Persistent Data-Only Malware

Function Hooks Without Code

Sebastian Vogl, Jonas Pfoh, Thomas Kittel, and Claudia Eckert
{vogls,pfoh,kittel,eckert}@sec.in.tum.de

Technische Universität München
Chair for IT-Security
Munich, Germany

26.02.2014

Outline

Protection Mechanisms

Motivation & Background

Protection Mechanisms

- $W \oplus X$

Motivation & Background

▸ Problem

Protection Mechanisms

- $W \oplus X$
- Signed Driver Loading

Motivation & Background

Protection Mechanisms

- $W \oplus X$
- Signed Driver Loading
- Secure Boot

Motivation & Background

Protection Mechanisms

- $W \oplus X$
- Signed Driver Loading
- Secure Boot
- Code Integrity Checking

Motivation & Background

▸ Problem

Protection Mechanisms

- $W \oplus X$
- Signed Driver Loading
- Secure Boot
- Code Integrity Checking

$\Rightarrow$ **It is getting more and more difficult to introduce malicious code into the system.**

Data-only Malware

Motivation & Background

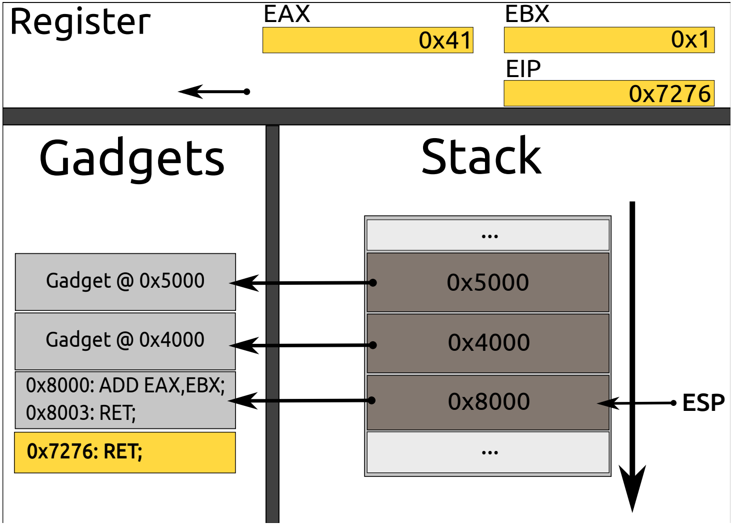
▸ Solution

Data-only Malware

Hund et. al [1]: Return-oriented Rootkits (2009)

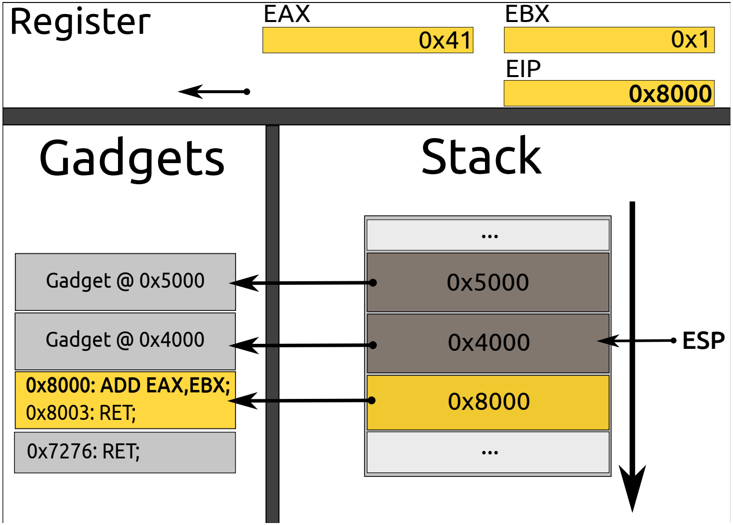
Motivation & Background

▸ Return-oriented Programming (ROP)



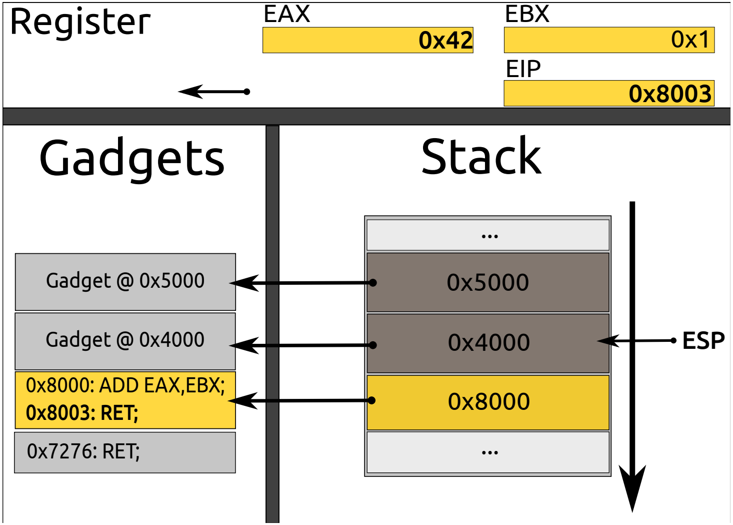
Motivation & Background

▸ Return-oriented Programming (ROP)



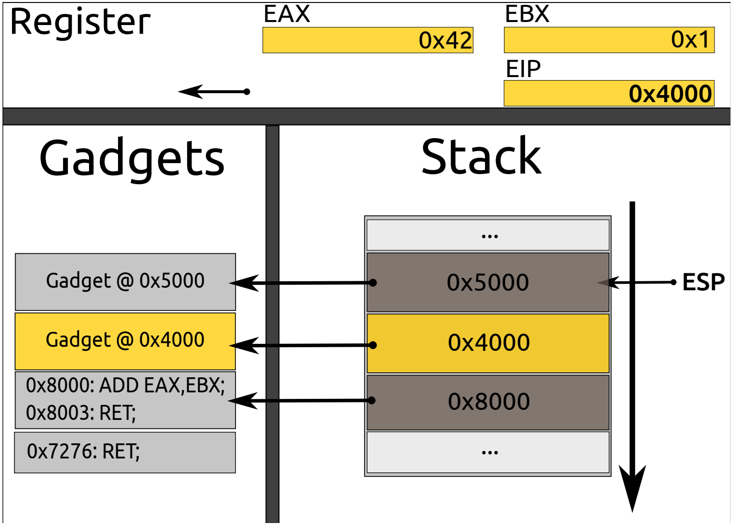
Motivation & Background

▸ Return-oriented Programming (ROP)



Motivation & Background

▸ Return-oriented Programming (ROP)



Data-only Malware

Hund et. al [1]: Return-oriented Rootkits (2009)

Data-only Malware

Hund et. al [1]: Return-oriented Rootkits (2009)

- “One-Shot-Attacks”

Data-only Malware

Hund et. al [1]: Return-oriented Rootkits (2009)

- "One-Shot-Attacks"
- **Triggered** by attacker

Motivation & Background

- Cannot **intercept events** within the system

Motivation & Background

▸ Persistence

- Cannot **intercept events** within the system
- Usually accomplished with **hooks**

Motivation & Background

- Cannot **intercept events** within the system
- Usually accomplished with **hooks**
- **Where** should these hooks point to?

Motivation & Background

- Cannot **intercept events** within the system
- Usually accomplished with **hooks**
- **Where** should these hooks point to?

Persistence?

Outline

Persistent data-only malware

▸ Challenges

1. Finding a suitable **location** for the **persistent** control structure

Persistent data-only malware

▸ Finding a suitable location for the persistent control structure

- Control structure must be **exclusively** owned by the malware

Persistent data-only malware

▸ Finding a suitable location for the persistent control structure

- Control structure must be **exclusively** owned by the malware

  ⇒ Create **new** memory region (e.g. kmalloc)

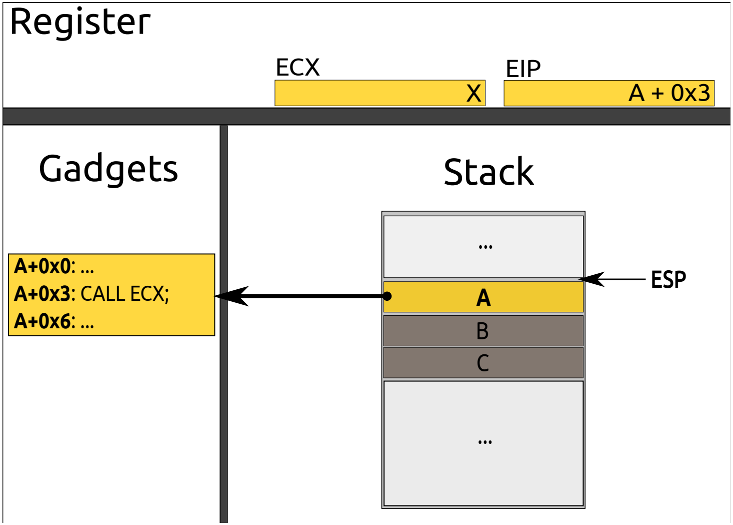
Persistent data-only malware

▸ Challenges

1. Finding a suitable **location** for the **persistent** control structure
2. Protecting against **overwrites**

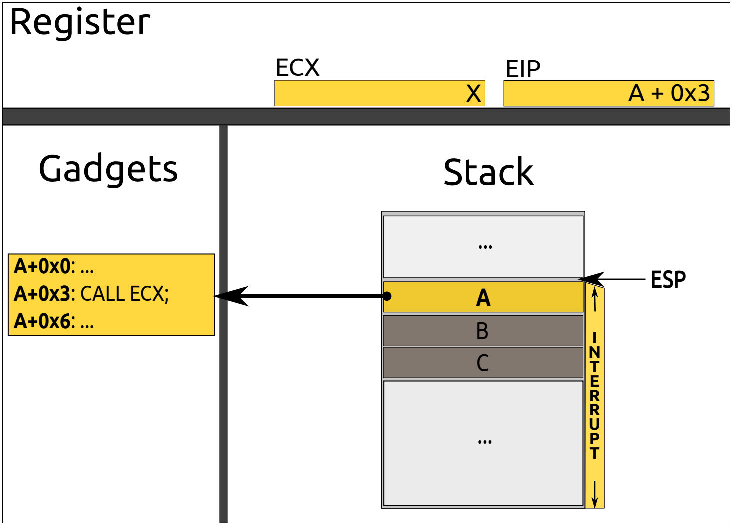
Persistent data-only malware

▸ Protecting against overwrites



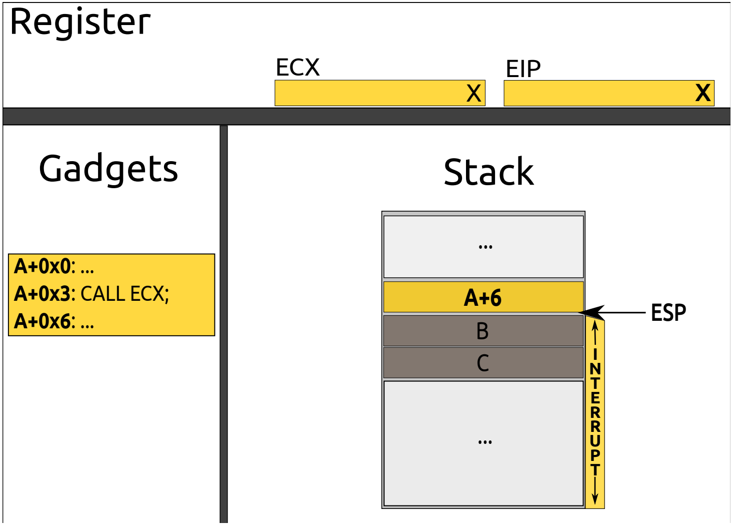
Persistent data-only malware

▸ Protecting against overwrites



Persistent data-only malware

▸ Protecting against overwrites



Persistent data-only malware

▸ Protecting against overwrites

Two types of overwrites:

- **Interrupt-induced** overwrites

Persistent data-only malware

- Protecting against overwrites

Two types of overwrites:

- **Interrupt-induced** overwrites
    - *Disable* interrupts

Persistent data-only malware

▸ Protecting against overwrites

Two types of overwrites:

- **Interrupt-induced** overwrites
  - *Disable* interrupts
  - No *external* function calls

Persistent data-only malware

▸ Protecting against overwrites

Two types of overwrites:

- **Interrupt-induced** overwrites
  - *Disable* interrupts
  - No *external* function calls
- **Self-induced** overwrites

Persistent data-only malware

▸ Protecting against overwrites

Two types of overwrites:

- **Interrupt-induced** overwrites
  - *Disable* interrupts
  - No *external* function calls
- **Self-induced** overwrites
  - *Carefully* design persistent chain

Persistent data-only malware

1. Finding a suitable **location** for the **persistent** control structure
2. Protecting against **overwrites**
3. **Resuming** the original control flow

Persistent data-only malware

▸ Resuming the original control flow

- Registers must be saved **before use**

Persistent data-only malware

- Resuming the original control flow

- Registers must be saved **before use**
- Control flow most be **restored** after execution

Persistent data-only malware

▸ Challenges

1. Finding a suitable **location** for the **persistent** control structure
2. Protecting against **overwrites**
3. **Resuming** the original control flow
4. **Activating** the *persistent* control structure

Persistent data-only malware

▸ Activating the persistent control structure

- *Control structures* **somewhere** in memory

Persistent data-only malware

- Activating the persistent control structure

- *Control structures* **somewhere** in memory
- Only control the **Instruction Pointer** when a hook is triggered

Persistent data-only malware

- Activating the persistent control structure

- *Control structures* **somewhere** in memory
- Only control the **Instruction Pointer** when a hook is triggered
- Must not use **general purpose registers** (backup!) for the switch

Persistent data-only malware

▸ Activating the persistent control structure

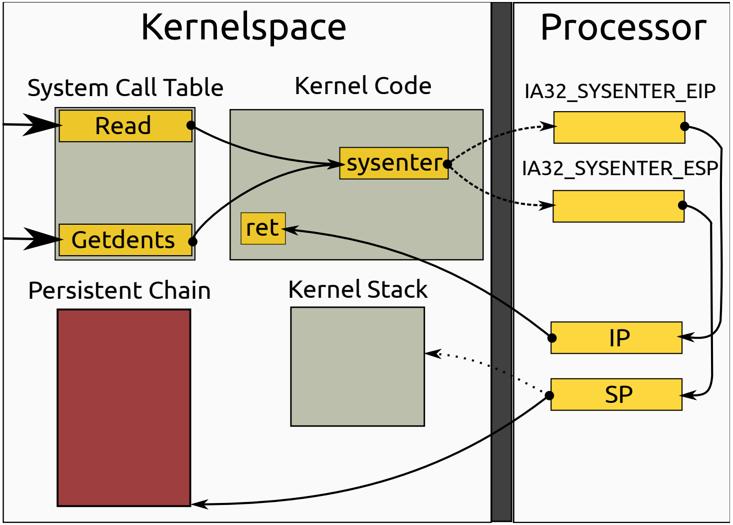
- *Control structures* **somewhere** in memory
- Only control the **Instruction Pointer** when a hook is triggered
- Must not use **general purpose registers** (backup!) for the switch

Solution

sysenter

Persistent data-only malware

- Activating the persistent control structure



Outline

Proof of Concept

Victim	Ubuntu 64-bit Server (Kernel 3.8) with secure boot (UEFI)

Proof of Concept

Victim	Ubuntu 64-bit Server (Kernel 3.8) with secure boot (UEFI)
Vulnerability	CVE-2013-2094 Local Root Exploit Data-only version

Proof of Concept

Victim	Ubuntu 64-bit Server (Kernel 3.8) with secure boot (UEFI)
Vulnerability	CVE-2013-2094 Local Root Exploit Data-only version
Hooks	sys_read, sys_getdents

Proof of Concept

Victim	Ubuntu 64-bit Server (Kernel 3.8) with secure boot (UEFI)
Vulnerability	CVE-2013-2094 Local Root Exploit Data-only version
Hooks	sys_read, sys_getdents
Functionality	Key logging, process hiding, file hiding

Outline

Conclusion

- **Persistent** data-only malware is possible

Conclusion

- **Persistent** data-only malware is possible
- Mainly **technical** challenges

Conclusion

- **Persistent** data-only malware is possible
- Mainly **technical** challenges
- Data-only malware is a **realistic** threat

Conclusion

- **Persistent** data-only malware is possible
- Mainly **technical** challenges
- Data-only malware is a **realistic** threat

POC code available on our website

http://www.sec.in.tum.de/persistent-data-only-malware/

Bibliography I

Ralf Hund, Thorsten Holz, and Felix C. Freiling.
Return-oriented rootkits: Bypassing kernel code integrity protection mechanisms.
In *Proceedings of 18th USENIX Security Symposium*, 2009.