UFO: Operating System Fingerprinting for Virtual Machines

Syscan Taipei, August 19th, 2010

NGUYEN Anh Quynh, Kuniyasu SUZAKI

AIST, Japan

Who am I?

- From the National Institute of Advanced Industrial Science and Technology (AIST), Japan
- NGUYEN Anh Quynh, Postdoc researcher
  - VNSecurity member (http://vnsecurity.net)
- Multiple interests: Operating System, Virtualization, Trusted computing, malware analysis, forensic, rootkits, IDS, ...

Motivation

- Problems of available Operating System Fingerprinting (OSF) tools against Virtual Machine (VM)
- A new approach and new tool named UFO to fingerprint OS inside VM, and fix some problems
  - Super fast
  - Accurately identify guest OS, regardless of OS tweaking
  - VM independence

Agenda

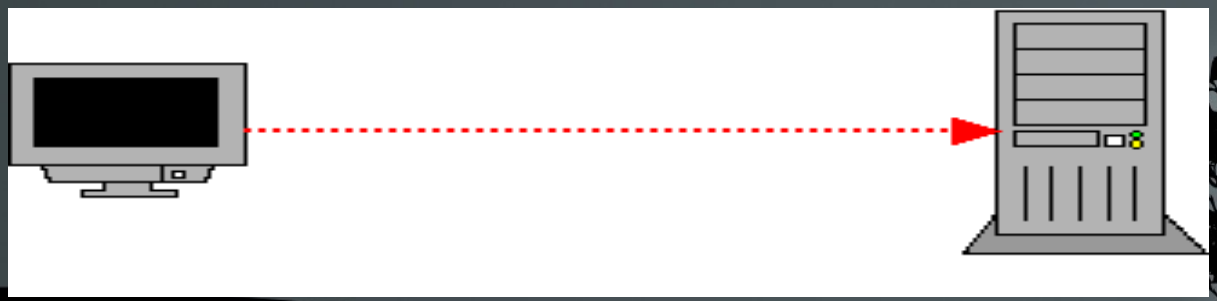
- Problems of available OSF solutions for OS in VM environment
- UFO solution
  - Design & Implementation
  - Main features
- Live demo
- Discussions
- Conclusions
- Q & A

Part I

- Problems of available OSF solutions for OS in VM environment
- UFO solution
  - Design & Implementation
  - Main features
- Live demo
- Discussions
- Conclusions
- Q & A

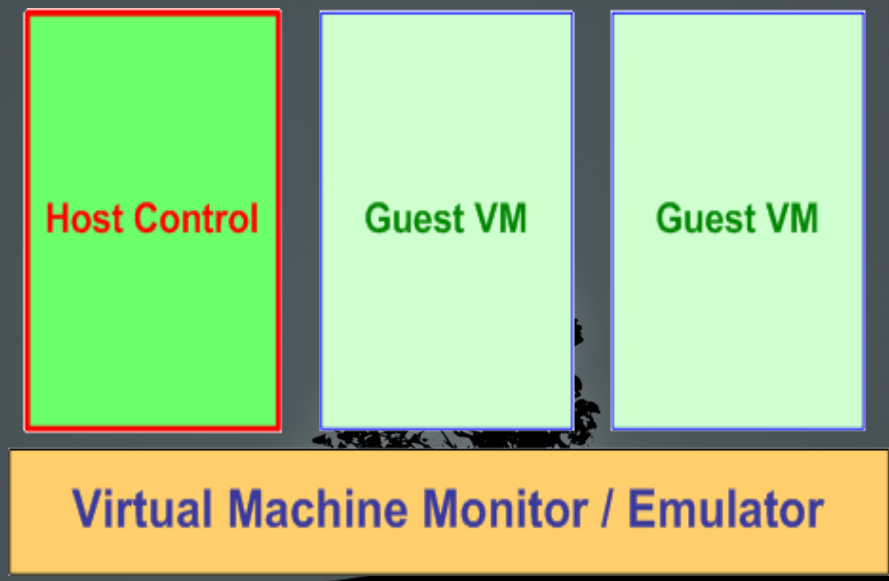
Traditional OSF problem

- Indentifying the OS (variant, version, Service Pack, …) without the access to the target
  - Traditionally mean fingerprinting remote target via network
- Why?
  - To defense
  - To attack
- An well-known problem in computer security field



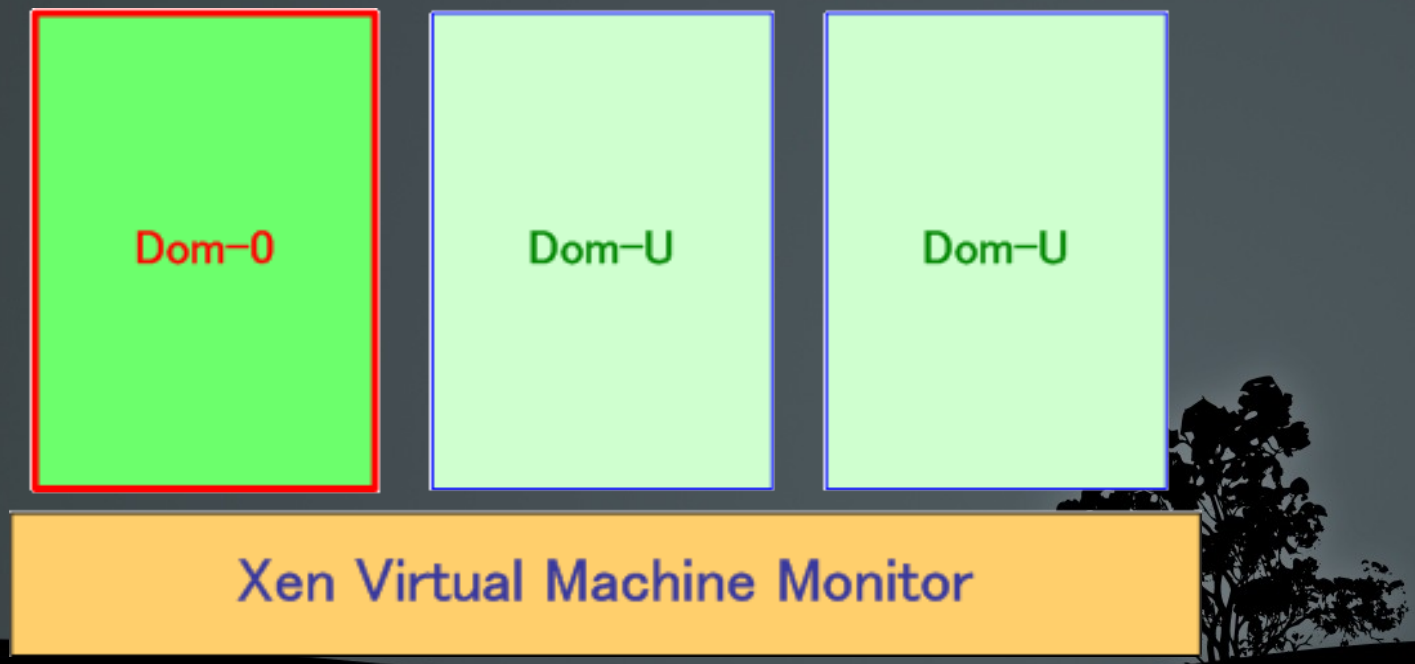
Virtual Machine Concept

- Running multiple virtual systems on a physical machine at the same time
  - Privilege VM
  - Guest VM
  - Multiple Operating Systems are supported
    - Windows, Linux, BSD, MacOSX, ...



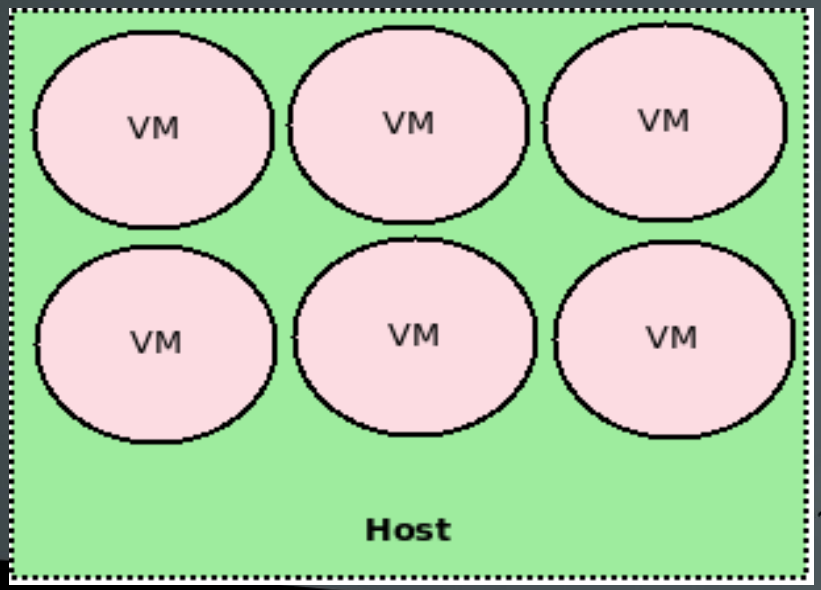
Xen Virtual Machine

- Host VM: Dom0
- Guest: DomU
  - Paravirtualized guest
  - Full-virtualized guest (HVM)

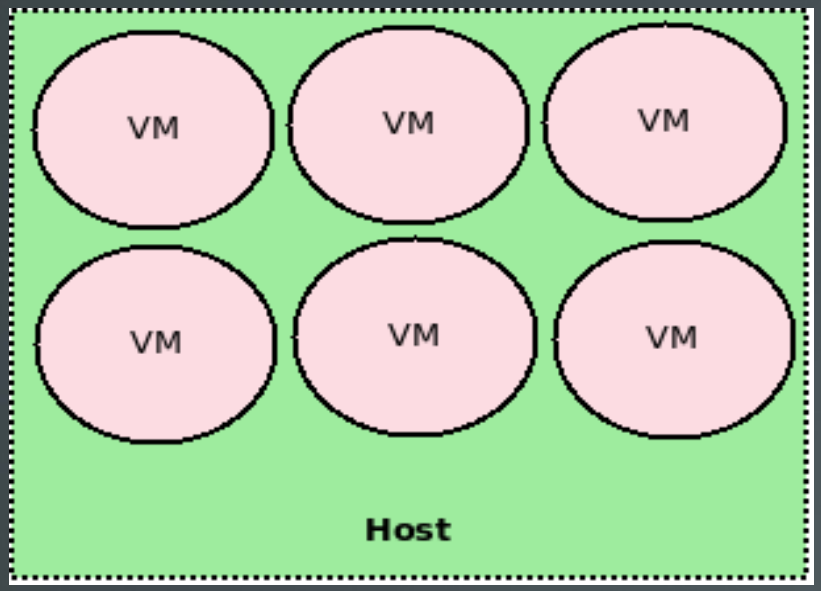


OSF problem for VM

- Indentifying the guest OS (variant, version, Service Pack, …), with legitimate access to the host
- Why?
  - To defense
  - First step to solve ”VM instrospection” problem
- A new problem in a very hot field



OSF for VM?



Network based OSF

- Traditional approach
  - Craft network packets to send to guest OS, and analyze the returned packets
- Multiple problems with modern OS-es
  - Port filtered/closed by default firewall
    - nmap fails to work
  - ICMP packets are dropped
    - xprobe fails to work
  - Sometimes slower than desired
    - Over 30 seconds for one guest OS, even on the same physical machine
  - Tweaking network stack is trivial

VM-specific solutions (1)

- Extract out information from guest's filesystem
  - Feasible on VM
- Multiple problems, however
  - Understanding guest filesystem is far from trivial
    - Hyper-V ?
  - Require good understanding on OS & FS
  - Encrypted disk?

VM-specific solutions (2)

- Extract out information from memory
  - Feasible on VM
  - VM instrospection problem
- Multiple problems
  - VM introspection always requires to know OS first, not the other way
  - Signature based on code & data value depends on compiler

Part II

- Problems of debugger in malaware analysis
- UFO solution
  - Design & Implementation
  - Main features
- Live demo
- Discussions
- Conclusions
- Q & A

UFO solution

- Look into where nobody looks at that yet!
  - Network?
  - Memory?
  - Filesystem?

Anything else?

CPU context

- Characteristic of OS in protected mode
  - Memory segments → segment registers
    - CS, DS, SS, ES, GS, FS
  - Global Descriptor Table (GDT): segment information
    - Segment's selector, base & limit
  - Local Descriptor Table (LDT)
  - Interrupt Descriptor Table (IDT)
  - Task register (TR)
  - Features
    - 64-bit, Non-Executable, Fast-syscall

CPU parameters

- Segment parameters: CS, DS, ES, FS, GS, SS
  - Selector, Base, Limit
  - Distinguish by ring levels: CS0.Sel, CS0.Base, CS0.Limit, CS3.Sel, CS3.Base, CS3.Limit, ...
- Task Register (TR) parameter
  - Selector, ~~Base~~, Limit (ring 0 only)
  - TR.Selector, TR.Limit
- Global Descriptor Table (GDT) parameter
  - ~~Base~~, Limit (ring 0 only)
- Interrupt Descriptor Table (IDT) parameter
  - ~~Base~~, Limit (ring 0 only)
- Feature parameters
  - 64-bit, Fast-Syscall, Non-Executable (CR4 & MSR_EFER values)

CPU params – Example (1)

- GDT.limit
  - Windows: 0x3ff
  - Linux: 0xff
- CS0.Selector & CS3.Selector
  - Sun Solaris: 0x158 & 0x168
  - Haiku: 0x8 & 0x1b
- DS3.Limit
  - OpenBSD: 0xcfbfdfff (for W^X trick)
  - Everybody else: 0xffffffff

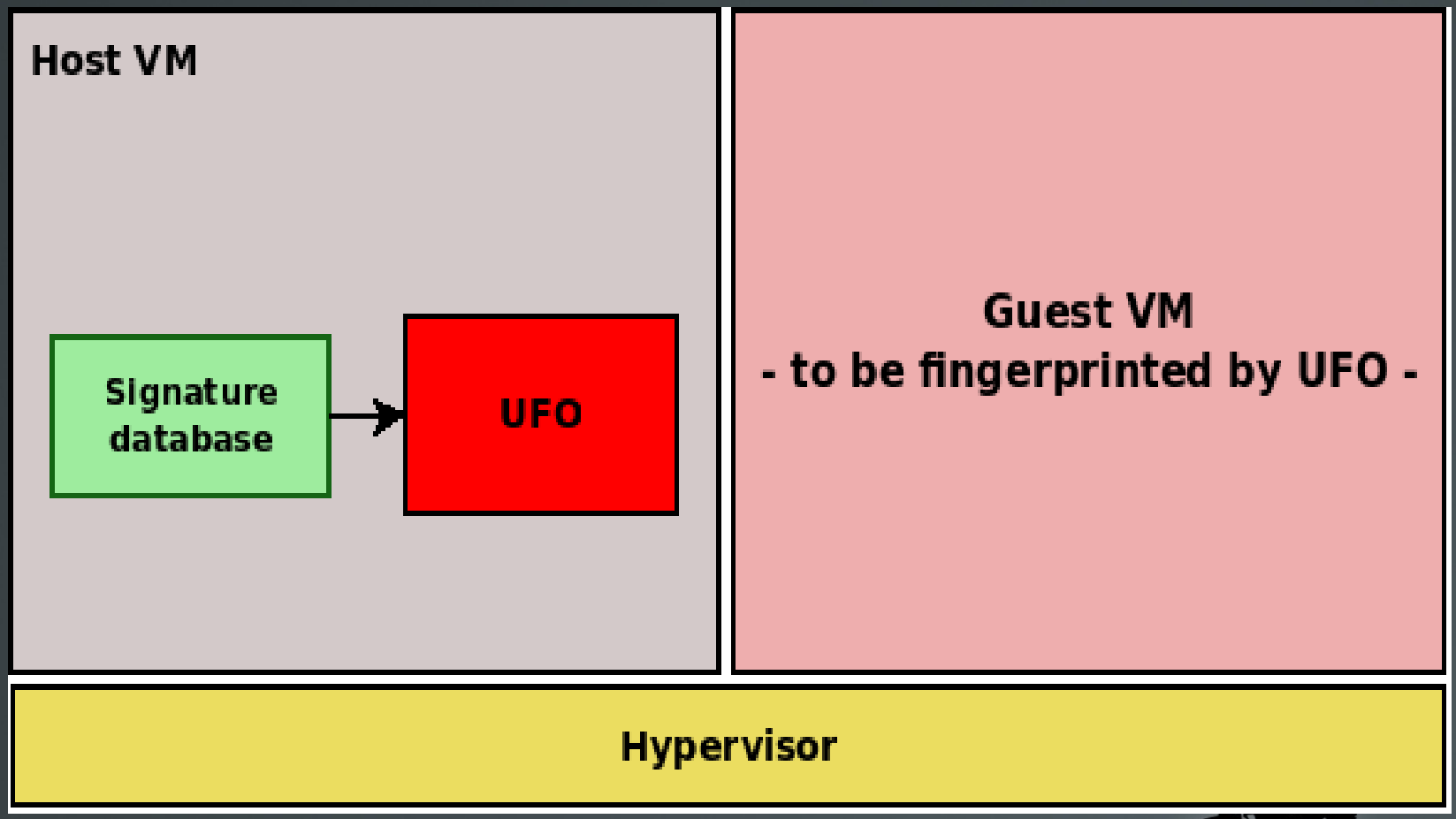
CPU params – Example (2)

- Features
  - Fast-syscall (Sysenter)
    - Windows XP+, Linux 2.6+
    - *BSD: None
  - Non-Executable (NX-bit)
    - Windows XP-SP2+
    - Linux 2.6.14+

OS signagture for UFO

- Retrieve CPU params can be done by UFO
  - All hypervisors should provide interface for this
    - Xen, KVM, Qemu, Hyper-V
    - VMWare with VMSafe API
- CPU params of OS represent OS!
  - OS variant, OS version, ServicePack
  - OS signature!

UFO architecture



Generate OS signature?

Generate OS signature (1)

- Run related OS inside QEMU
- Query for CPU context at run-time
  - Done from outside
  - Preodically inquiry for new CPU context
    - Tight loop
    - Save new CPU context
    - Automatically generate OS signature from gathered CPU context
- Pros: easy to implement
- Cons: miss CPU context

Generate OS signature (2)

- Run related OS inside QEMU
- Instrument QEMU to profile OS to create OS signature
  - Instrument instructions that can change OS context
    - mov, int, iret, sysenter, sysexit, syscall, sysret, jmp far, call far
    - lds, les, lfs, lgs
    - lgdt, lidt, ltr
    - wrmsr, loadall
  - Instrument system events modifying OS context
    - Task-switching
    - Interrupt handling
  - Gather all the context value and generate signature automatically

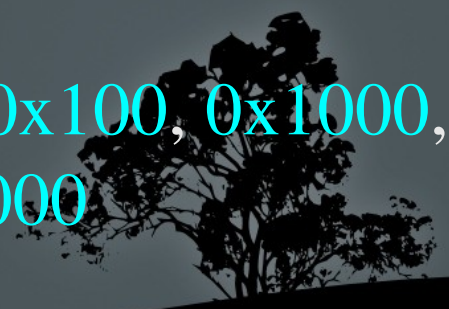
Signature on GDT table

- The only exception on signature relying on memory content
  - List all of non-zero selectors of GDT table
  - Optional to avoid being fooled by attacker from inside guest VM

gdt: [3ff, 8, 10, 1b, 23, 28, 30, 3b, 43, 50, 58, 60, 68, 70, 78, 80, 88]

Invariant OS params

- To combine similar params into one
  - So the signature is as small as possible
- Only focus on segment params
  - ~~Segment.Sel~~, Segment.Base, Segment.Limit
- Simple invariant patterns
  - Special ”constant” values
    - Base of 0, Limit of 0xFFFFFFFF
  - Base values with more than 5 different values counted as ”anything”
  - Limit has same modulo on 0x10, 0x100, 0x1000, 0x10000, 0x1000000, 0x10000000



Sample OS signature

- Text file of signatures
  - No special tool is needed to edit signatures
- JSON-like format
  - Intuitive and easy to modify

Option to modify OS params

- OS might have options to change its OS params
  - Linux PAX UDEREF
    - Dedicate last part of memory segments to trap NULL-dereference exploitation
  - Exec-Shield (Ubuntu, RedHat)
    - Turn off NX & Fast-syscall feature
- Support OS option in signature
  - ”option” signature
    - Same syntax as ”normal” OS signature
    - Reduce the number of OS signatures by applying the ”option” signature on top of other OS signatures

Sample OS signature + option

Fingerprint OS

- Retrieve CPU context from guest OS
  - Using interface provided by hypervisors
- Match CPU params against signature database
  - Match against every signature, one by one
  - Fuzzy matching method
    - +1 for one param matching, +0 for unmatched param
    - If ”options” param is specified, match signature with and without options
  - Highest score is the right OS!

Demo

Advantages

- Fuzzy matching can report best-match OS
  - Good in case signature is missing
- Performance: Lightening fast
- Accuracy is high
- No OS tweaking available to modify OS context
  - We cover possible problems with ”options” signature
- Signature is hypervisor independent

Problems of UFO

- Attacker might have full control on guest OS from inside to defeat UFO
  - Selectively modify OS context
    - Need to mess with OS kernel
  - Modify OS code (source?) to change OS context
- A range of OS versions might have OS context unchanged
  - Report as a whole

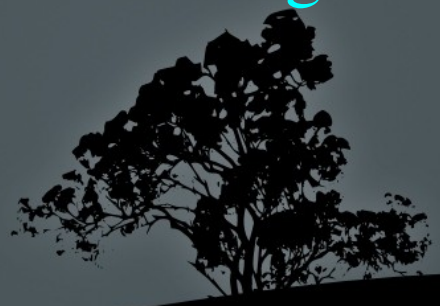
Detecting future OS

- Feasible because OS continuosly evolves
  - Thread Environment Block (TIB) is addressed by FS segment in Windows OS
    - Containing information about process
      - Windows XP
        - FS0.Limit = 0x1FFF
      - Windows Vista
        - FS0.Limit = 0x2128
      - Windows 7
        - FS0.Limit = 0x3748

Conclusions

- UFO is a new OSF tool specially designed to fingerprint the OS running inside VM
  - Extremely fast
  - Highly accuracy
  - Resistance against OS tweaking
  - Hypervisor independent
- Pre-release available under LGPL license at

  http://force.vnsecurity.net/download/aquynh/ufo-0.1.tgz



UFO: Operating System Fingerprinting for Virtual Machine

Q & A

NGUYEN Anh Quynh <aquynh @ gmail.com>