Security of Cloud Computing

Fabrizio Baiardi
f.baiardi@unipi.it

Syllabus

- Cloud Computing Introduction
  - Definitions
  - Economic Reasons
  - Service Model
  - Deployment Model
- Supporting Technologies
  - Virtualization Technology
  - Scalable Computing = Elasticity
- Security
  - New Threat Model
  - New Attacks
  - Countermeasures

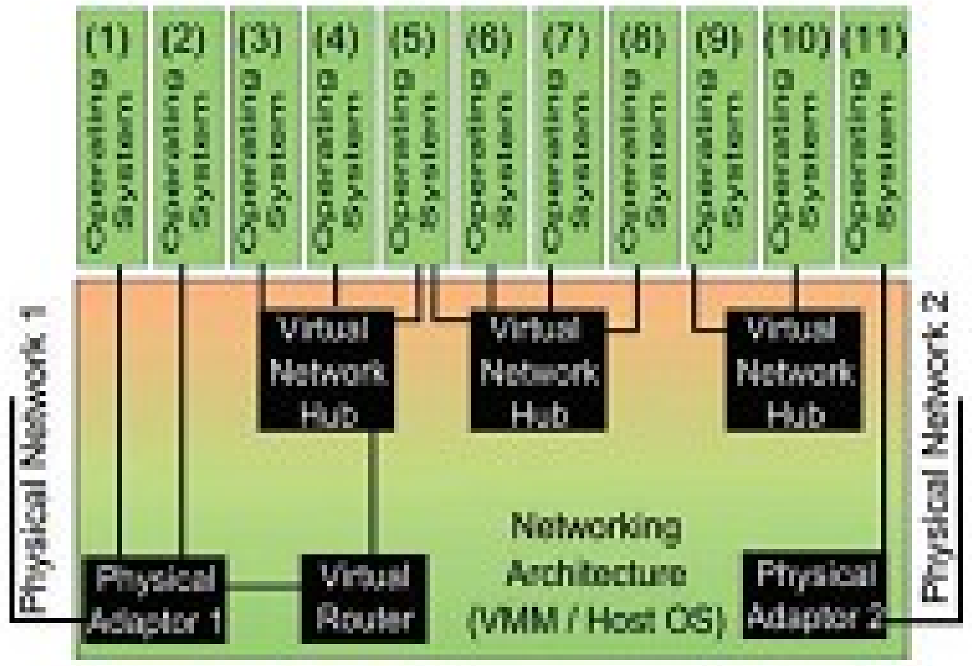
A first set of VM and VMMM Vulnerabilities and of attacks

VM detection

How a user can detect that the application is running on a VM as a first step to attack the VM itself?

- Detect VM Artifacts in Processes, File System, and/or Registry
- Look for VME Artifacts in Memory
  - The Red Pill (Matrix)
- Look for VME-specific virtual hardware
- Look for VME-specific processor instructions and capabilities

A typical virtual architecture



Ten VMs connected to virtual networks in various arrangements.

9+10 = isolated network

3-8 = connected network

VM detection – Artifacts in Process etc.

Some VMEs insert elements into the Guest that can be easily found

- Running processes or services
- Files and/or directories
- Specific registry keys
- Some Phatbot malware specimens use this technique
- In a VMware Workstation WinXP Guest:
  - Running “VMtools” service
  - Over 50 different references in the file system to “VMware” and vmx
  - Over 300 references in the Registry to “VMware”
- This method is of limited utility, easily fooled
  - Rootkits tweak the operating system to hide artifacts from users
  - Similar techniques could be applied to hide VME

VM detection – Artifacts in Memory

The Guest system memory map has some differences from the Host memory map

- Strings found in memory
  - By dumping RAM of VMware Workstation WinXP Guest
  - Over 1,500 references to “VMware” in memory
  - Rather a heavy-weight approach, but could be refined to focus on specific regions
- Some critical operating system structures located in different places
- Much quicker, easier, and hard to fool without redesign of VME

One particular memory difference is the location of the Interrupt Descriptor Table (IDT)

- On Host machines = it is typically low in memory
- On Guest machines = it is typically higher in memory
- Cannot be the same, because the processor has a register pointing to it (IDTR)

Memory technique is usable across different VMEs (VirtualPC and VMware) and more difficult to fool

VM detection – Memory – Blue Pill - 1

- In November 2004, Joanna Rutkowska released a tool, "The Red Pill" that reliably detects virtual machine usage without looking for file system artifacts

  http://invisiblethings.org/index.html#redpill
- This tool runs a single machine language instruction, SIDT

  "Store Interrupt Descriptor Table"
- This instruction, which can be run in user mode, takes the location of the Interrupt Descriptor Table Register (IDTR) and stores it in memory where it is analyzed

VM detection – Memory – Blue Pill - - 2

- On VMware guest machines, the IDT is typically located at 0xffXXXXXX
- On VirtualPC guests, it is located at 0xe8XXXXXX
- On host operating systems, it is located lower than that,
  - 0x80ffffff (Windows)
  - 0xc0ffffff (Linux)
- The Red Pill merely looks at the first byte returned by SIDT
  - If it’s greater than 0xd0, you've got a virtual machine
  - If it is less than or equal to 0xd0, you are in a real machine

Look for VME-Specific Virtual Hardware

- VME introduces virtualized hardware
  - Network
  - USB controller
  - Audio adapters
- Some of these have distinct fingerprints
  - MAC addresses on NICs
  - USB controller type
  - SCSI device type
- Also, anomalies in the way the Guest system clock is updated
- Easy-to-write code, but likely easily fooled

Look for VME-Specific Virtual Hardware

Linux version of Doo

- Simple shell script looks for “VMware” located in:
  - /proc/iomem
  - /proc/ioports
  - /proc/scsi/scsi
  - dmesg command (print kernel ring buffer; holds boot messages and related logs from kernel)
- Also looks in dmesg output for “BusLogic BT-958” and “pcnet32” - These are known VMware devices

Windows version of Doo:

- Uses Windows Scripting Host to read 2 registry keys associated with SCSI to look for “VMware”
- Uses WSH to read 2 other registry keys associated with specific class ID of VMware virtualized hardware

Look for VME-Specific Processor Instructions and Capabilities

Some VMEs introduce “extra” machine-language instructions beyond the standard x86 instruction set to foster Guest-to-Host communication or for other virtualization issues

- Code could play a non-standard x86 instruction used by VMware, VirtualPC, or Xen into processor to see if it rejects it or handles it
- Alternatively, code could look for unusual processor behavior associated with “normal” machine language instructions

Look for VME-Specific Processor Instructions and Capabilities

To detect VirtualPC, VMDetect:

- Registers its own handler for invalid OpCodes
- Runs a VirtualPC-specific non-standard IA32 instruction
- If the processor runs the instruction, it is VirtualPC
- If the handler for invalid OpCodes is called, it it’s a real machine

VMware Detection with VMDetect at the Machine Language Level

- The machine language looks for the VMware guest-to-host channel, by checking for a strange processor property of VMware guests.
- This code attempts to invoke the VMware guest-to-host communication channel created by overloading the functionality of a specific x86 instruction, “IN.”
- The IN instruction is used to read a byte, word, or dword of data from an I/O port. It has two parameters:
  - the register that is the data destination and
  - the port is to be accessed. This number is placed in the processor register DX before the instruction is executed.
- VMware monitors any use of the IN instruction, and captures any I/O destined for a specific port number (0x5658) but only when the value of another processor register, EAX, is a very specific “magic” number.

VMware Detection with VMDetect at the Machine Language Level

- The code detects VMware by first loading EAX with this magic value, “VMXh”. A specific “command” for VMware to process (in this case 0x0A or decimal 10) is loaded in ECX and parameter data (in this case, 0) is loaded into register EBX. The special port number (0x5658 which also stands for the characters “VX”) is loaded into register EDX. Then the code executes IN.
- On a non-virtual machine, this will cause a processor exception and trigger specifically provided exception handling code within the software.
- On a VMware machine, the instruction will be monitored and allowed to succeed without error by VMware which will then change the values in the processor’s registers before returning execution to the code.
- The end result will be that the magic value, “VMXh” will be moved into register EBX. The code then compares EBX to this value and continues on, knowing for certain that it is running in a virtual environment.

VMware Detection with VMDetect at the Machine Language Level

```
MOV EBX,0
MOV ECX,0A
MOV EDX,5658 <-- "VX"
IN EAX,DX <-- Check for VMWare
CMP EBX,564D5868
```

- First, the EAX register is loaded with the “magic value” to the use the communication channel between the real and the virtual machine(“VMXh”)
- ECX is loaded with a command value (0x0A which is used to request VMware version information from the host)
- Any parameters needed for the command (in this case there are none) are loaded in EBX
- Finally, the IN instruction (used for port I/O) is used, which would normally attempt to load data from port 0x5658 (“VX”)
- If we are outside VMware, a privilege error occurs
- If we’re inside VMware, the magic value (VMXh) is moved to register EBX; otherwise it is left at 0
- Based on the version values returned, we can determine the VMware product

Compatibility is not Trasparency

- As pointed out in

  Compatibility is Not Transparency: VMM Detection Myths and Realities, Garfinkel, Adams , Warfield and Franklin, 11° Workshop on Hot Topics in Operating Systems , HotOS-XI, May 2007

  it should be noticed that the goal of VM technology has been, till now, a minimal loss of performance due to the virtualization
- “The belief that VMM transparency is possible is based on a mistaken intuition that compatibility and performance imply transparency i.e. that once VMMs are able to run software for native hardware at native speeds, they can be made to look like native hardware under close inspection. This is simply not the case”
- Why bother about transparency?

CVE-2007-4496

VMWare ESX 3.0.1

- http://www.vmware.com/support/vi3/doc/esx-8258730-patch.html
- Found by Rafal Wojtczuk (McAfee)
- September 2007
- Guest OS can cause memory corruption on the host and *potentially* allow for arbitrary code execution on the host

CVE-2007-0948

Microsoft Virtual Server 2005 R2

- http://www.microsoft.com/technet/security/bulletin/ms07049.mspx
- Found by Rafal Wojtczuk (McAfee)
- August 2007
- Heap-based buffer overflow allows guest OS to execute arbitrary code on the host OS

CVE-2007-4993

Xen 3.0.3

- http://bugzilla.xensource.com/bugzilla/show_bug.cgi?id=1068
- Found by Joris van Rantwijk
- September 2007
- By crafting a grub.conf file, the root user in a guest domain can trigger execution of arbitrary Python code in domain 0.

Grub = GRand Unified Bootloader

Cloudburst -1

Combination of 3/4 bugs in the VMware emulated video device

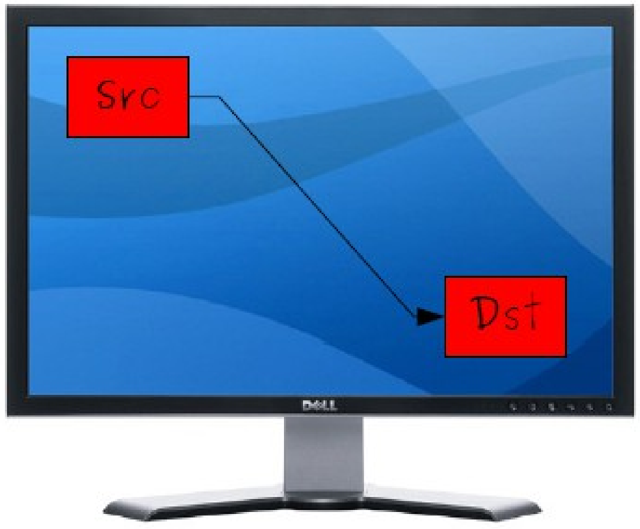
- Host memory leak into the Guest
- Host arbitrary memory write from the Guest
  - Relative
  - Absolute
- And some additional DEP friendly goodness

Reliable Guest to Host escape on recent VMware products

Cloudburst - 2

SVGA_CMD_RECT_COPY

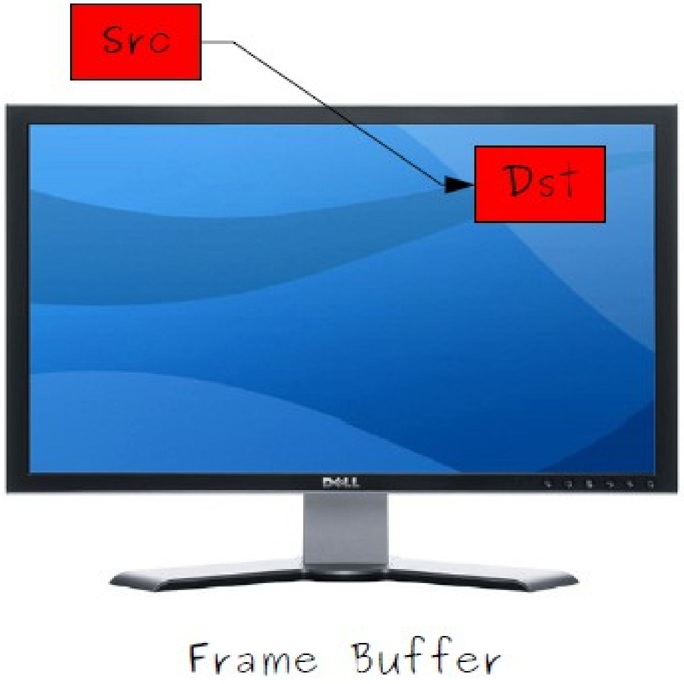
- Copies a rectangle in the Frame Buffer from a source X, Y to a destination X, Y



Frame Buffer

Cloudburst - 3

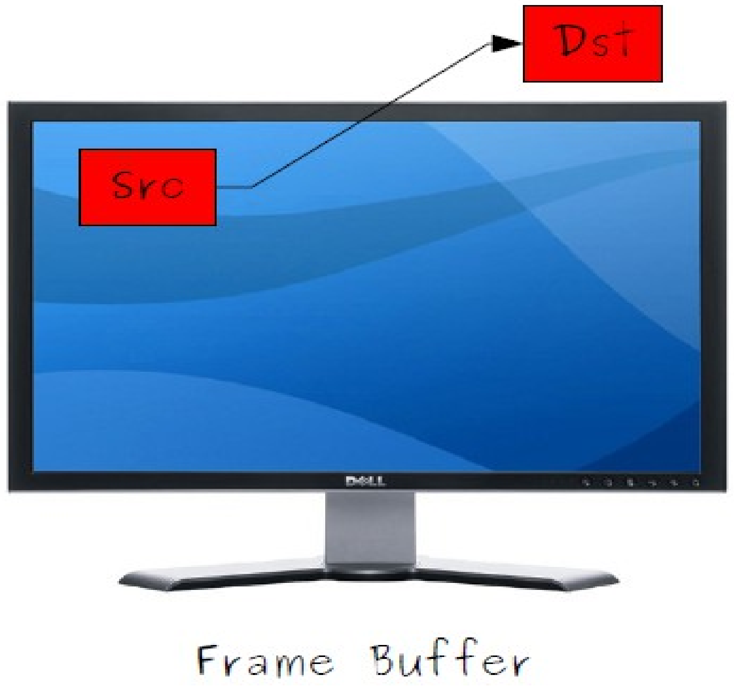
- Boundaries checks on the source location can be bypassed



Can be used to access information on the host

Cloudburst - 4

- Boundaries checks on the destination location can be bypassed (to a lower extent than source)



Can be used to update information on the host

Cloudburst - 5

SVGA_CMD_DRAW_GLYPH

- Draws a glyph into the frame buffer
- Requires `svga.yesGlyphs="TRUE"`

Cloudburst - 6

SVGA_CMD_DRAW_GLYPH

- There is no check on the X, Y where the glyph is to be copied

Cloudburst - 7

- To defeat Vista Data Execution Prevention Technology, the attack has 12 steps
- This confirms that in the case of clouds attacks are possible but they may require a larger number of steps than traditional ones
- Attack graphs can simplify the detection of attacks in this class

And now back to distinct vulnerabilities ….

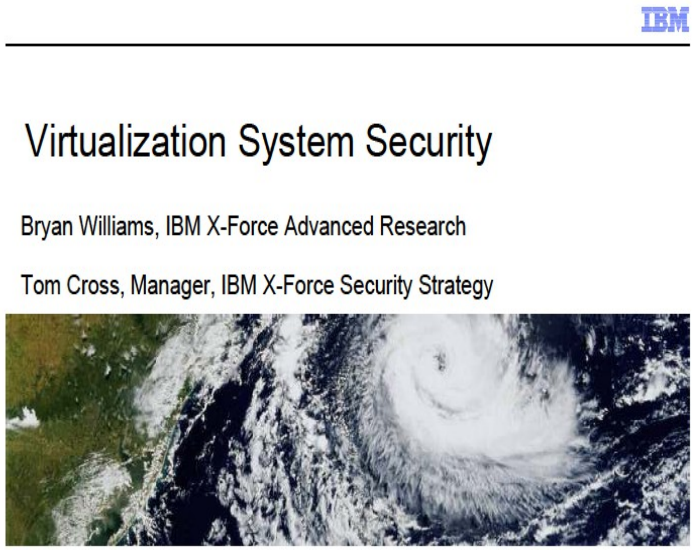
Statistics and some general attacks

- Virtualization System Security

  Bryan Williams, IBM X-Force Advanced Research

  Tom Cross, Manager, IBM X-Force Security Strategy

- Describes some statistics about vulnerabilities in virtualization components
- Discuss a first set of VM attacks


IBM

Virtualization System Security

Bryan Williams, IBM X-Force Advanced Research

Tom Cross, Manager, IBM X-Force Security Strategy

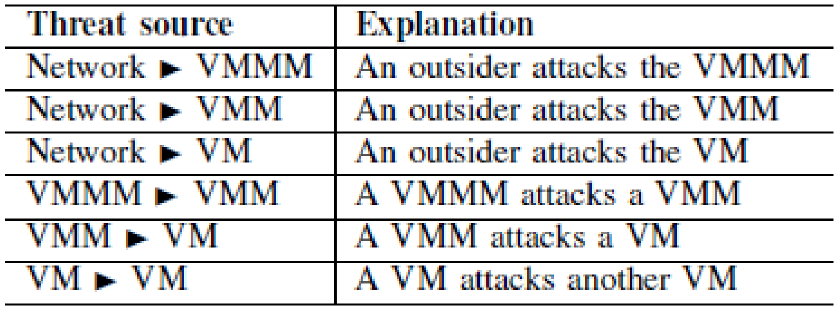
C:\Users\Hp\Documents\Magic Briefcase\cloud\riferimenti\VirtualizationSecurity.pdf

Other classification

Security Implications of Virtualization: A Literature Study

Andre van Cleeff, Wolter Pieters, Roel Wieringa

- Proposes a classification in terms of the attack enabled by a vulnerability



Threat source	Explanation
Network ► VMMM	An outsider attacks the VMMM
Network ► VMM	An outsider attacks the VMM
Network ► VM	An outsider attacks the VM
VMMM ► VMM	A VMMM attacks a VMM
VMM ► VM	A VMM attacks a VM
VM ► VM	A VM attacks another VM

VMMM = virtual machine monitoring and management

- It points out an important feature of virtualization: loss of monotonicity

  Virtualization technology causes a server's history to stop being a straight line. Instead it becomes a graph, where branches are made on replication and copy operations, and a previous state can be reached when a restore is performed. Data cannot be deleted easily, there can be many copies and the VM can be restored to an earlier version.

Loss of monocity

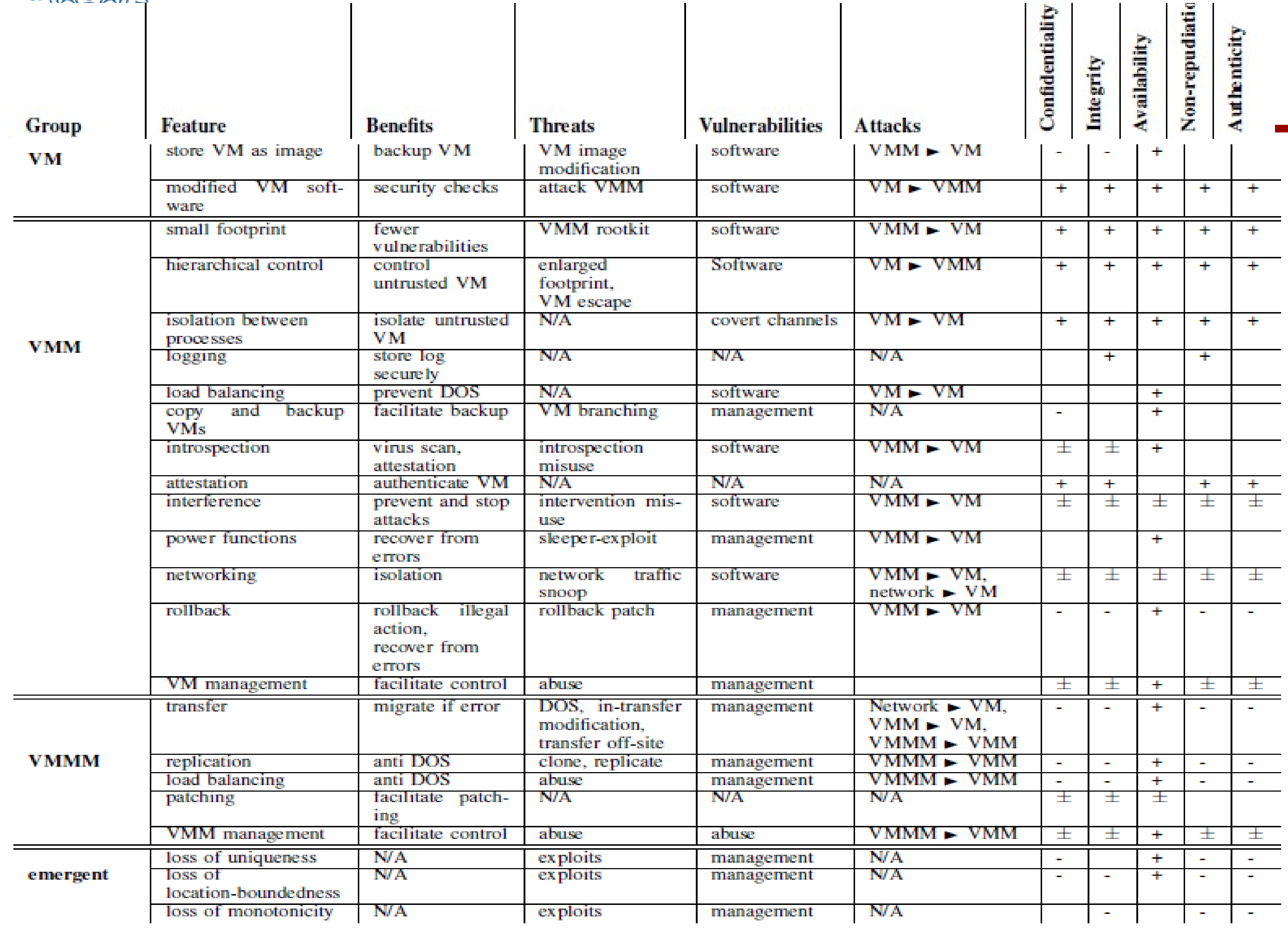
a) log of a virtual machine ???

b) physical location when several copies of the same machine exist

c) patching of virtual machine. Reverting to an old version may remove the patch. Several versions may be run

- One patched
- One unpatched

d) if several versions exist, all must be located



Group	Feature	Benefits	Threats	Vulnerabilities	Attacks	Confidentiality	Integrity	Availability	Non-repudiatio	Authenticity
VM	store VM as image	backup VM	VM image modification	software	VMM ► VM	-	-	+		
	modified VM software	security checks	attack VMM	software	VM ► VMM	+	+	+	+	+
VMM	small footprint	fewer vulnerabilities	VMM rootkit	software	VMM ► VM	+	+	+	+	+
	hierarchical control	control untrusted VM	enlarged footprint, VM escape	Software	VM ► VMM	+	+	+	+	+
	isolation between processes	isolate untrusted VM	N/A	covert channels	VM ► VM	+	+	+	+	+
	logging	store log securely	N/A	N/A	N/A		+		+	
	load balancing	prevent DOS	N/A	software	VM ► VM			+		
	copy and backup VMs	facilitate backup	VM branching	management	N/A	-		+		
	introspection	virus scan, attestation	introspection misuse	software	VMM ► VM	±	±	+		
	attestation	authenticate VM	N/A	N/A	N/A	+	+		+	+
	interference	prevent and stop attacks	intervention misuse	software	VMM ► VM	±	±	±	±	±
	power functions	recover from errors	sleeper-exploit	management	VMM ► VM			+		
	networking	isolation	network traffic snoop	software	VMM ► VM, network ► VM	±	±	±	±	±
	rollback	rollback illegal action, recover from errors	rollback patch	management	VMM ► VM	-	-	+	-	-
	VM management	facilitate control	abuse	management		±	±	+	±	±
VMMM	transfer	migrate if error	DOS, in-transfer modification, transfer off-site	management	Network ► VM, VMM ► VM, VMMM ► VMM	-	-	+	-	-
	replication	anti DOS	clone, replicate	management	VMMM ► VMM	-	-	+	-	-
	load balancing	anti DOS	abuse	management	VMMM ► VMM	-	-	+	-	-
	patching	facilitate patching	N/A	N/A	N/A	±	±	±		
	VMM management	facilitate control	abuse	abuse	VMMM ► VMM	±	±	+	±	±
emergent	loss of uniqueness	N/A	exploits	management	N/A	-		+	-	-
	loss of location-boundedness	N/A	exploits	management	N/A	-	-	+	-	-
	loss of monotonicity	N/A	exploits	management	N/A		-		-	-

Reincarnation Attack

Controlled Reincarnation Attack to Subvert Digital Rights Management
F. John Krautheim and Dhananjay S. Phatak

- A licensing mechanism based solely on local state (system time, processor identification number) is defeated
  - by capturying the state of the VM immediately after activating a license
  - by restarting the VM from the same point at any reactivation
- since the VM can be contained in a file, it can easily be distributed in the pristine state and multiple copies of the software can be used simultaneously
- this attack is easily defeated by using an ongoing communication with an activation server to ensure that the license remains valid throughout the session and life of the product.
- A more complex attack captures the communications between the VM and activation server and performs a replay attack. Since the VM is started in a known state, all communications should be exactly the same every time, so that the malicious middleware fakes the software into believing it is talking to a real activation server.

References

- Chow, R.; Golle, P.; Jakobsson, M.; Shi, E.; Staddon, J.; Masuoka, R.; Molina, J. Controlling data in the cloud: outsourcing computation without outsourcing control. Proceedings of the 2009 ACM Workshop on Cloud Computing Security (CCSW 2009); 2009 November 13; Chicago, IL. NY: ACM; 2009; 85-90.
- Manadhata, P; Wing, J; , "An Attack Surface Metric," *Software Engineering, IEEE Transactions on* , vol.PP, no.99, pp.1, 0 doi: 10.1109/TSE.2010.60