CrashOS: Hypervisor testing tool

ISSRE 2017

Anaïs GANTET - Airbus *Digital Security*

October 2017

AIRBUS

AIRBUS

Outline

1. Why CrashOS?
2. CrashOS presentation
3. Vulnerability research and results

AIRBUS

Outline

AIRBUS

Project context

Goal

- Test the robustness of virtualization solutions

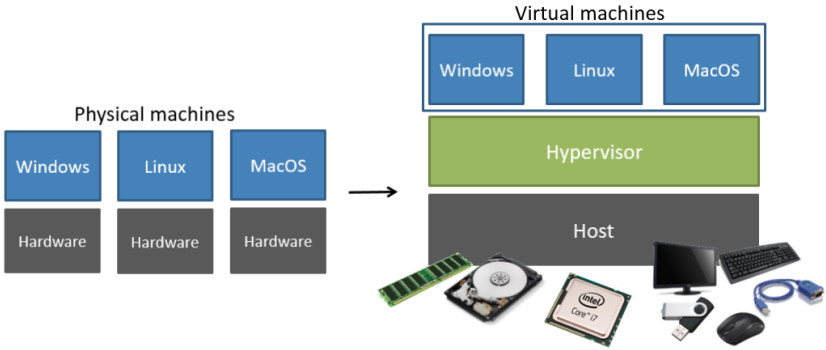
Scope

- Targeted architecture: Intel x86
- Initial targeted software: Ramooflax (`https://github.com/airbus-seclab/ramooflax`)
- Other targeted software: VMware, Xen, etc.

AIRBUS

Virtualization in a nutshell

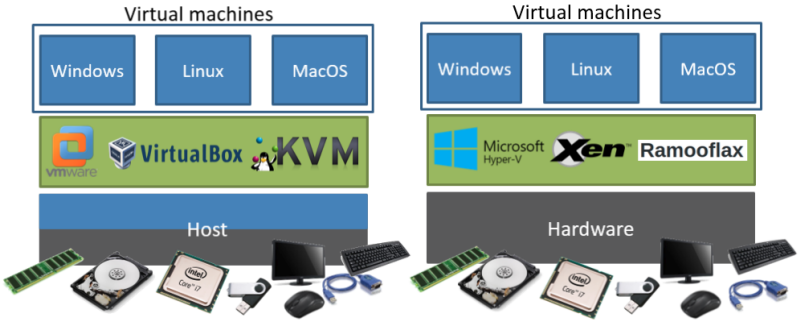
Aim: allow several Operating Systems (OS) to share hardware resources



AIRBUS

Virtualization in a nutshell

Aim: allow several Operating Systems (OS) to share hardware resources



AIRBUS

Hypervisor role

To provide an equivalent environment to a physical machine

- Processor virtualization
  - Let directly execute non-sensitive instructions
  - Trap and virtualize sensitive instructions
- Memory access virtualization
- Device virtualization

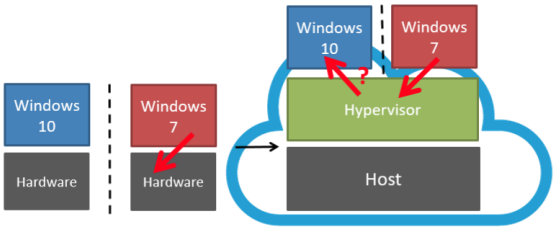
Hypervisors are

- Complex programs
- Operating at low level

AIRBUS

Security issues

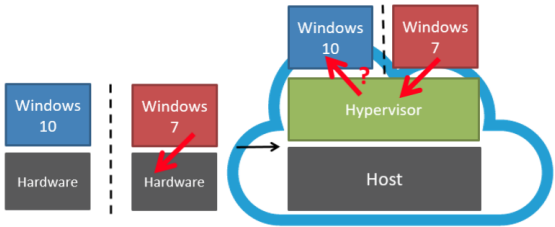
- Without virtualization: physical isolation
- With virtualization: software isolation (implemented by hypervisors)



AIRBUS

Security issues

- Without virtualization: physical isolation
- With virtualization: software isolation (implemented by hypervisors)



Problem

Are hypervisors reliable?

AIRBUS

Our approach

Behaviour analysis

- Build a testing VM
- Imagine tests involving the hypervisor
  - Be aware of known hypervisor weak points (CVE)
    - Instruction disassembly
    - Sensitive instruction emulation
    - Unusual CPU configuration
    - Device virtualization
    - etc.
- Launch them and observe hypervisor behaviour

AIRBUS

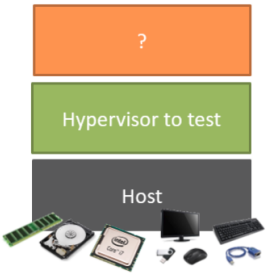
Outline

AIRBUS

Our way to build a malicious VM

Potential candidates?

- Windows, Linux: only partial control of hardware communication
- Simple Operating System, OSv, etc.: not appropriate for vulnerability research
- VESPA: Only part targeting device virtualization is published

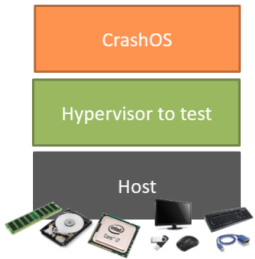


AIRBUS

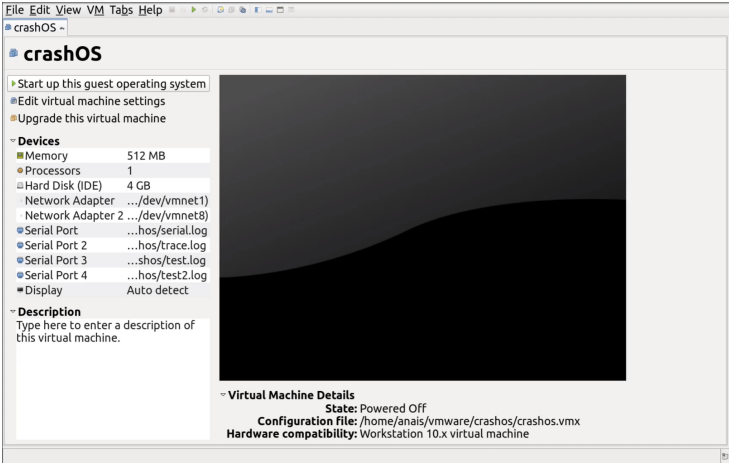
Our way to build a malicious VM

Potential candidates?

- Windows, Linux: only partial control of hardware communication
- Simple Operating System, OSv, etc.: not appropriate for vulnerability research
- VESPA: Only part targeting device virtualization is published

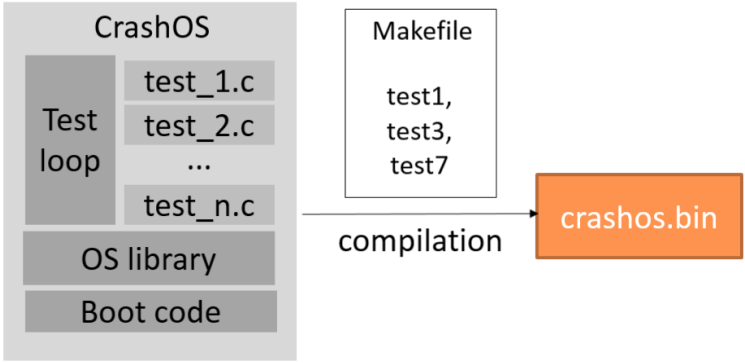


Our choice: develop our own OS dedicated to test hypervisors


File Edit View VM Tabs Help
crashOS
crashOS
Start up this guest operating system
Edit virtual machine settings
Upgrade this virtual machine
Devices
Memory 512 MB
Processors 1
Hard Disk (IDE) 4 GB
Network Adapter .../dev/vmnet1)
Network Adapter 2 .../dev/vmnet8)
Serial Port ...hos/serial.log
Serial Port 2 ...hos/trace.log
Serial Port 3 ...shos/test.log
Serial Port 4 ...hos/test2.log
Display Auto detect
Description
Type here to enter a description of this virtual machine.
Virtual Machine Details
State: Powered Off
Configuration file: /home/anais/vmware/crashos/crashos.vmx
Hardware compatibility: Workstation 10.x virtual machine

AIRBUS

How does CrashOS work?



AIRBUS

CrashOS test format

- Init function
  - Save the current machine state
  - Define the desired context
- Test function
  - Execute instruction
  - Print adapted logs
- Restore function
  - Reinit the saved machine state

```
test_t test_x = {
  .name = "test name",
  .desc = "test description",
  .init = init_test_x,
  .test = test_test_x,
  .fini = restore_test_x
};
DECLARE_TEST(test_x)
```

AIRBUS

CrashOS: Core and libraries

System features

- Physical memory access
- Memory protection mechanisms
- Interrupt handling
- Device communication
- Paravirtualization support

Example:

```
#define enable_paging()
  asm volatile (
  "mov %%cr0, %%eax        \n"
  "or $0x80000000, %%eax   \n"
  "mov %%eax, %%cr0":::"eax" )
```

AIRBUS

CrashOS: Core and libraries

System features

- Physical memory access
- Memory protection mechanisms
- Interrupt handling
- Device communication
- Paravirtualization support

Example:

```
#define enable_paging()
  asm volatile (
  "mov %%cr0, %%eax        \n"
  "or $0x80000000, %%eax   \n"
  "mov %%eax, %%cr0":::"eax" )
```

Other interesting features

- Printing logs on the screen
- Printing logs on the serial port

AIRBUS

Outline

AIRBUS

Attack approach

Prerequisites

- Advanced understanding of Intel processor mechanisms
- Good knowledge of hypervisor role

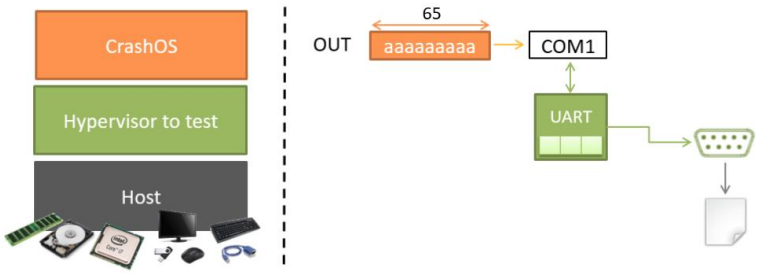
Elaboration of relevant tests

- Not only fuzz all instructions
- Always involve a specific hypervisor handling
  - Sensitive instructions
  - Memory management
  - Device emulation
  - etc.

AIRBUS

A first example: write a ’big’ buffer on serial port COM 1

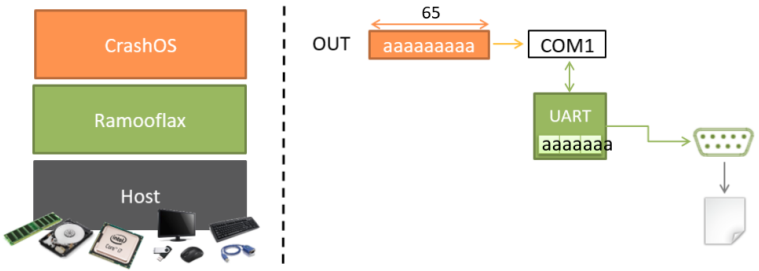
```
// REP OUTS instruction
  asm volatile (
          "mov $0x3f8,     %%edx \n" /* serial port COM 1 */
          "mov $0x400000,  %%esi \n" /* buffer address    */
          "mov $65,        %%ecx \n" /* number of byte    */
          "rep outsb             \n" :::);
```



AIRBUS

A first example: write a ’big’ buffer on serial port COM 1

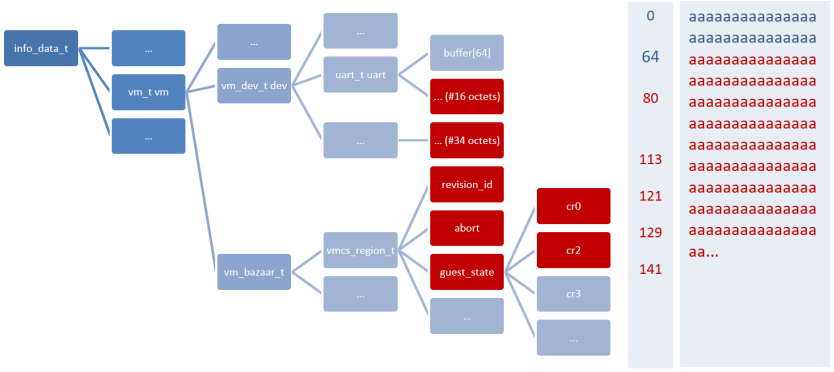
```
// REP OUTS instruction
  asm volatile (
          "mov $0x3f8,     %%edx \n" /* serial port COM 1 */
          "mov $0x400000,  %%esi \n" /* buffer address    */
          "mov $65,        %%ecx \n" /* number of byte    */
          "rep outsb             \n" :::);
```



AIRBUS

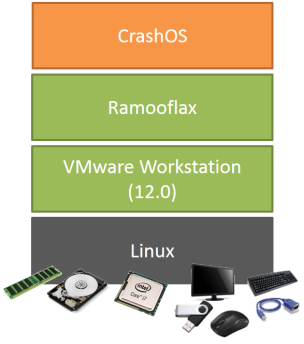
A first example: write a ’big’ buffer on serial port COM 1

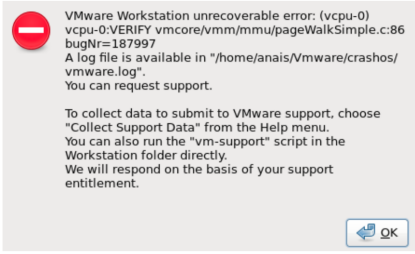
Buffer overflow and memory corruption of Ramooflax data (including VMCS)



AIRBUS

A first example: write a ’big’ buffer on serial port COM 1





Buffer overflow in Ramooflax causes a VMware crash

AIRBUS

A second example: Privilege changing (FAR JMP)

Our test case

Launch FAR JMP from high level to low level (not allowed - segmentation fault must occur)

What is FAR JMP instruction?

- Changes the current execution context to a new one
  - Can modify the current privilege level

Why to test this instruction?

- FAR JMP success: only if parameters meet about 20 conditions
- Sensitive instruction, intercepted and handled by hypervisors
- Test based on CVE-2014-8595 - Xen HVM

AIRBUS

A second example: Privilege changing (FAR JMP)

Xen (HVM):

```
root@debian:~# xl list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0   395     1      r-----     220.9
crashos                                     28   507     1      ---sc-       5.7
root@debian:~# xl dmesg_
```

```
(XEN) domain_crash called from vmx.c:2274
(XEN) Domain 28 (vcpu#0) crashed on cpu#0:
(XEN) ----[ Xen-4.2.4-pre  x86_32p  debug=n  Not tainted ]----
(XEN) CPU:    0
(XEN) EIP:    0030:[<00201b95>]
(XEN) EFLAGS: 00000006   CONTEXT: hvm guest
(XEN) eax: 000b8000   ebx: 0002dae0   ecx: 000b8000   edx: 000b8000
(XEN) esi: 00054769   edi: 0005476a   ebp: 0020721c   esp: 00207...
(XEN) cr0: 00000011   cr4: 00000000   cr3: 00800000   cr2: 00000...
(XEN) ds: 0023   es: 0000   fs: 0000   gs: 0000   ss: 0023   cs: 0030
```

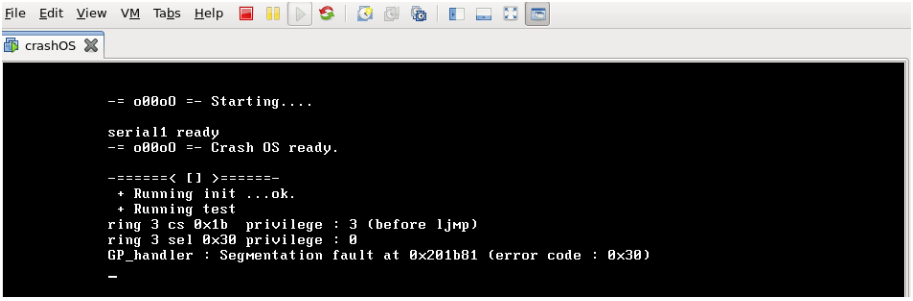
EIP_2

CS_2

AIRBUS

A second example: Privilege changing (FAR JMP)

VMware:



AIRBUS

CrashOS: Some test cases

	VMware	Ramooflax	Xen(HVM)
Access physical memory out of RAM size			
Use memory with bad rights			
Mix 16-bits and 32-bits code			
Read or write on all I/O ports			
Badly configure the paging tables			
Badly emulate context switching			
...			

	Correctly handled
	Not implemented
	VM crash (DoS)
	More critical impacts

AIRBUS

Results and discussions

- Vmware:
  - *Monitor panic* VERIFY
  - *Monitor panic* NOT_IMPLEMENTED
  - *Monitor panic* NOT_REACHED
  - *Monitor panic* EPT Misconfiguration

  Contact: VMware Security Response Center
- Xen
  - Bad emulation of far jmp and far call instructions reported
  - Ongoing to be fixed (`https://lists.xenproject.org/archives/html/xen-devel/2017-09/msg03701.html`)
- Ramooflax
  - All bugs fixed

AIRBUS

Conclusion and future outlook

CrashOS today

- Opensource
- Configurable minimal OS
- Simplified framework to write attacks on hypervisors
- First results on VMware, Xen and Ramooflax

Future outlook

- Elaborate new tests
- Implement new CrashOS features (64 bit support, *nested virtualization*, etc.)
- Test other hypervisors (Xen PV, KVM, Virtualbox, etc.)

AIRBUS

Thanks for your attention

Questions?

https://github.com/airbus-seclab/crashos

French paper: https://www.sstic.org/media/SSTIC2017/SSTIC-actes/crashos/SSTIC2017-Article-crashos-gantet.pdf

anais.gantet@airbus.com