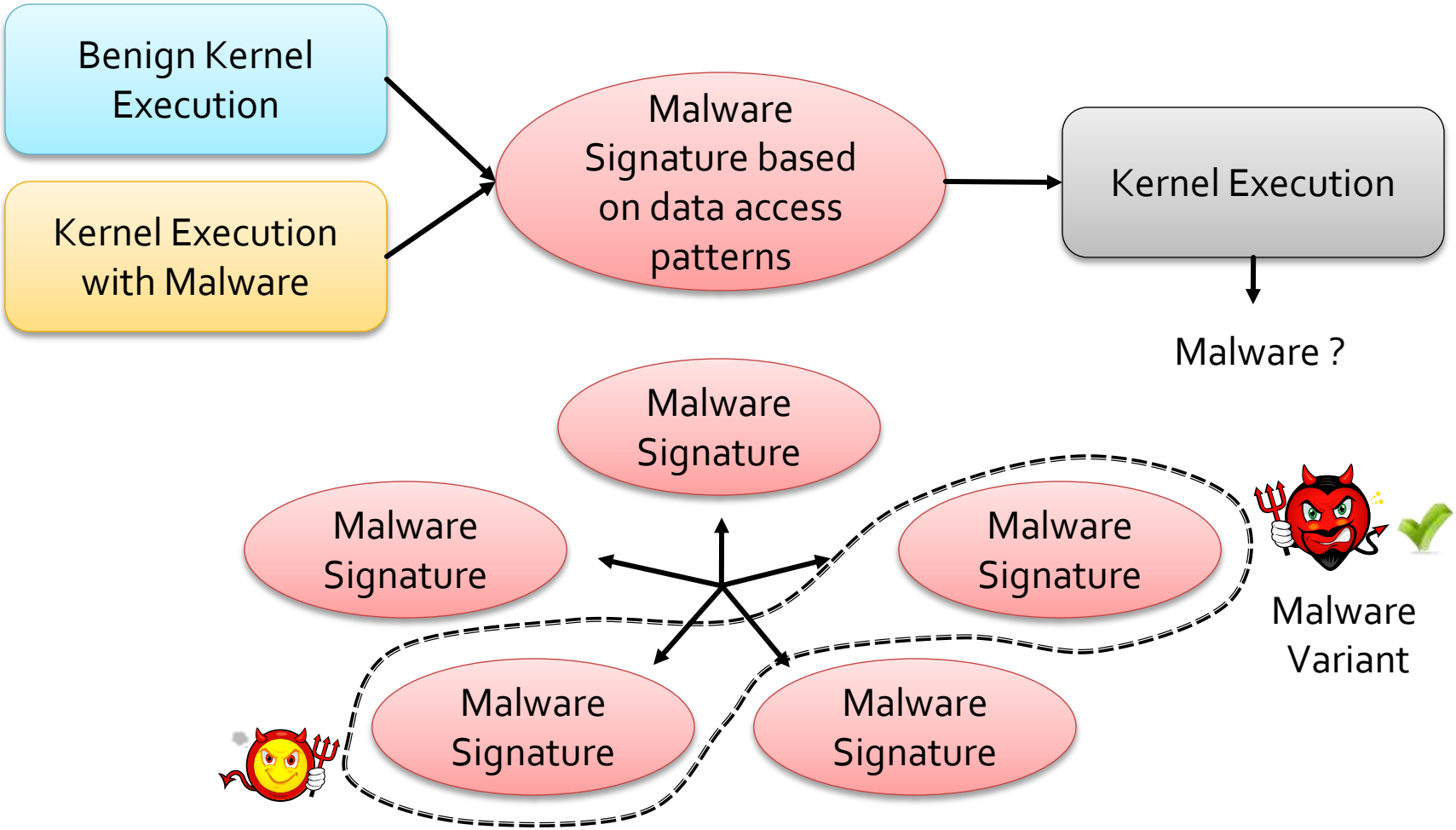


Data-centric Malware Characterization

- Many malware attacks involve kernel data accesses.
  - Kernel Rootkit Profiling [Eurosys'09], [RAID'09]
- Use unique data access patterns as kernel malware signatures

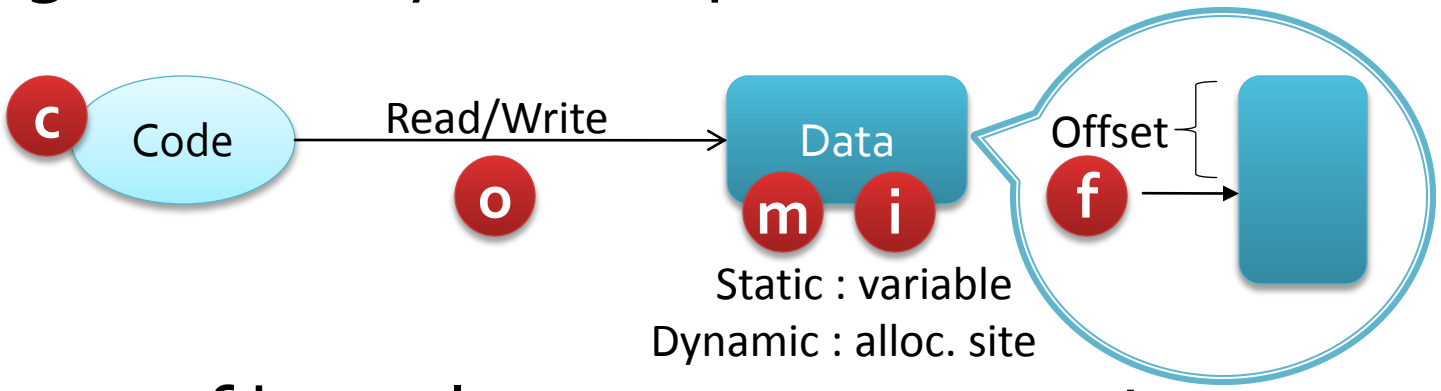


Design of DataGene

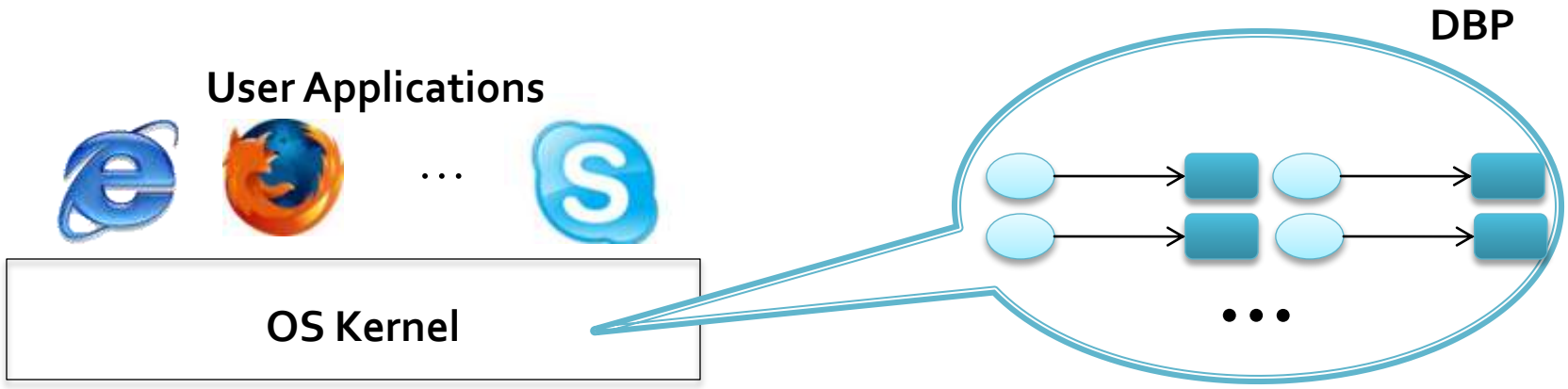


Characterizing Kernel Data Access

- Encoding a memory access pattern

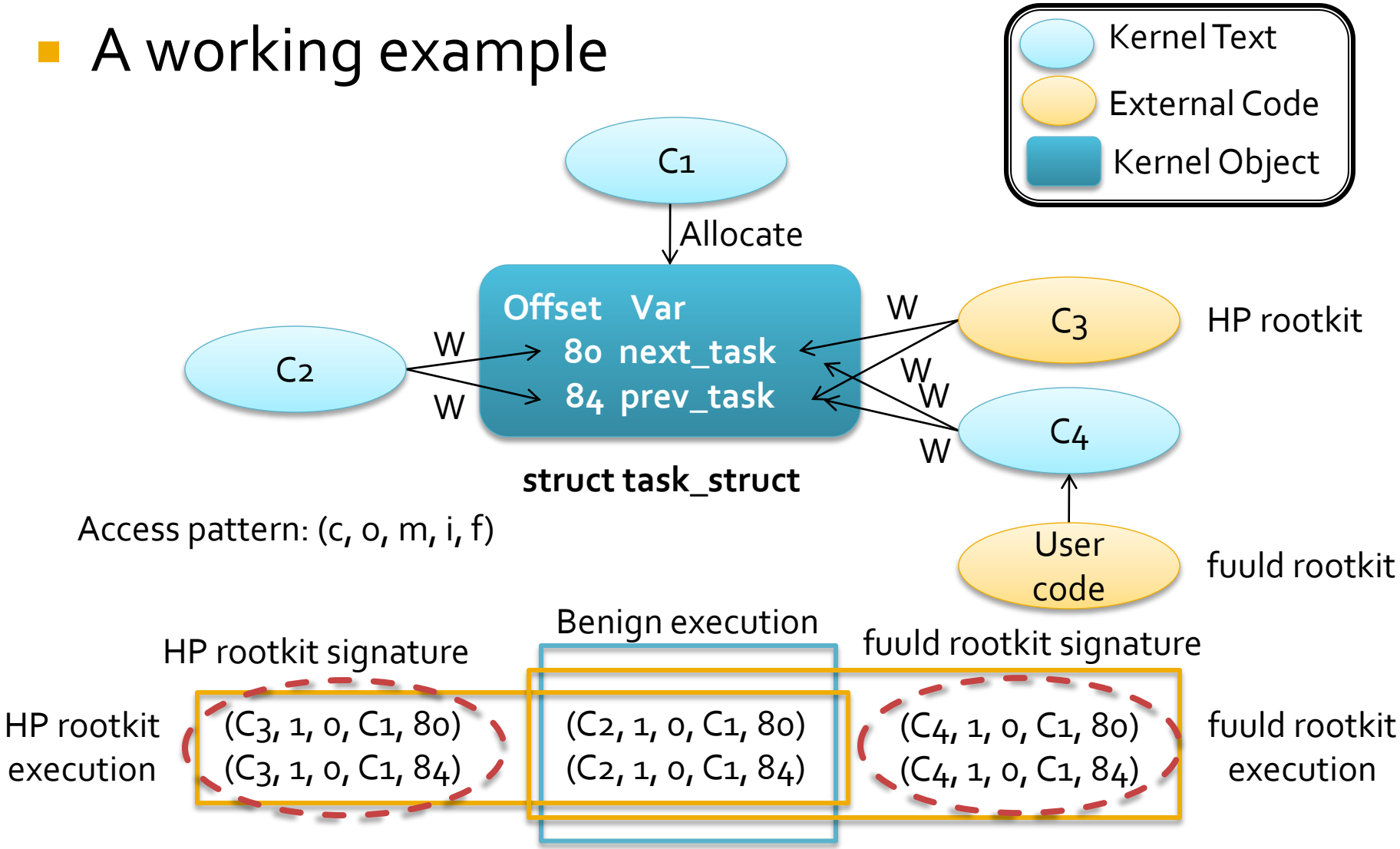


- A summary of kernel memory accesses in a kernel run (Data Behavior Profile-DBP)



Characterizing Kernel Data Access

- A working example



Characterizing Malware Behavior

- Deriving unique malware data access patterns
  - A set of accesses that consistently appear in malicious runs (D_M) and never appear in benign runs (D_B)

$$S_M = \bigcap_{j \in [1,n]} D_{M,j} - \bigcup_{k \in [1,m]} D_{B,k}$$

HP rootkit Run 1	(C3, 1, 0, C1, 80) (C3, 1, 0, C1, 84)	(C2, 1, 0, C1, 80) (C2, 1, 0, C1, 84)	Benign Run 1	(C2, 1, 0, C1, 84)
HP rootkit Run 2	(C3, 1, 0, C1, 80) (C3, 1, 0, C1, 84)	(C2, 1, 0, C1, 84)	Benign Run 2	(C2, 1, 0, C1, 80)
HP rootkit Run 3	(C3, 1, 0, C1, 80) (C3, 1, 0, C1, 84)	(C2, 1, 0, C1, 80) (C2, 1, 0, C1, 84)	Benign Run 3	(C2, 1, 0, C1, 80) (C2, 1, 0, C1, 84)

Characterizing Malware Behavior

- Deriving unique malware data access patterns
  - A set of accesses that consistently appear in malicious runs (D_M) and never appear in benign runs (D_B)

$$S_M = \bigcap_{j \in [1,n]} D_{M,j} - \bigcup_{k \in [1,m]} D_{B,k}$$

$\cap D_{M,j}$

(C3, 1, 0, C1, 80)
(C3, 1, 0, C1, 84)
(C2, 1, 0, C1, 84)

$-$ $\cup D_{B,k}$

(C2, 1, 0, C1, 80)
(C2, 1, 0, C1, 84)

Characterizing Malware Behavior

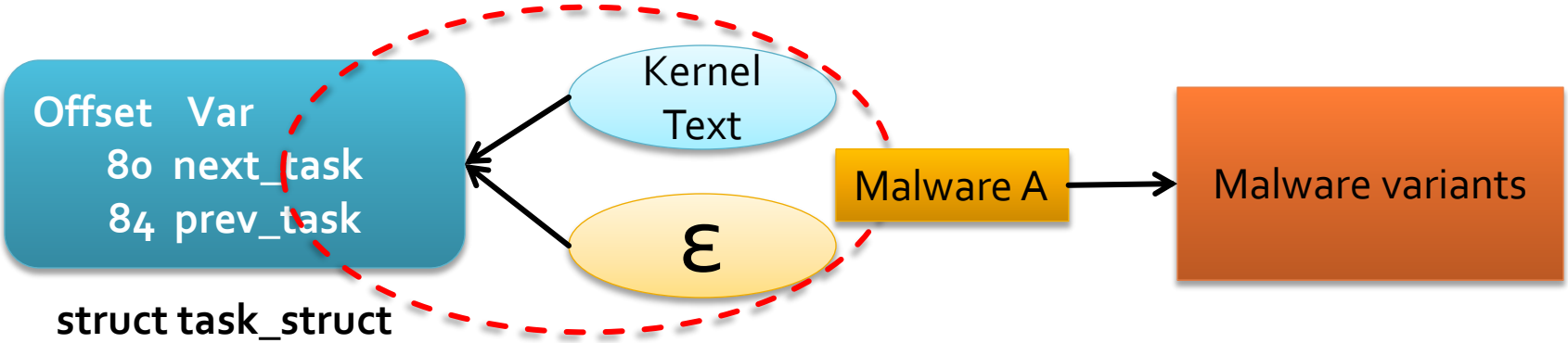
- Deriving unique malware data access patterns
  - A set of accesses that consistently appear in malicious runs (D_M) and never appear in benign runs (D_B)

$$S_M = \bigcap_{j \in [1,n]} D_{M,j} - \bigcup_{k \in [1,m]} D_{B,k}$$

S_M (C3, 1, 0, C1, 80)
(C3, 1, 0, C1, 84)

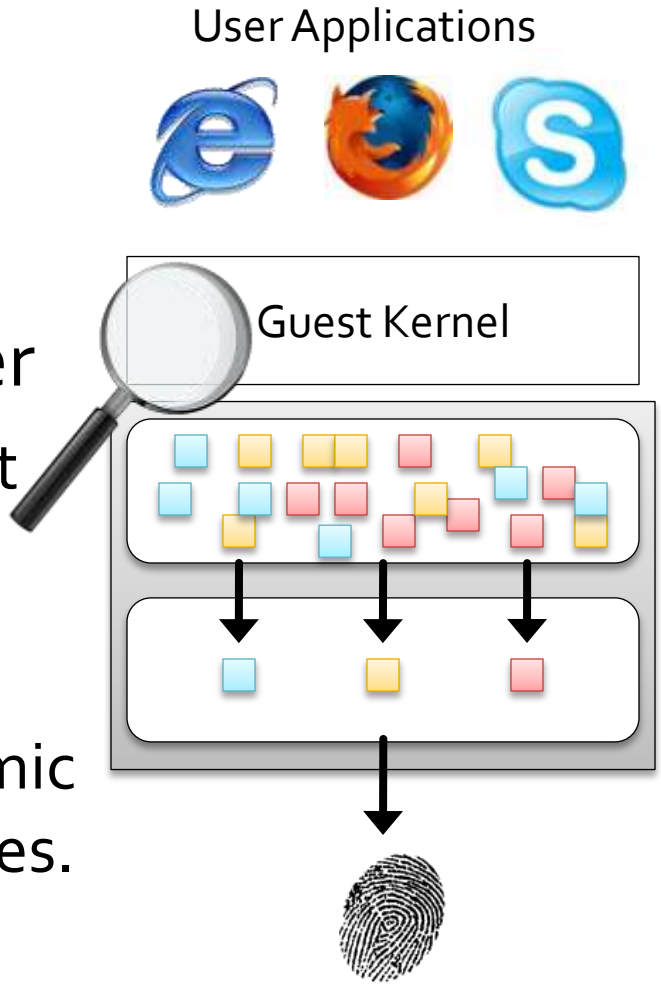
Characterizing Malware Behavior

- Generalizing malware code identity
  - DataGene disregards specific code information of kernel drivers.
  - It can match malware variants.



Implementation

- Virtual machine: QEMU+KQEMU
  - Host: 3.2Ghz, Pentium 4, 2GB Memory
  - Guest: Redhat 8, 256MB Ram
- Runtime kernel memory mapper
  - Identifies kernel data structures at runtime
- Data aggregator
  - Summarize the accesses to dynamic data instances with their alloc. sites.



Deriving Malware Signatures

- Benign kernel execution
  - Kernel compile, ssh, scp, lsmod, ps, top, find, etc.
  - Some workloads for several hours
- Malware execution
  - Classic rootkits : adore 0.38, SucKIT, modhide
- Kernel execution samples
  - In our experiments, 5 sets of malware and benign runs provided enough information to extract unique malware behavior without false positives.
  - More number of runs will improve signature quality.

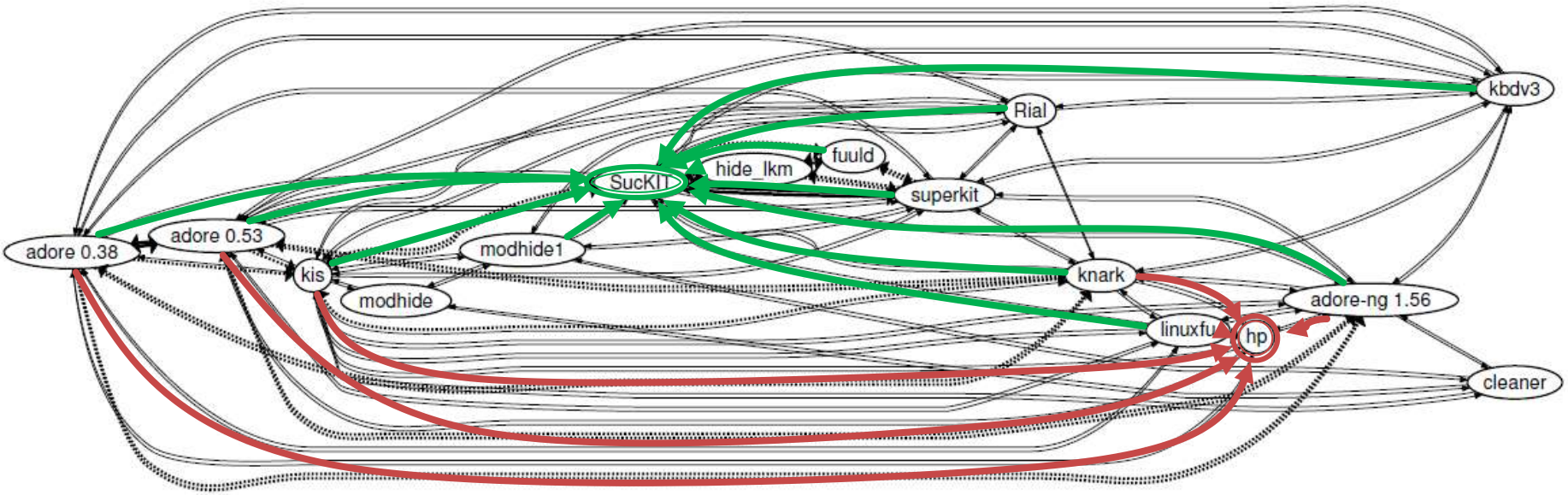
Overlapping Malware Signatures

Signature		Kernel rootkit execution														
Rootkit Name	Signature Size	Adore 0.38	Adore 0.53	Adore-ng 1.56	fuuld	Hide_lkm	SucKIT	superkit	hp	kbdv3	knark	linuxfu	Rial	cleaner	kis	modhide	modhide1
Adore 0.38	45	45	37	16	0	0	10	10	3	5	20	5	7	0	17	0	0
SucKIT	14797	3	2	1	16	13	14797	14767	0	1	2	1	1	0	18	0	1
modhide	3	0	0	0	0	0	0	0	0	0	0	0	0	2	1	3	2
Detect?		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

- 3 rootkit signatures based on access patterns can detect 16 kernel rootkits

Rootkit Name	Size	Adore 0.38	Adore 0.53	Adore-ng 1.56	fuuld	hide_lkm	SucKIT	superkit	hp	kbdv3	knark	linuxfu	Rial	cleaner	kis	modhide	modhide1
Adore 0.38	45	45	37	16	0	0	10	10	3	5	20	5	7	0	17	0	0
Adore 0.53	53	37	53	26	0	0	10	10	3	5	19	4	7	0	29	0	0
Adore-ng 1.56	106	16	26	106	0	0	1	1	2	4	9	8	0	3	6	0	0
fuuld	37	0	0	0	37	13	16	16	0	0	0	0	0	0	0	0	0
hide_lkm	4827	0	0	0	13	4827	13	13	0	0	0	0	0	0	0	0	0
SucKIT	14797	3	2	1	16	13	14797	14766	0	1	2	1	1	0	18	0	1
superkit	14783	3	2	1	16	13	14769	14783	0	1	2	0	1	0	3	0	2
hp	27	3	3	2	0	0	0	0	27	0	1	5	0	0	1	0	0
kbdv3	16	5	5	4	0	0	2	2	0	16	4	0	6	0	3	0	0
knark	73	20	19	9	0	0	10	10	1	4	73	1	4	0	19	0	0
linuxfu	46	5	4	8	0	0	1	0	14	0	1	46	0	0	1	0	0
Rial	57	6	6	0	0	0	5	5	0	2	4	0	57	0	10	0	2
cleaner	5	0	0	3	0	0	0	0	0	0	0	0	0	5	1	2	2
kis	32097	3	3	4	0	0	17	2	1	1	3	1	2	1	32097	1	5
modhide	3	0	0	0	0	0	0	0	0	0	0	0	0	2	1	3	2
modhide1	8	0	0	0	0	0	1	2	0	0	0	0	2	2	5	2	8
# matched		10	10	10	3	3	12	11	6	8	10	7	8	4	13	3	6

Similarity among Malware Signatures



- In the experiments with 16 rootkits, a kernel rootkit can be detected by using 3 ~ 13 other rootkit signatures (more than seven in average).

Related Work

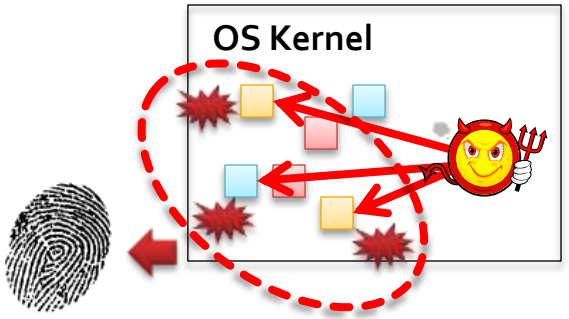
- Malware defense based on code integrity
  - NICKLE [RAID'08], SecVisor [SOSP'07]
- Malware defense based on code control flow
  - Bayer et al. [NDSS'10], Christodorescu et al. [Security'03], Balzarotti et al. [NDSS'10], etc.
- Kernel memory mapping
  - SBCFI [CCS'07], Gibraltar [ACSAC'08], KOP [CCS'09], PoKeR [Eurosys'09], rkprofiler [RAID'09]
- Inter-relation between code and data
  - Bratus et al. [TRUST'10]

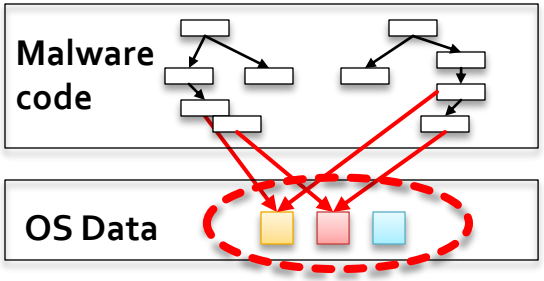
Discussions and Limitations

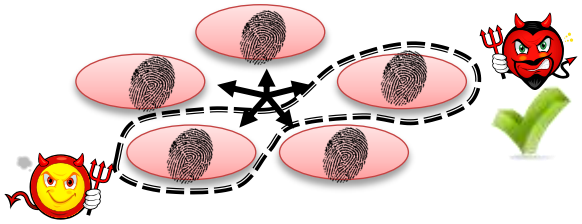
- Malware with no overlapping targets
- Potential false positives
- Mainly for non-production environments
  - Honeypot, kernel malware analysis
  - Aftersight [USENIX'08] can be leveraged for inspecting production runs.

Conclusion

- DataGene derives data access patterns unique to malware and use them to generate malware signatures.
- Data access patterns do not depend on temporal control flow of malware.
- Data access patterns can effectively match malware variants with common data targets.







Thank you, Questions?

For more information

rhee@cs.purdue.edu