Subverting the Xen hypervisor

Rafał Wojtczuk
Invisible Things Lab

Black Hat USA 2008, August 7th, Las Vegas, NV

Xen 0wning Trilogy

Part One

Known virtulization-based rootkits

- Bluepill and Vitriol
- They install a malicious hypervisor in run-time
  - ... On a system where no hypervisor is present
- What if there is a legal hypervisor already running ?

Subverting a legal hypervisor

- Many analysts predict that in near future, many systems will run some sort of hypervisor by default
- If we modify the code or data structures of a legal hypervisor, we may achieve capabilities similar to the ones available for Bluepill or Vitriol (stealth !)

Challenges

- The legal hypervisor can protect itself against runtime modification, even if the attacker has all privileges in the management OS.
- It may be nontrivial to reliably integrate the backdoor code with the legal hypervisor

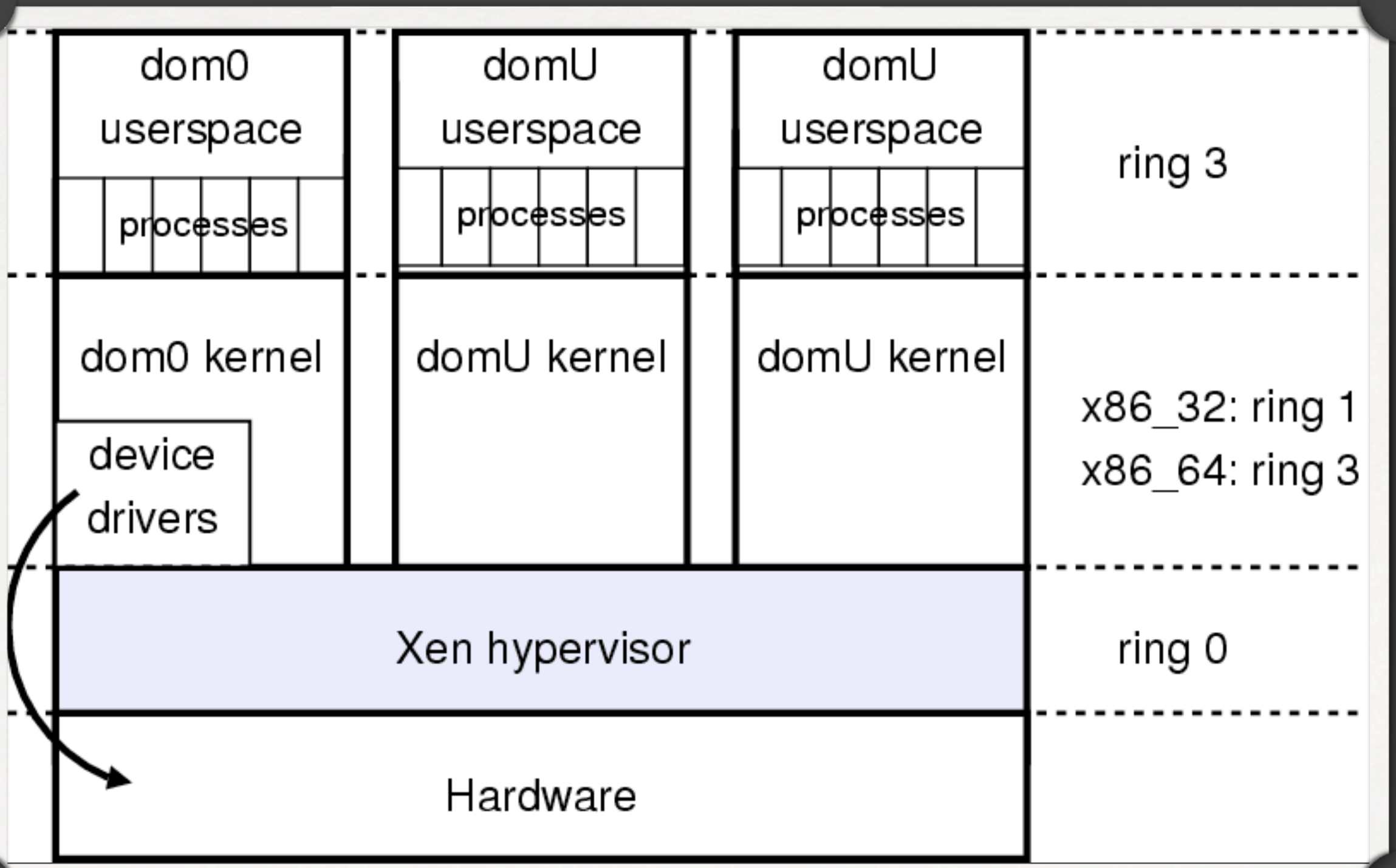
Opportunities

- No need to hide the mere presence of a malicious hypervisor
  - As the side-effects of its presence can be attributed to the legal hypervisor
- A lot of required functionality is already implemented in the legal hypervisor
  - Particularly, protection of the hypervisor code from the underlying VMs

Subverting Xen 3.x

- We will discuss: how Xen 3.x hypervisor code or data (on x86_32 or x86_64) can be modified in runtime to allow for backdoor functionality
- The implementation of a framework allowing to load compiled C code into the hypervisor
- The implementation of two backdoors

Xen architecture



Xen architecture, cntd

- Only Xen code runs in ring 0
  - Directly loaded by a bootloader
  - All administrative actions are done in a selected VM (dom0)
    - We would like to have backdoor access to dom0
    - Dom0 is under full control of the hypervisor

Xen architecture, cntd

- Dom0 has direct access to most of the hardware
  - So that device drivers written for dom0's OS (usually Linux) can be used
- Paravirtualization vs full virtualization
  - Dom0 is a paravirtualized domain

Xen hypercalls

- Reminder: on Linux, system calls are reachable from usermode by invoking int 0x80; the handler for this interrupt invokes the appropriate system call
- Similarly, on Xen, hypercalls are reachable from the guest by int 0x82
- Examples: do_mmu_update, do_set_gdt

Getting control over Xen

- We assume attacker has root in dom0 and wants to install a stealth backdoor
  - There were vulnerabilities related to pygrub allowing to escape from domU to dom0 (CVE-2007-4993, CVE-2007-5497)
- Still, there is no supported method for dom0 to alter Xen's memory
- Rebooting into modified Xen is noisy

Papers by Luic Duflot

- Pacsec2007: Programmed I/O accesses: a threat to Virtual Machines ?
  - Abuses GART and USB subsystem in order to overwrite arbitrary phys memory
- Cansecwest 2006: Security Issues Related to Pentium System Management Mode
  - SMM memory is locked nowadays...

How to abuse DMA

- Any DMA-capable device can access arbitrary physical memory
  - Including Xen's code
  - Dom0 can setup DMA !
  - domU can be allowed to access some hw
- Can we conveniently setup an arbitrary DMA transfer with a device used by OS, not disturbing normal operation ?

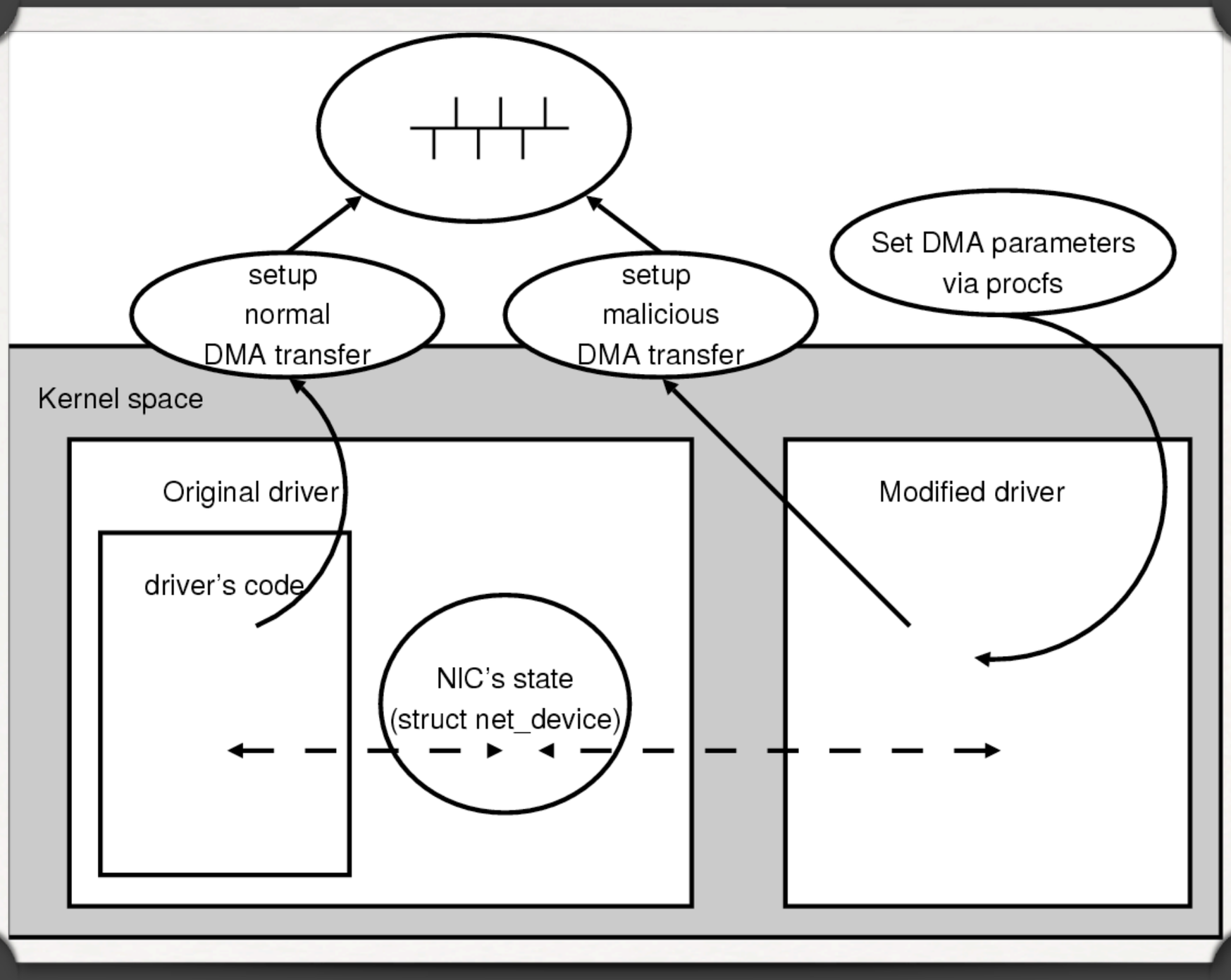
Mitigation

- AMD's IOMMU and Intel's VT-d can restrict the range of addresses that a DMA device can access
  - Not many chipsets support it today
- This presentation stresses the importance of these mechanisms
- ...but in the next talk we will show the ways to modify Xen, regardless of VT-d...

DMA with a NIC

- Loopback mode of NIC allows to copy data between two locations in RAM
  - It will disconnect us from the net for a fraction of second; blame ISP ☺
- We will load a modified NIC driver that will reuse data structures of the original driver (as we can't unload the original)
- Still, we need to modify each NIC driver

NIC DMA diagram

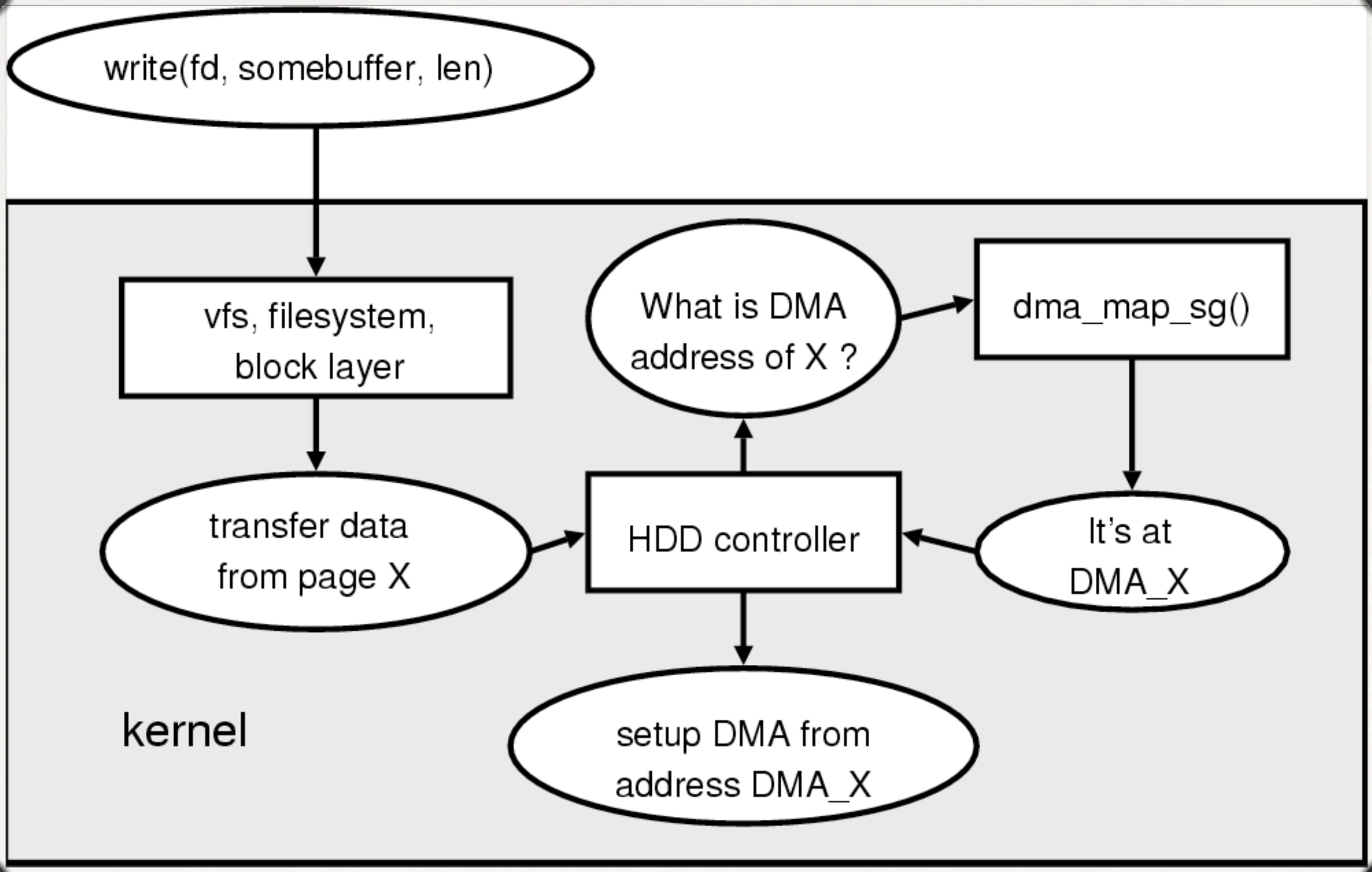


Tg3dma demo

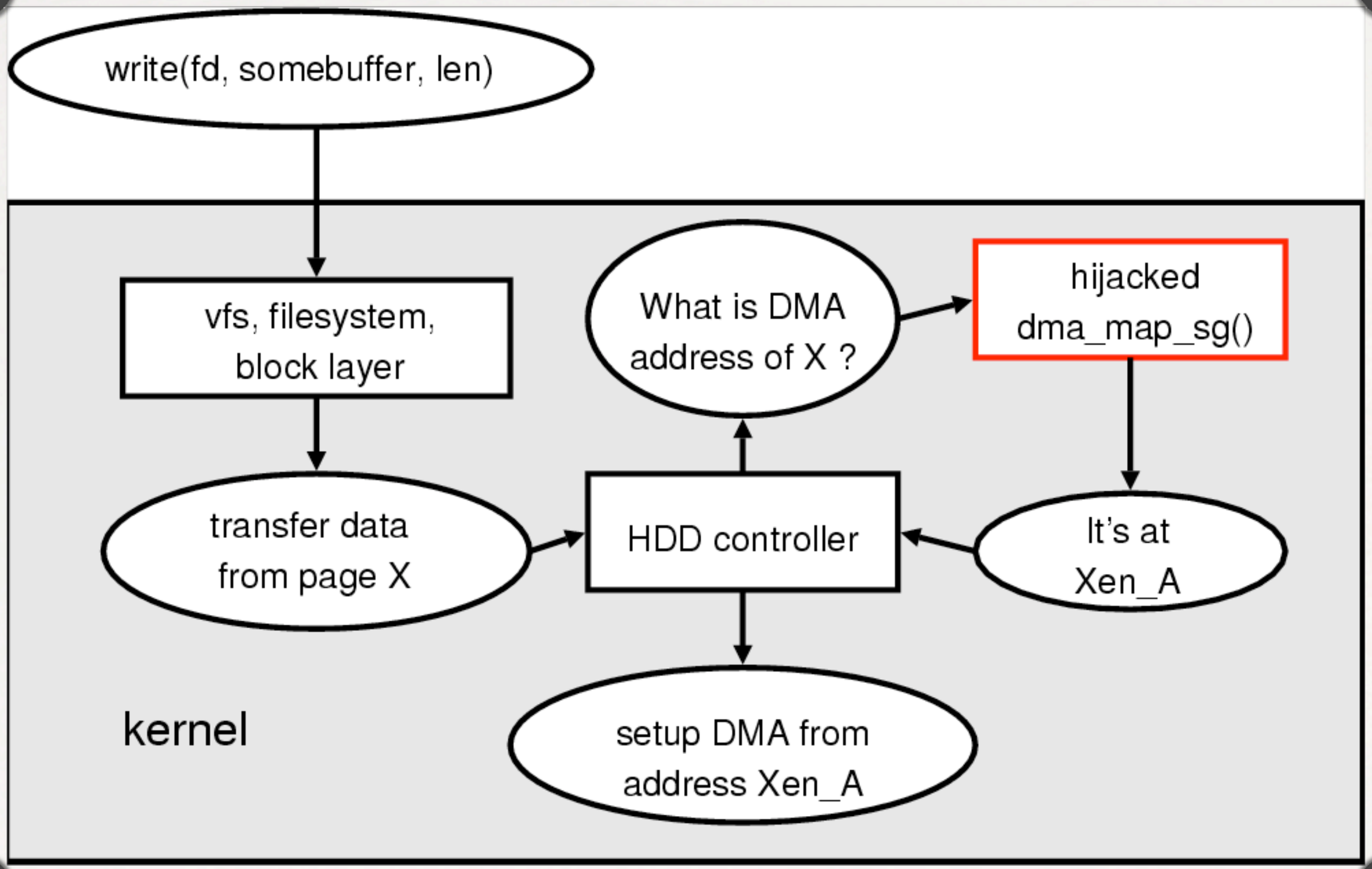
Abusing HDD controller

- All tested HDD controllers called dma_map_sg() in order to retrieve a suitable bus address for a transfer
- Thus, we can use _any_ controller, without modifying it: just change the behavior of dma_map_sg()

Normal HDD operation



Hijacked dma_map_sg



When to cheat in dma_map_sg hook ?

- We cannot pass Xen_A for all HDD operations – fs or kernel corruption
- Let usermode call *write(somefd, somebuffer, length);* will the dma_map_sg() parameters correspond to *somebuffer* address ?
- Yes, but only in O_DIRECT mode; otherwise, filesystems cache pages will interfere

Getting ring0 without DMA

- Even if we have IOMMU or VT-d (configured properly), there may be bugs in Xen code:
  - Buffer overflow (or similar)
  - Logic error in crucial code paths, e.g. In memory management
- So the following slides are relevant

Xen memory layout

- What to overwrite with DMA ?
- Xen is loaded at physical address 0x00100000 by the bootloader
- On x86_32, substract 0xff000000 from the virtual address of a Xen function/structure to get its phys address

Important Xen functions

- Hypercalls: stored in *hypercall_table* array
- Exception handlers: stored in *exception_table*
- *Printk*, *copy_from_user*, *xmalloc*
- We can get their virtual addresses from xen-syms file; if the latter is not available, we need some simple RE

What to overwrite ?

- We will overwrite (with DMA) the body of *do_ni_hypercall* with „call first_argument” assembly instruction
  - Normally, *do_ni_hypercall* consists of „return –ENOSYS”, it is unused
- Then if we invoke hypercall no 11 with the first parameter X, the code at X will be executed in ring 0; int run0(void *fun)

Xen loadable modules

- *Xenload* tool allows to load a relocatable ELF object (module.o) into Xen; the algorithm:
  - Link module.o with xenlib.o to module.xko
  - Allocate space on Xen's heap via run0(xmalloc)
  - Copy module.xko via run0(memcpy)
  - Run0(init_module); DEMO

Xen backdoors design

- When idle, it must not modify anything in dom0
  - ... So that a /dev/mem scanner running in dom0 cannot detect it
- Arbitrary shell commands execution in dom0 when a „magic” network packet is seen
  - The executed shell commands are not hidden (more on this later)

Debug register backdoor

- Dr backdoor sets dr3 and dr7 registers so that when dom0 executes netif_rx(), debug exception is raised
- When the (replaced) debug exception handler sees exception from eip=netif_rx, it scans the packet payload for the presence of „magic” pattern; if found, execute shell with parameters taken from the payload
- Dom0 can’t see the changed debug regs

Dr backdoor, cntd

- The fault to Xen happens very low in the network stack, *before* the firewall sees the packet
- The „magic” packet is never delivered to dom0

Dr backdoor, cntd

- Dr3 and dr7 are set:
  - During inter-VM context switch to dom0
    - In do_set_debugreg hypercall, when dom0 sets the relevant dr7 bits to 0 (indicating it does not use dr3)
- Dr3 and dr7 are unset (released for dom0 use) in do_set_debugreg hypercall when dom0 wants to use dr3

Dr backdoor, cntd

- When dom0 queries dr value via do_get_debugreg hypercall, it is given fake „shadow” values
- Unfortunately, due to the „lazy” handling of dr assignments by Linux kernel, dr3 may remain set (in-use by dom0) even when the debugged process has exited

Dr backdoor, cntd

- When „magic” packet is seen, the exception handler copies „trampoline” code to a preallocated page in dom0 and transfers control there
- The trampoline forks a shell by calling *call_usermodehelper_keys()* and then self-destructs by returning into *memset*

Dr backdoor demo

Dr backdoor detection

- When idle, it cannot be detected by memory scan
- Perhaps the debug regs handling is not entirely transparent...
- ... But the main problem is the timing analysis; the first instruction of netif_rx() takes too much time to execute !
  - If the NIC drivers were in Xen space... ☹

Foreign backdoor

- Dr backdoor hooks a function in dom0 and thus is subject to timing analysis
- We need other method to inspect dom0's state
- Solution: instead of hooking, scan periodically
- The „magic" condition must last longer than the scan interval

Foreign backdoor, cntd

- Xen provides API for a domain to map pages from other domain
  - Only dom0 is allowed to do this
- We will start a „lurker” domain and make it privileged (by altering Xen structures)
- When the lurker sees a „magic” condition, it will spawn a shell in dom0

Foreign backdoor, cntd

- Currently the magic condition is: an sshd process in dom0 received a „magic” identification string
- When it happens, the lurker domain will overwrite sshd stack and saved registers in the kernel stack so that shell is executed
- What if dom0 is firewalled ? No big deal.

Foreign backdoor, cntd

- Xen offers API to
  - Retrieve cr3 of a target domain
  - Map a page by its physical address
- Libxenctrl library combines the two to produce kernel virtual address resolution
- **Xenaccess** project can resolve userland virtual addresses as well; but due to some problems we don't use it (and had to recode this functionality)

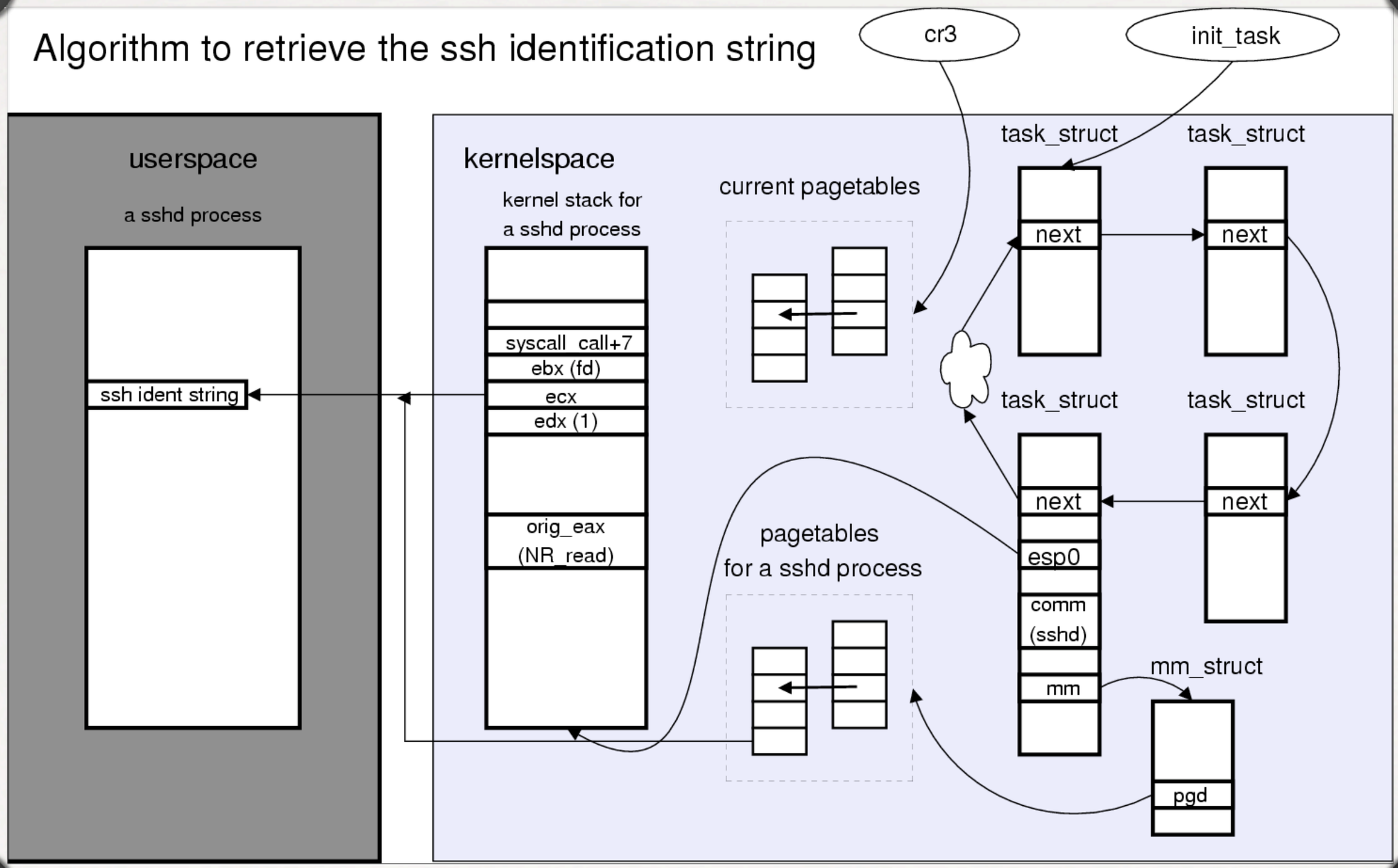
Foreign backdoor, cntd

- Opensshd reads ident into a stack buffer

```
for (i=0; i<BUFSIZE; i++)
        read(sock_in, &buf[i], 1)...
        if (buf[i]==‘\n’) break
```

- So when the whole ident string has been received, sshd sleeps in the *read* syscall, with the *count* parameter being 1, and the *buffer* parameter pointing just after the received

Algorithm to retrieve the ssh identification string



Foreign backdoor, cntd

- How to change the sshd process into shell ?
  - We can set eip (on the kernel stack), we can set the stack content, any problem ?
  - NX+ASLR (present on modern Linux distributions) is a problem
  - Solution used: transit through sys_sigreturn and sys_mprotect to make the stack executable; we need *no* offsets !

Foreign backdoor demo

Foreign backdoor, detection

- This time, timing analysis will not help, as the backdoor executes in time slices devoted for other domains
- The lurker domain can be hidden
  - More work on this is needed, the „domlist” loadable module is a good start
  - Still, all information comes from the hypervisor, so we can filter it

IPfrag backdoor

- Executes during inter-VM context switch, in Xen address space
  - So no need for a separate domain
- Walks the *ipq_hash* kernel table to locate all IP fragments received so far; scans them for a magic pattern
  - Spoofed source IP may evade firewall
  - No need to resolve userland addresses

IPfrag backdoor, cntd

- Note: foreign backdoor, if using legal domain, can leave Xen code intact
- Show me the code !

- Not yet; there are significant implementation differences in IP fragments handling between Linux kernel versions
- Ipq_hash is not exported in kallsyms, need a reliable way to find it

More on stealth

- The processes executed by the described backdoors are well-visible
- We could hook int 0x80 handler to provide system calls filtering...
- ... But it will fool only dom0 userland, not the kernel
- So not implemented at all

More on stealth, cntd

- Instead of executing a separate process, we could force some kernel thread (e.g. *khelper*) or any other existing process to do the desired action, maybe setup syscall proxy
- Better, but still visible from dom0 kernel

More on stealth, cntd

- The only really stealth operation is viewing of the domain memory
  - Maybe writing as well, in some cases
  - If we want a backdoor capable of silently extracting e.g. crypto keys from some dom0 process, it can be done
  - It can „phone home” by many means; for example, by altering contents of fragmented ICMP echo requests

Thank you!

Xen 0wning Trilogy to be continued in:

“Preventing and Detecting the Xen Hypervisor Subversions” (after the lunch)

by Invisible Things Lab

Bibliography

- Joanna Rutkowska, *Subverting VistaTM Kernel for Fun and Profit*, http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Rutkowska.pdf
- Dino Dai Zovi, *Hardware Virtualization Rootkits,* http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Zovi.pdf
- VMware Server, http://www.vmware.com/products/server/
- CVE-2007-4993,http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-4993
- Multiple integer overflows in e2fsprogs (Xen related),http://www.mcafee.com/us/local_content/misc/threat_center/e2fsprogs.pdf
- AMD, *AMD IOMMU specification 1.2*, http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/34434.pdf

Bibliography cntd

- CVE-2007-4993,http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-4993
- Multiple integer overflows in e2fsprogs (Xen related),http://www.mcafee.com/us/local_content/misc/threat_center/e2fsprogs.pdf
- AMD, *AMD IOMMU specification 1.2*, http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/34434.pdf
- Intel,Intel Virtualization Technology for Directed I/O (Intel VT-d), http://www.intel.com/technology/magazine/45nm/vtd-0507.htm?iid=techmag_0507+rhc_vtd

Bibliography cntd

- Luic Duflot, *Programmed I/O accesses: a threat to Virtual Machines ?*, htttp://www.ssi.gouv.fr/fr/sciences/fichiers/lti/pacsec2007-duflot-papier.pdf
- Luic Duflot, *Security Issues Related to Pentium System Management Mode*, http://cansecwest.com/slides06/csw06-duflot.ppt
- halfdead@phear.org, Mistifying the debugger, ultimate stealthness, http://www.phrack.com/issues.html?issue=65\&id=8
- XenAccess Library, http://code.google.com/p/xenaccess/