Ret2dir: Rethinking Kernel Isolation

Kermerlis et al. Columbia University

Presented by Lucas Copi

Introduction

- In the past attackers have focused efforts on server and client applications
  - Due to lesser complexities and simpler vulnerabilities
- Recently, attacks have become more focused on the Kernel
  - User-level software has become increasingly more secure
- In 2013 there were 355 reported kernel vulnerabilities-140 more than 2012

Introduction Continued

- Why the kernel?
  - Gives greater amount of control over system
    - The kernel is mapped into the address space of each process
    - Allows attackers to execute code in privileged mode
  - Larger code base
    - 16.9 MLOC in Linux Kernel v. 3.10
  - Increased vulnerabilities
    - Stack, heap, buffer overflows
    - Integer overflows
    - Missing authorization checks

Purpose of paper

- Expose design weaknesses in memory management subsystems of Linux
- Introduce new methodology for mounting ret2dir attacks
  - Bypassing current security methods embedded in chip architectures
  - Intel and ARM introduced SMEP, SMAP, and PXN processor features in an attempt to prevent control flow transfers from the kernel space to the user space
- Evaluate effectiveness of ret2dir attacks
- Present new designs and implementation of memory schemes to mitigate future attacks
  - Implemented with minimal overhead

Return to User Exploits

- Made possible by a shared address space between user and kernel processes
- Attackers can hijack privileged execution paths and redirect them to the user space
  - Allows attackers to execute shell code with kernel privileges

Source: http://hypervsir.blogspot.com/2014/11/defending-against-ret2dir-attacks.html

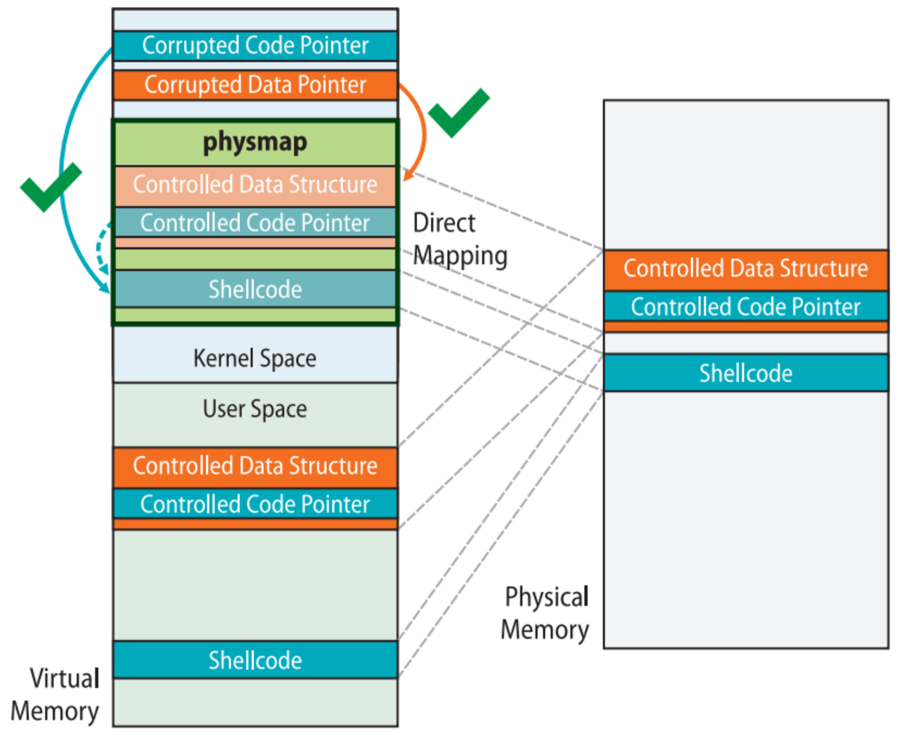

Figure 2: Overall ret2dir operation. The physmap (direct-mapped RAM) area enables a hijacked kernel code or data pointer to access user-controlled data, without crossing the user-kernel space boundary.

Ret2usr continued

- Carried out through memory corruption bugs in kernel code
- Attacks aim for control data
  - Return addresses
  - Function pointers
  - Pointers to critical data structures in kernel's heap
- Attacks focus on data that affects the control flow of the kernel
  - Allows attack to reroute execution to user space code
- Attacks can be carried out on multi-OS and multi-architectures

Protections against ret2usr

- Several defense mechanisms exist
- PaX KERNEXEC and UDEREF
  - prevents control flow transfers from kernel to user space
    - Relies on memory segmentation that returns a memory fault whenever privileged code attempts to dereference a pointer
    - Memory remapping contains significant overhead
    - Ported to ARM architecture but are supported for only A Arch32

Protections continued

- SMEP/SMAP/PXN
  - Intel features that facilitate stronger address space separation that are similar to KERNEXEC and UDEREF
- kGuard
  - Cross-platform compiler extension that protects kernel without relying on hardware features
  - Enforces address space segregation by preventing transitions between privileged execution paths to user space at compile time

Attack overview

- Linux's design trades weaker kernel to user segregation in favor of higher performance
- Research findings show vulnerabilities rooted in the Linux memory management that can be exploited to weaken memory isolation between kernel and user space
- Assumes Linux kernel with ret2usr protection mechanisms previously discussed
- Assume an unprivileged attacker with local access
- Do not make assumptions about the type of target—either data or code

New Threat Model

- Implicit physical memory sharing between user processes and the kernel allow attackers to deconstruct the isolation offered by the ret2usr protections
  - Made possible through process called physmap
  - Large virtual memory region inside kernel address space that contains direct mappings to physical memory
  - Allows kernel to allocate dynamic memory efficiently
- Although physmap is unique to Linux, other OSes have similar functions
- This means the memory of an attacker controlled user process is accessible through a kernel synonym

Mounting a ret2dir attack

- First step in attack is to place the exploit code(“payload”) in the user space
- It is necessary to understand how physmap operates in order to locate the physical kernel space synonyms for the payload
  - Physmap is architecture independent
  - The mapping between the virtual address space and the physical memory always starts at a fixed known location
  - In x86-64 architecture physmap is mapped as RWX (readable, writable, executable) in almost all kernel versions

Locating Synonyms

- Locating synonyms relies on accessing pageframe information available through the pagemap interface of the /proc filesystem
  - This pageframe information is available to all users including non-privileged users on all major Linux distributions
- Once the virtual page number is found, the physical synonym can be calculated
  - Due to aforementioned fixed starting point in memory space
- For systems with partial direct-mapped Ram mm allocates zones for page frame requests
  - Attackers exploit the levels of zones by making repeated memory allocation calls forcing mm into lower and more privileged zones
  - Once a lower zone is allocated, it is guaranteed to be present in physmap and susceptible to the previous vulnerability

Locating synonyms with physmap spraying

- In case where PFN information is unavailable, attack can work backwards
  - Pick arbitrary physmap address and place exploitable payload in mapped user page
- This is achieved in a similar way to heap spraying by exhausting the address space with copies of the exploit payload
  - Done similar to previous PFN attacks using continuous memory calls until mm swaps “sprayed” pages to disk
  - In order to continuously tie up memory, background processes write access allocated memory pages to ensure mm does not reallocate the page
- This method has a probability of success as high as 96%

Bypassing ret2usr security

- Using the above methods to find synonyms and exploit physmap attackers are able to corrupt kernel data pointers kdptr and kfptr
- Because security methods for preventing ret2usr kernel attacks focus on the segregation between user space and kernel space memory, attacks can overwrite kernel data pointers with data located in the synonym memory pages of the payload exploit
- Once the attackers control a kernel pointer they have the capability to change the memory access rights of user pages—specifically the pages where the payload is located
  - Due to the RWX vulnerability of phymap previously discussed

Security evaluation

- Researchers took 8 ret2usr exploits and ran them against both protected and unprotected kernels
  - Results were as expected: all exploits succeeded against nonhardened kernels and failed against hardened kernels
- Exploits were then modified to carry out ret2dir exploits instead of ret2usr
  - Newly designed exploits were able to successfully bypass all security mesures in place for ret2usr
  - For attacks utilizing physmap spraying the probability of success increased with size of memory
  - For 1Gb systems 65%, 2Gb 88%, and 16GB 96%

Defending against ret2dir attacks

- Initial security advancements can be found by increasing permission levels needed to access /proc filesystem—eliminating ability of an attacker to gain pageframe information
- Researchers present a new page frame ownership called eXclusive Page Frame Ownership (XPFO) for Linux kernel that provides protection against ret2dir attacks
  - System is designed to minimize performance overhead
  - Done by leaving physmap and kernel components that interact with buddy allocator untouched

XPFO

- Thin management layer that enforses exclusive page frame ownership
  - Page frames may never be assigned to both the kernel and the user space
- Whenever a page frame is assigned to a user process XPFO unmaps its respective synonym from the physmap
  - Ensures malicious code can no longer be injected into the kernel space
- When a page frame is released back to the kernel XPFO maps the corresponding pages back into physmap
  - Process must ensure pages are sanitized before returning them to the kernel

Evaluation

- XPFO provides protection against ret2dir attacks but does not protects against attacks that use generic data sharing between the user and kernel space
- XPFO was implemented on the kernel versions previously used to test the effectiveness of the ret2dir attacks
- The same set of attacks were carried out against the systems with XPFO
- In each scenario XPFO was able to prevent the attack
- XPFO introduces minimal overhead
  - Ranging between .18-2.91%
- XPFO is available at http://www.cs.columbia.edu/~vpk/research/ret2dir/

Conclusion

- As kernel code bases continue to expand and OS level applications continue to become more secure, kernel based attacks will become more prevalent
- Although the paper specifically targets Linux kernels, the ret2dir attack can be modified for many different machines
- Systems that are designed for increased performance often create vulnerabilities in the memory management processes
- In the future kernels need to be designed with increasing security measures to combat the increase in attacks
- The paper shows there are methods for implementing more secure systems while maintaining current efficiency

Reference

- ret2dir: Rethinking Kernel Isolation. Vasileios P. Kemerlis, Michalis Polychronakis, and Angelos D. Keromytis. In UsenixSecurity'14.

ret2dir: Rethinking Kernel Isolation

Vasileios P. Kemerlis, Michalis Polychronakis, and Angelos D. Keromytis. In UsenixSecurity'14

Paper Discussion

- Lucas Copi
- CSC 6991
- OS Security
- The paper *ret2dir: Rethinking Kernel Isolation* discusses the increase in kernel based attacks and the reimplementation of attacks that were previously used as ret2usr attacks. The paper details the technologies currently employed by several architectures to combat ret2usr attacks and suggests a new page frame management system to combat future attacks.
- While current architectures have methods in place to ensure the segregation of the user and kernel space and thus prevent ret2usr attacks, new attacks are able to leverage vulnerabilities in Linux kernel distributions to carry out similar attacks in a different fashion. Ret2dir attacks utilize the synonyms of user space pages to overwrite kernel data pointers and carry out malicious code with higher privilege levels. Because the current security methods for preventing ret2usr attacks focus only on the segregation between the kernel and user space, attackers are able to overwrite kernel pointers with malicious data stored in a kernel level page synonymously mapped to a user memory page. The synonyms can be calculated due to the memory allocating processes fixed starting point or through a process known as physmap spraying.
- The paper also details a new management system that prevents ret2dir attacks by forcing pages to be mapped exclusively to the user or kernel space. This new management layer known as XPFO was able to prevent all of the ret2dir attacks carried out on the Linux distributions in the previous sections of the paper and was implemented with minimal and often negligible overhead.

Paper Discussion

- Zhenyu Ning
- CSC 6991 – Advanced Computer System Security

- As well-known ret2usr attacks has been prevented from both architectures of Inter and that of ARM by a series features like KERNEXEC, UDEREF, SMEP, SMAP, PXN and kGuard, this paper present a similar attack named ret2dir which can bypass all the protection mechanisms mentioned above.

- This attack leverages vulnerabilities of a memory area named physmap, which is shared by kernel space and user space using a virtual memory aliases for better performance of allocating and managing dynamic memory. Ret2dir attack firstly try to allocate memory of certain size, and then get both its the page frame number(PFN) and the corresponding memory address alias in physmap by predefined formulas. If the attacker needs more physical contiguous memory, he can repeat this allocating until he gets contiguous PFNs. Then the attack can corrupt a kernel data pointer or function pointer with arbitrary value from a user-space address, as it is also a kernel-space address in physmap. After that, attack similar to ret2usr can be done without constraining of protection mechanisms. If memory addresses in physmap are marked as non-executable, ROP can be used to conquer it.

- The paper also concludes an exclusive page frame ownership scheme, XPFO, for the Linux kernel to defend ret2dir. This scheme unmap memory from physmap when the memory is assigned to a user process and map it back again when the user process releases it back to kernel. Though XPFO can efficiently defend ret2dir, other kinds of data sharing between user-space and kernel-space may also achieve similar attacks.
- EndFragment

Paper Discussion

- Sharani Sankaran
- CSC6991 Advanced Computer Security

- This paper ***ret2dir: Rethinking Kernel Isolation*** mainly discusses to redirect the corrupted kernel pointers residing in the user data space. In this paper they implement the implicit page frame sharing can be leveraged for the complete circumvention of software and hardware kernel isolation protections .

- A new kernel exploitation technique is also implemented return-to-direct-mapped memory (ret2dir), which bypasses all existing ret2usr defenses, namely SMEP, SMAP, PXN, KERNEXEC, UDEREF, and kGuard. Thus a fundamental design weakness in memory management subsystem by linux with the introduction of ret2dir attacks. Then the 2 techniques which forcing user-space exploit payloads to "emerge" within the kernel's direct-mapped RAM area. evaluate the effectiveness of ret2dir attacks using a set of nine exploits against different LINUX kernel configurations. The design, implementation, and evaluation of an exclusive page frame ownership scheme for the Linux kernel.
- XPFO is a thin management layer that enforces exclusive ownership of page frames by either the kernel or user-level processes that prevents e implicit sharing of physical memory

Paper Discussion

- Hitakshi Annayya

- The paper “**ret2dir: Rethinking Kernel Isolation**” discusses about new attack technique called return-to-direct-mapped memory - **ret2dir** which bypass all the existing defense such as namely SMEP, SMAP, PXN, KERNEXEC, UDEREF, and kGuard of ret2usr – is an another attack which redirects corrupted kernel pointers to data residing in user space. Ret2dir is much reliable against these architectures x86, x86-64, AArch32, and AArch64 Linux targets. The paper also discusses about defending technique for ret2dir attack (linux kernel) à page frame ownership scheme with negligible runtime overhead.

- Ret2dir attack threat model deeply rooted into the architecture of the Linux memory management subsystem (mm), which can be abused to weaken the isolation between kernel and user space.

- Defending against ret2dir attack presents the design of an eXclusive Page Frame Ownerwhip (XPFO) scheme for the Linux kernel that provides effective protection with low overhead which hinders but cannot prevent ret2dir attacks.

- **Limitations of XPFO**
- XPFO does not prevent data sharing between kernel and user space.

- To evaluate the effectiveness of the proposed protection scheme, XPFO patch to each of the six kernel versions tested ret2dir exploits when XPFO was enabled. In all cases, XPFO prevented the exploitation attempt. Overall, XPFO introduces a minimal overhead, ranging between 0.18–2.91%.

Term Project Proposal

- Next class: Proposal discussion

- Prepare a 10-minutes presentation
  - Define the problem
  - Explain related work
  - State your new approach