Cloud Computing Vulnerability

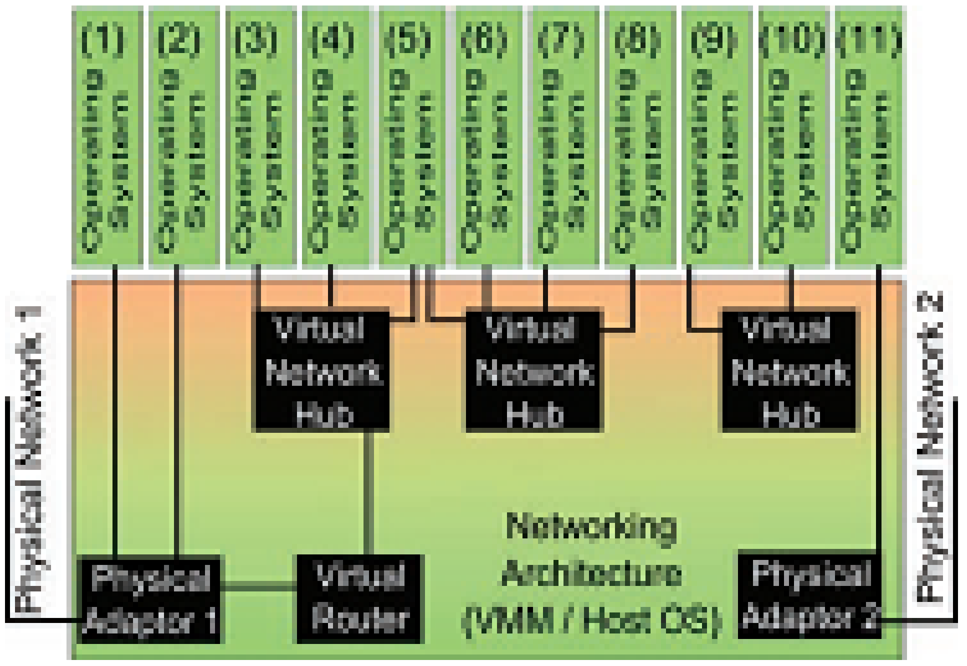
Fabrizio Baiardi
f.baiardi@unipi.it

Syllabus

- ICT Security
- Cloud Computing
- Security of Cloud Computing
  - New Threat Model
  - New Attacks
  - Countermeasures

A first set of VM and VMMM Vulnerabilities and of attacks

A typical virtual architecture



Ten VMs connected to virtual networks in various arrangements.

9+10 = isolated network

3-8 = connected network

VM detection

How a user can detect that an application is running on a VM as a first step to attack the VM itself or to obfuscate a malware in a testing environment

As discussed in the following this is a resurrection of the old debate transparency vs compatibility resurrects

Detection strategies

- 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

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 differs 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
  - A heavy-weight approach, 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 memory difference is the Interrupt Descriptor Table (IDT) location

- On Host machines = it is typically low in memory
- On Guest machines = it is typically higher in memory
- Distinct because the processor register IDTR points to it

  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"

- The instruction can be run in user mode, it 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

The IDT is typically located at

• 0xffXXXXXX	on VMware guest machines,
• 0xe8XXXXXX	on VirtualPC guests,
• 0x80ffffff	on Windows,
• 0xc0ffffff	on Linux

The Red Pill merely looks at the first byte returned by SIDT

a) If it’s greater than 0xd0, you've got a virtual machine

b) 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 (a second sometime is longer than a second)
- Easy-to-write code, but likely easily fooled

Look for VME-Specific Virtual Hardware

Linux version of ScoopyDoo

- 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 ScoopyDoo :

- 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.
- Guest to host channel used for network communications
- This code attempts to invoke the VMware guest-to-host channel created by overloading the functionality of IN an x86 instruction,
- IN reads a byte, word, or dword from an I/O port, two parameters:
  - the register that is the data destination and
  - the port 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 fragment to detect Vmware loads EAX with this magic value, “VMXh”. Then it loads into EBX a specific “command” for VMware (0x0A or decimal 10) and then loads parameter data (0) into register EBX. The special port number (0x5658 which stands for the characters “VX”) is loaded into register EDX. Then, the code executes IN.

- On a non-virtual machine, a processor exception is raised that triggers 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

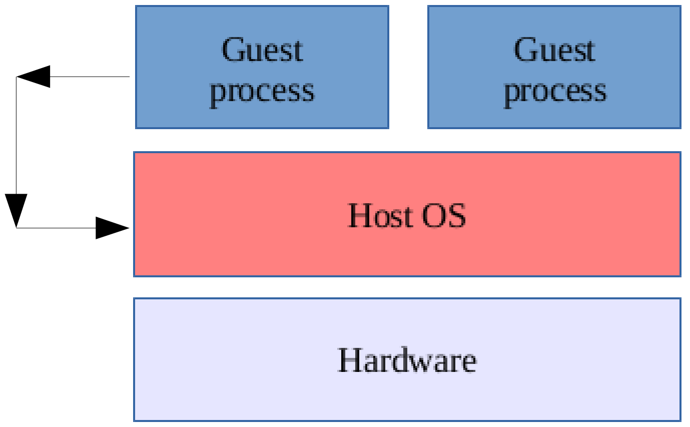
Compatibility vs Trasparency

Compatibility is Not Transparency: VMM Detection Myths and Realities, HotOS-XI, May 2007

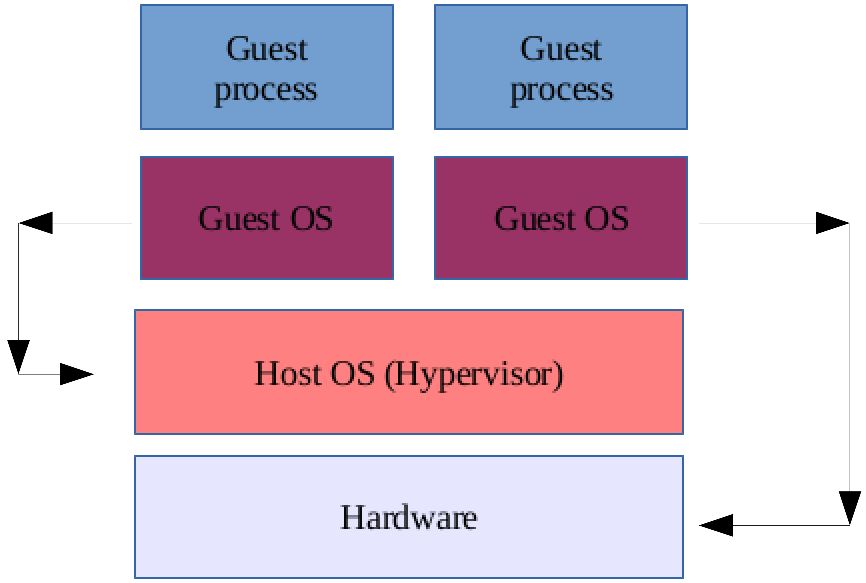
- the goal of VM technology has been, till now, a minimal loss of performance due to the virtualization
- The belief that VMM transparency is possible is based on a mistaken intuition that compatibility and performance imply transparency = once VMMs are able to run software for native hardware at native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

- Why bother about transparency?
- Compatibility is not isolation and is not security

VM vulns



OS-level virtualization



Hardware virtualization

VM vulns: 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

VM vulns: 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
- Domain 0 = Xen Hypervisor

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 -1

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

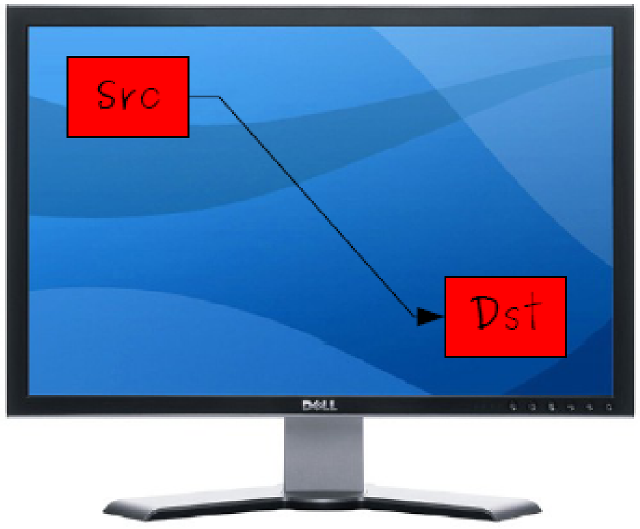
- Host memory leak into the Guest
- Host arbitrary memory write from the Guest
  - Relative
  - Absolute
- And some additional DEP friendly goodness

Reliable Guest to Host escape on recent VMware products

Cloudburst - 2

SVGA_CMD_RECT_COPY

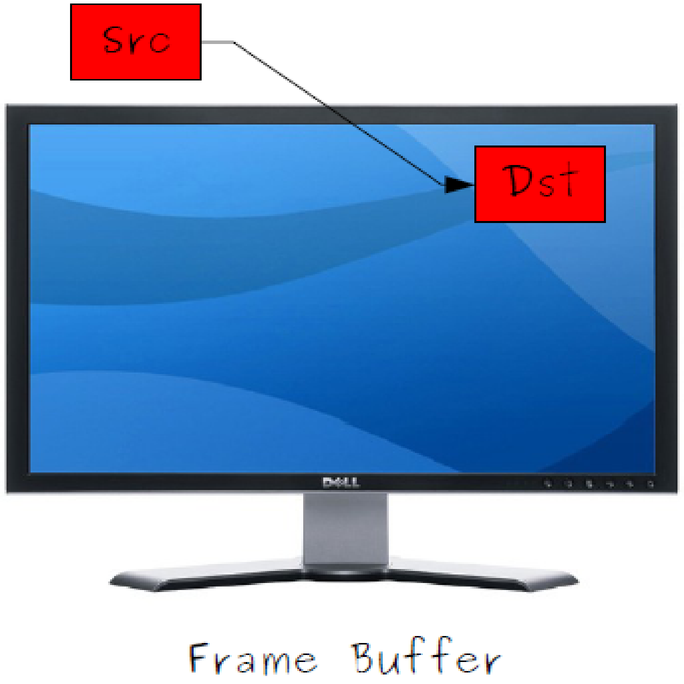
- Copies a rectangle in the Frame Buffer from a source X, Y to a destination X, Y



Frame Buffer

Cloudburst - 3

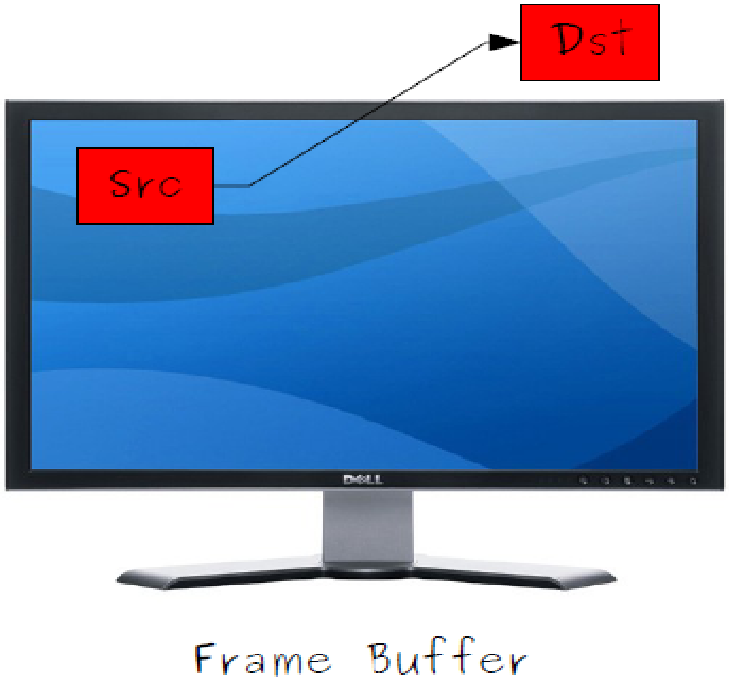
- Boundaries checks on the source location can be bypassed



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”`

Virtual Screen

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 intrusion has 12 steps = composes 12 elementary attacks
- Data Execution Prevention Technology merges hardware page protection and software controls on exception handling
- This confirms that clouds attacks are possible but require a larger number of steps than traditional ones
- Attack graphs simplify the detection of this class of attacks

And now back to distinct vulnerabilities ….

A more recent vuln …

A critical vulnerability in VMware's vCenter management product allowed any old bod on the same network to remotely create an admin-level user, research by Guardicore Labs has revealed. The astonishing vuln (CVE-2020-3952) was rated by VMware as CVSS v3 10.0, the highest level.

Guardicore researcher JJ Lehman told The Register: "You have to be network accessible but you don't have to be authenticated in any way to pull this off. Which means as an attacker who has breached the perimeter of a network you essentially control everything on their VMware hosts."

VMWare issued an advisory note and patch on 9 April that explained that a "malicious actor with network access to port 389 on an affected vmdir deployment may be able to extract highly sensitive information such as administrative account credentials

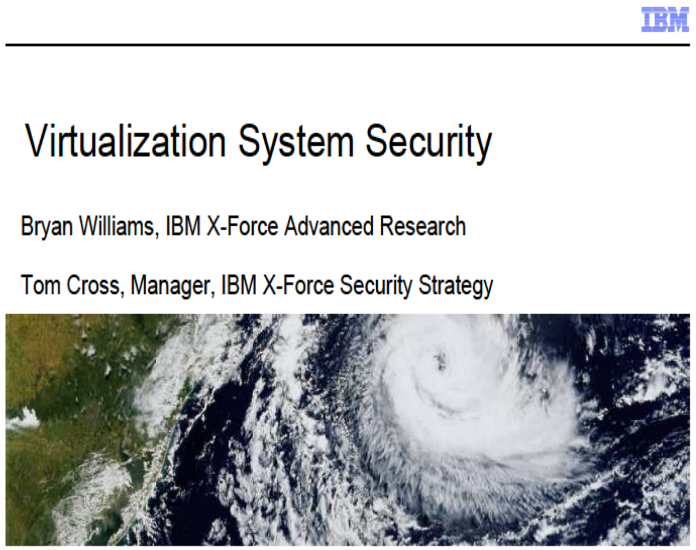
Statistics and some general attacks

- Virtualization System Security

  Bryan Williams, IBM X-Force Advanced Research

  Tom Cross, Manager, IBM X-Force Security Strategy

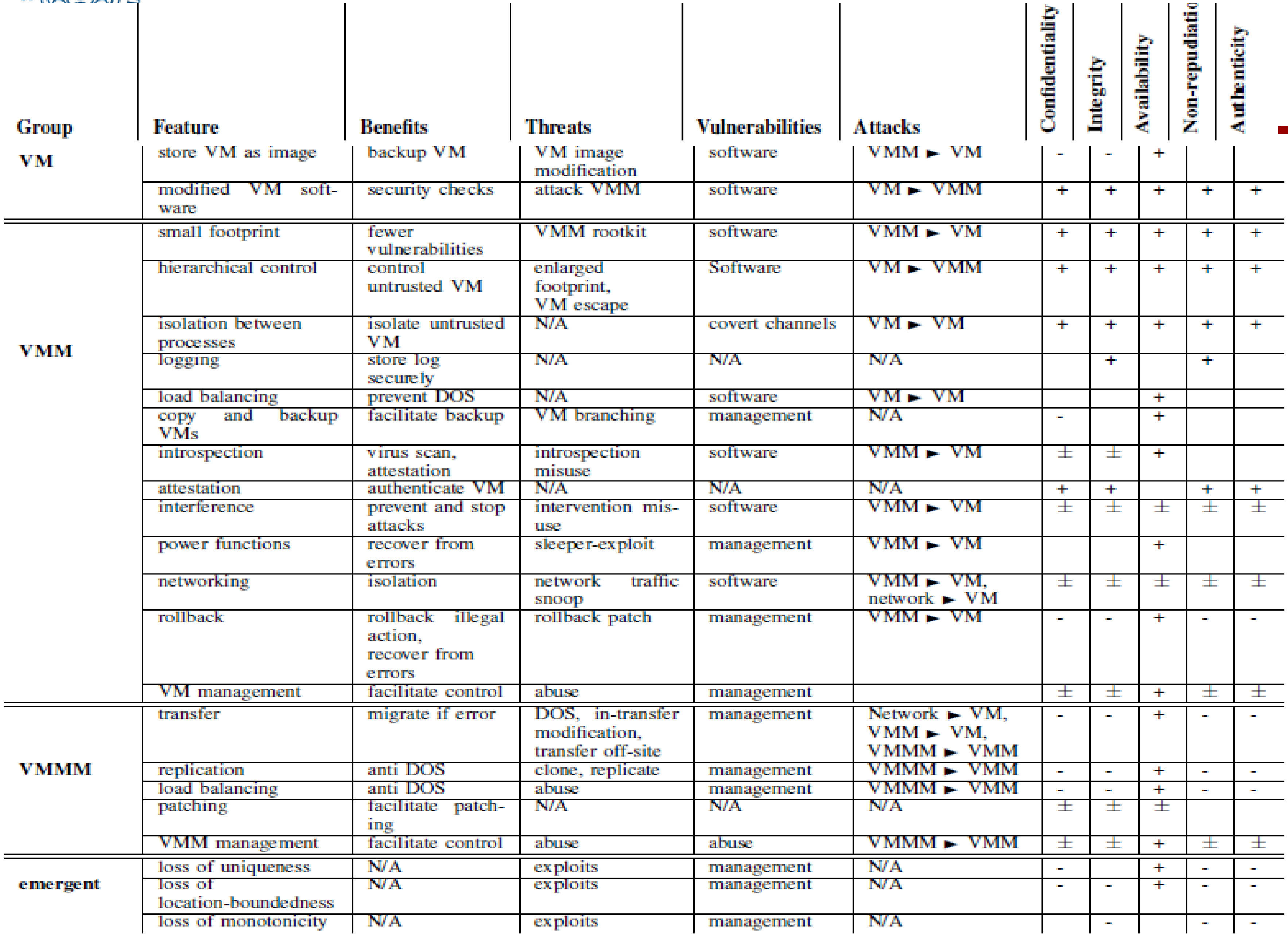
- Describes some statistics about vulnerabilities in virtualization components
- Discuss a first set of VM attacks


IBM

Virtualization System Security

Bryan Williams, IBM X-Force Advanced Research

Tom Cross, Manager, IBM X-Force Security Strategy



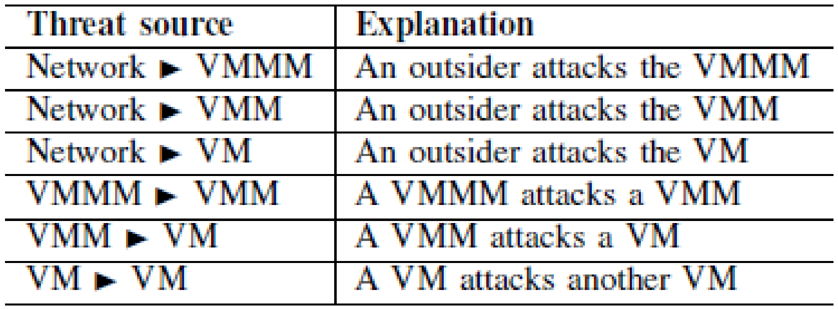
Group	Feature	Benefits	Threats	Vulnerabilities	Attacks	Confidentiality	Integrity	Availability	Non-repudiatio	Authenticity
VM	store VM as image	backup VM	VM image modification	software	VMM ► VM	-	-	+		
	modified VM software	security checks	attack VMM	software	VM ► VMM	+	+	+	+	+
VMM	small footprint	fewer 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		-		-	-

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 monotonicity

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

Reincarnation Attack: A malicious usage of loss of monoticity

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

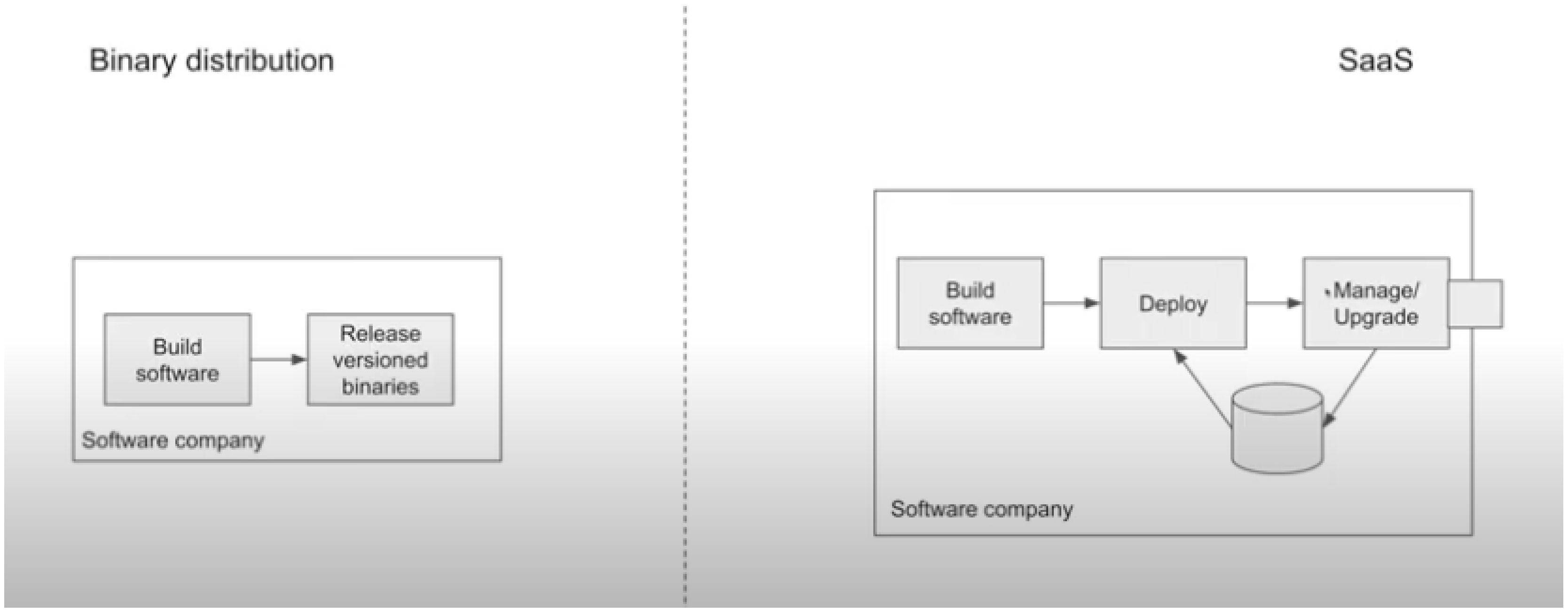
- A licensing mechanism based solely on local state (system time, processor identification number) is defeated

  a) by capturying the VM state immediately after activating a license

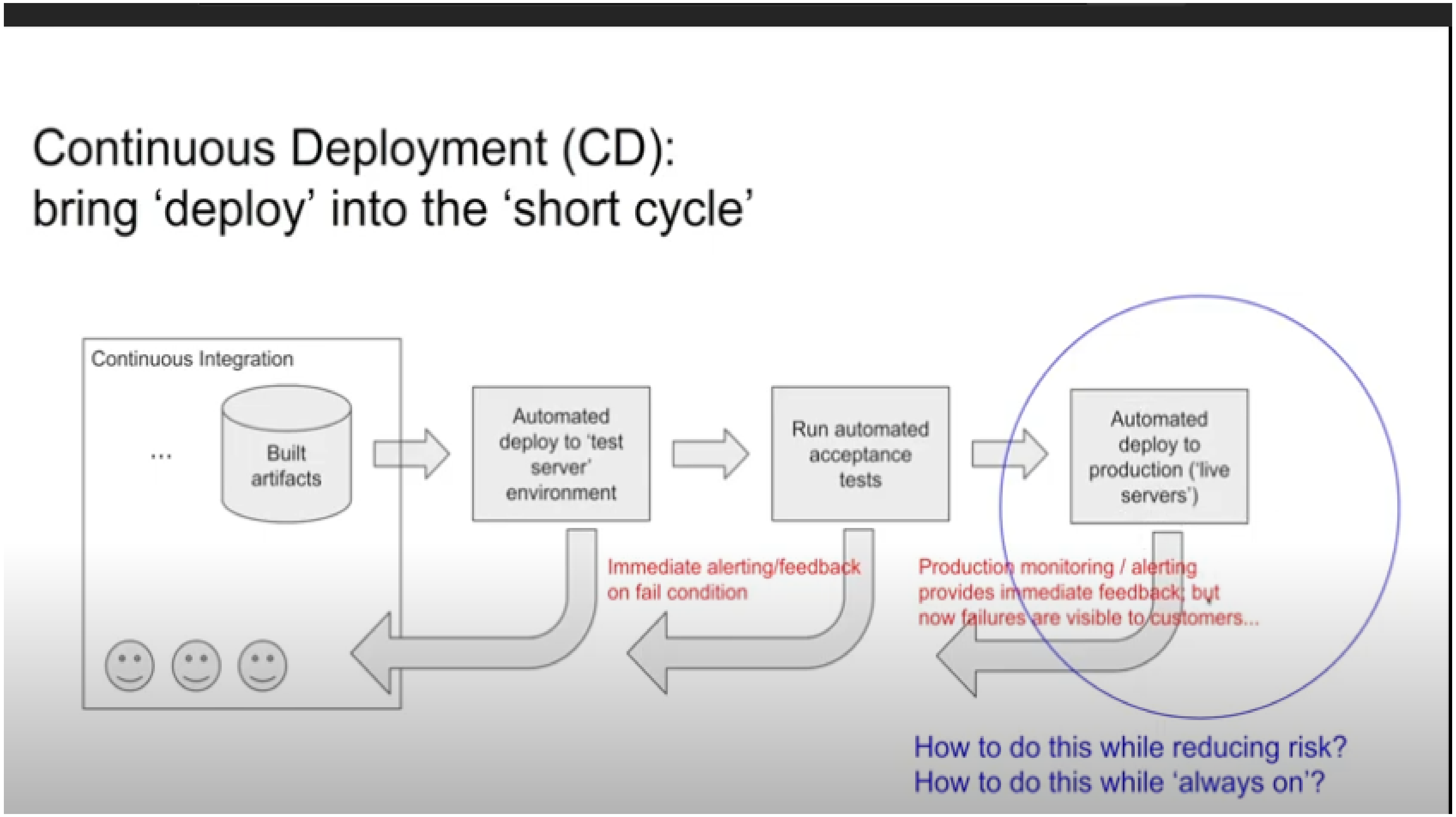
  b) by restarting the VM from the same point at any reactivation
- Since the VM is stored in a file, it can easily be distributed in the pristine state and multiple copies of the software can be used simultaneously
- It may be defeated by communicating with an activation server to ensure that the license remains valid throughout the session and product life
- Another attack capture and replays the communications between the VM and activation server. Since the VM starts in a known state, every time the communications are exactly the same so the malicious middleware fakes the software into believing it is talking to a real activation server.

Roll Deployment: Monotonicity and Errors

Saas completely changes the framework for software companies



Roll Deployment: Monotonicity and Errors

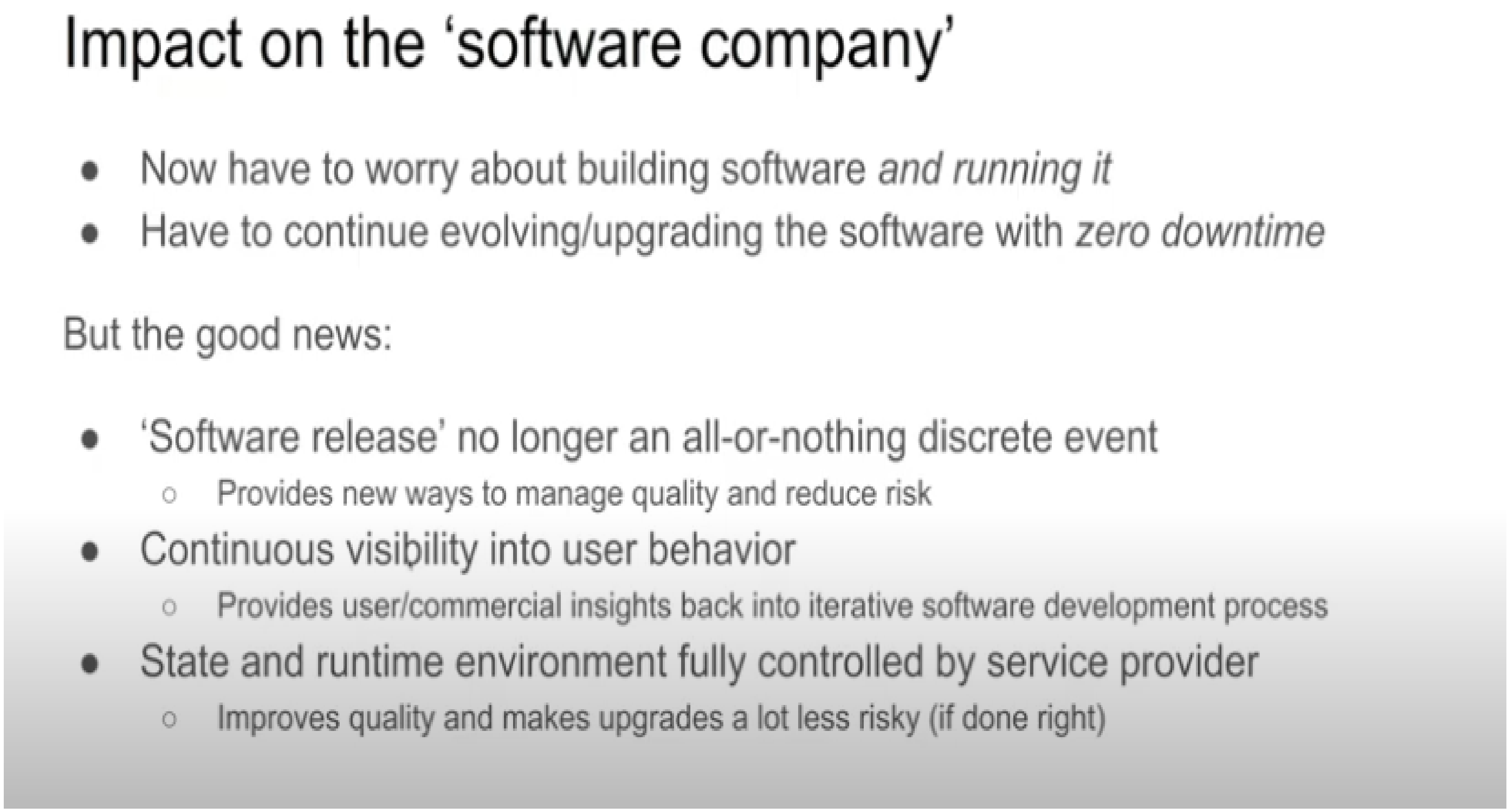


Continuous Deployment (CD):
bring 'deploy' into the 'short cycle'

Continuous Integration
...
Built artifacts
Automated deploy to 'test server' environment
Run automated acceptance tests
Automated deploy to production ('live servers')
Immediate alerting/feedback on fail condition
Production monitoring / alerting provides immediate feedback; but now failures are visible to customers...

How to do this while reducing risk?
How to do this while 'always on'?

Roll Deployment: Monotonicity and Errors



Impact on the 'software company'

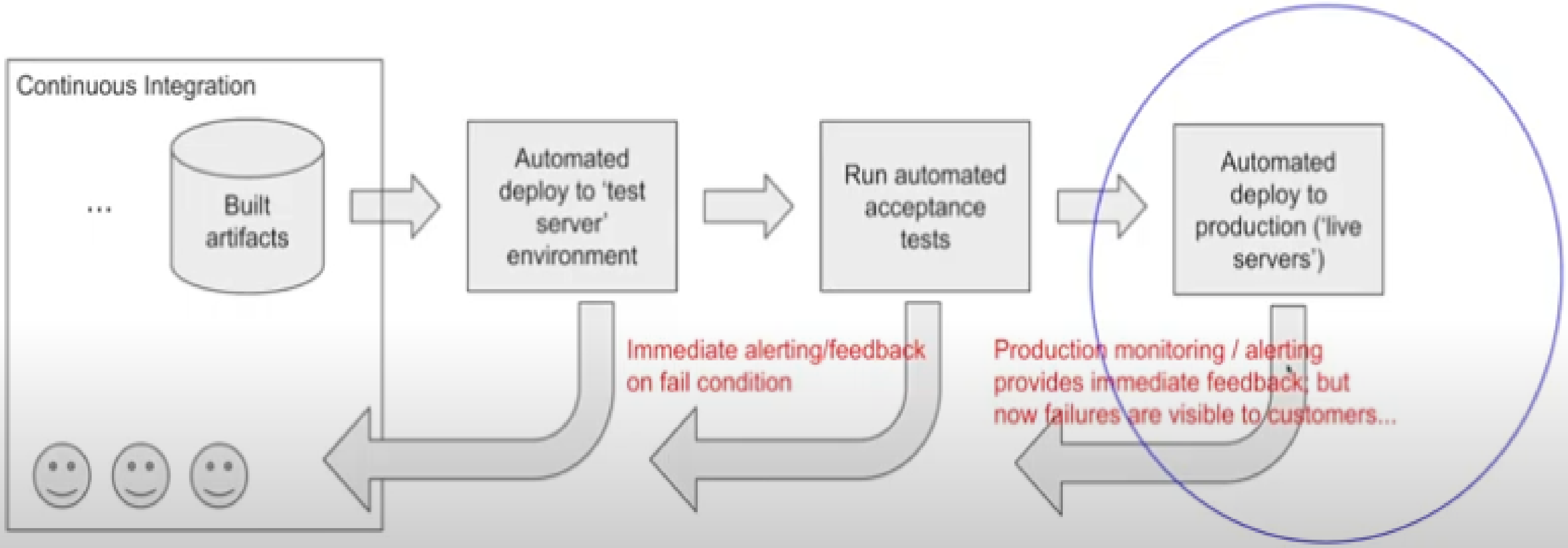
- Now have to worry about building software *and running it*
- Have to continue evolving/upgrading the software with *zero downtime*

But the good news:

- 'Software release' no longer an all-or-nothing discrete event
  - Provides new ways to manage quality and reduce risk
- Continuous visibility into user behavior
  - Provides user/commercial insights back into iterative software development process
- State and runtime environment fully controlled by service provider
  - Improves quality and makes upgrades a lot less risky (if done right)

Roll Deployment: Monotonicity and Errors

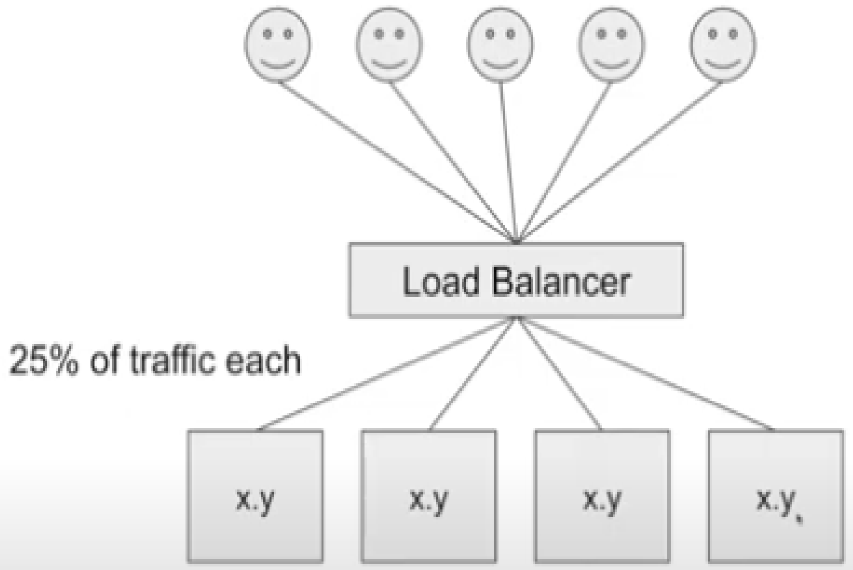
Continuous Deployment (CD):
bring 'deploy' into the 'short cycle'


Continuous Integration
...
Built artifacts
Automated deploy to 'test server' environment
Run automated acceptance tests
Automated deploy to production ('live servers')
Immediate alerting/feedback on fail condition
Production monitoring / alerting provides immediate feedback; but now failures are visible to customers...

How to do this while reducing risk?
How to do this while 'always on'?

Roll Deployment: Monotonicity and Errors

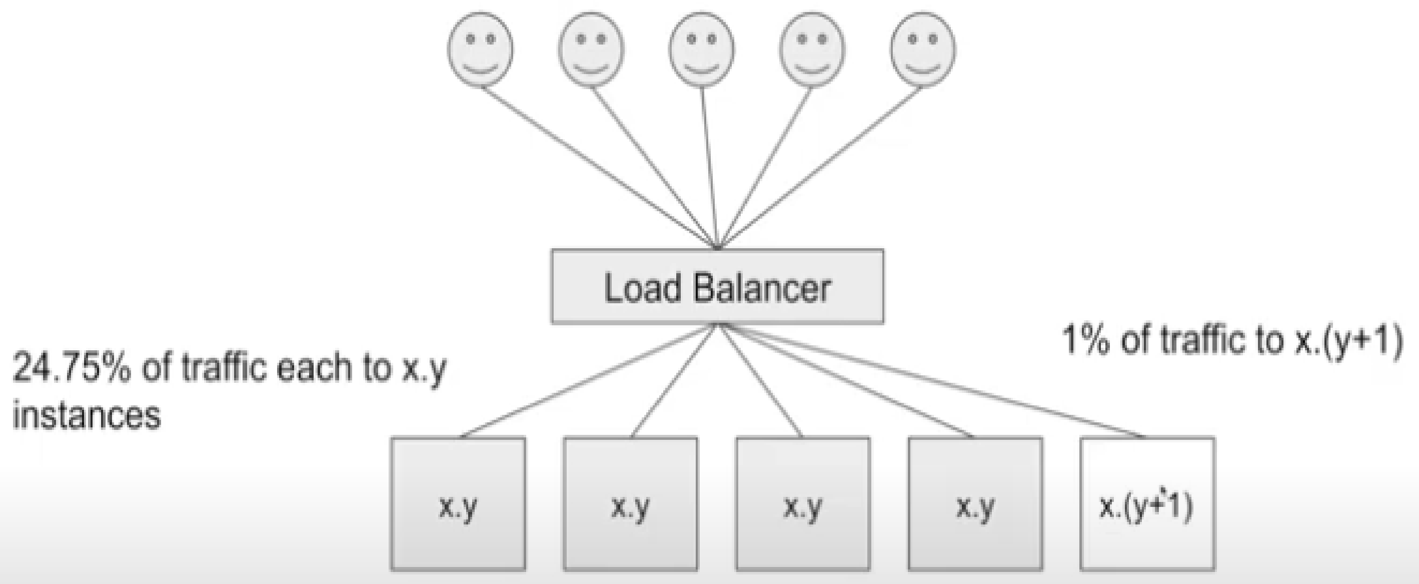
Rolling deploy



Note: these resources are usually running in a cloud platform. So virtual machines, load balancers, storage, network etc. can all be provisioned and configured through the cloud platform's APIs.

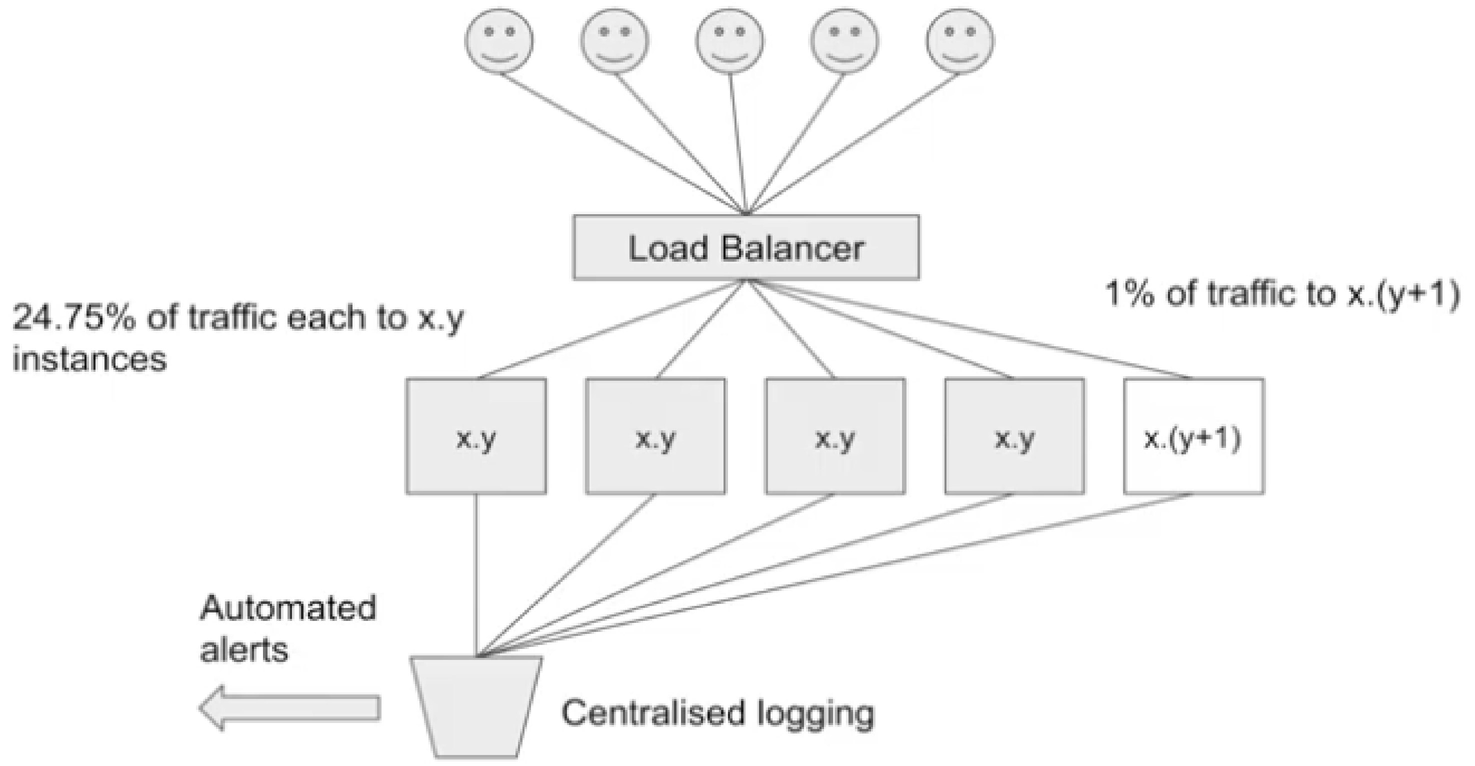
Roll Deployment: Monotonicity and Errors

Rolling deploy: 1) Deploy 'canary' (limit exposure/risk)



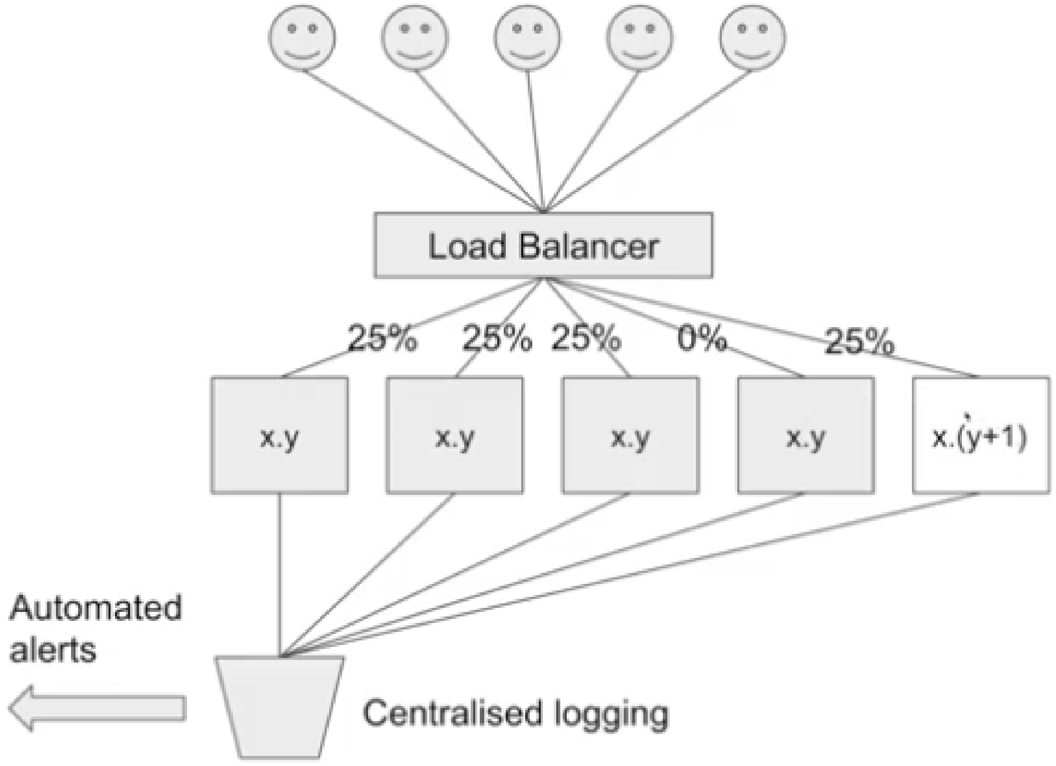
Roll Deployment: Monotonicity and Errors

Rolling deploy: 2) Automated monitoring of error rates - OK?

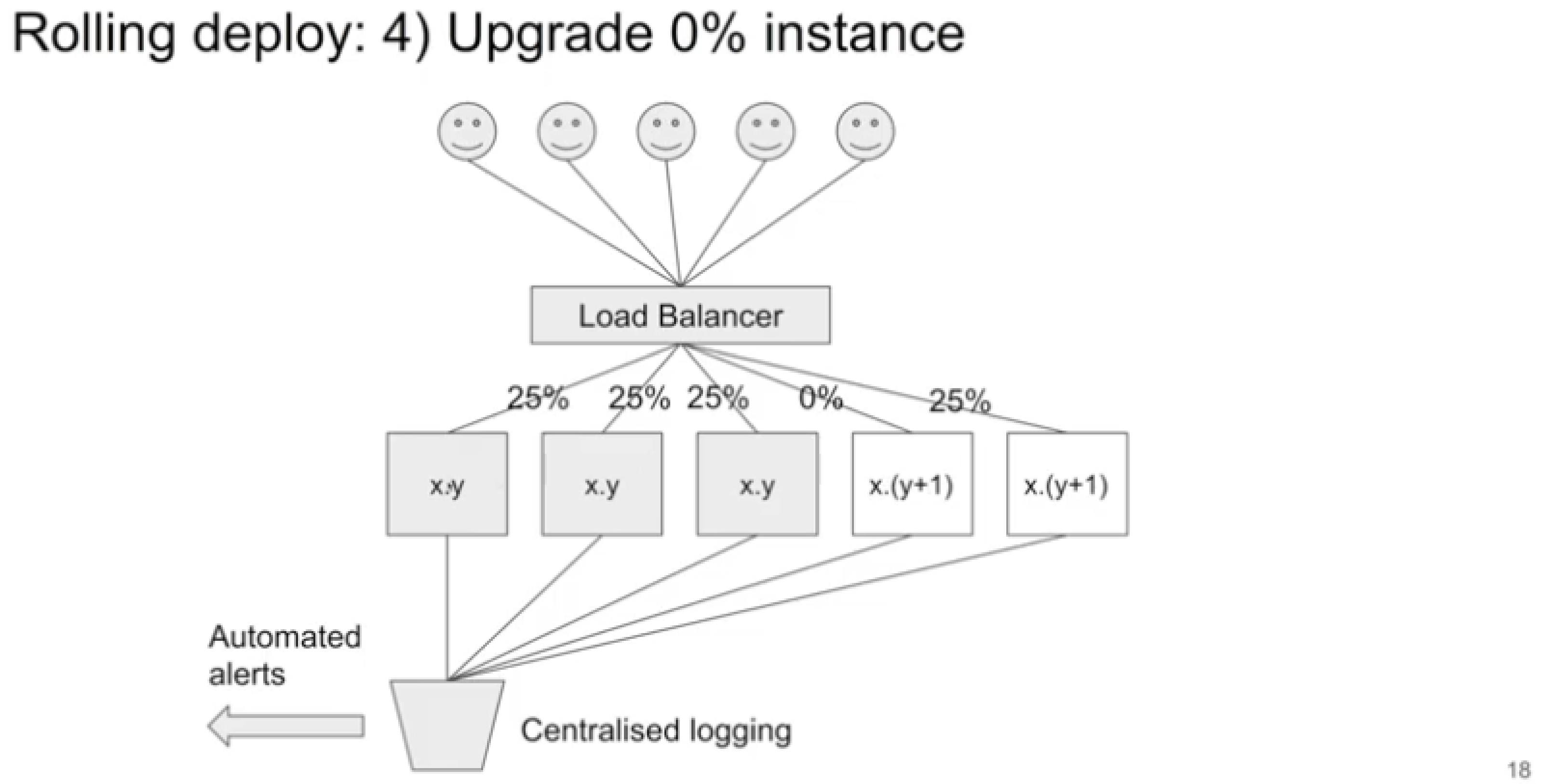


Roll Deployment: Monotonicity and Errors

Rolling deploy: 3) Move traffic from old instance to new



Roll Deployment: Monotonicity and Errors


Rolling deploy: 4) Upgrade 0% instance

Roll Deployment: Monotonicity and Errors

- If errors are detected, all the traffic is directed to the old instances, and the new ones are removed
- In this case the monotonicy of the deployment is lost to be able to guarantee availability
- The system is always available in spite of updates
- Useful to deploy patches to remove vulnerabilities without checking in advance if all the functions are still available

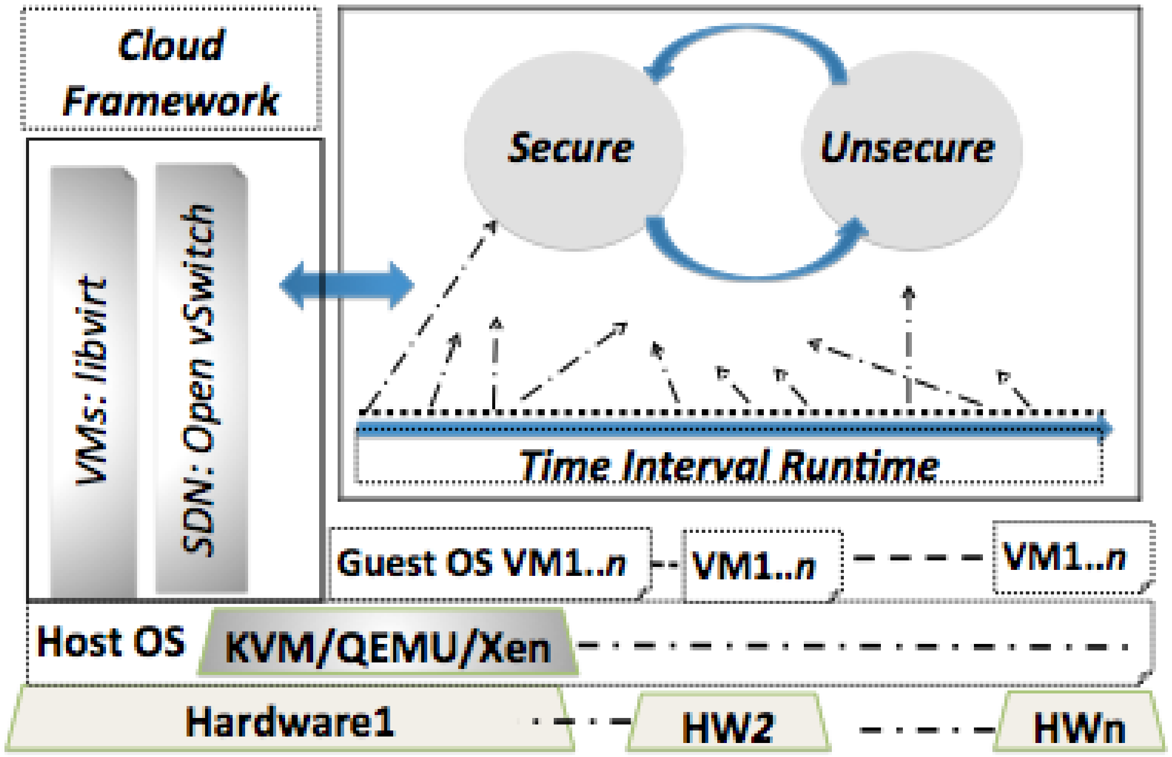
Virtual Reincarnation : A defense strategy using loss of monoticity

- The main idea is to stress diversity by running an application on distinct nodes and distinct OSes at distinct times
- A VM is migrated to a distinct node so that if the underlying VMM or hardware has been attacked, the attacker loses control
- In distinct moments the application is run on distinct VMs so that if the OS is attacked, the attacker loses control
    - This is implemented by trasmitting to the second VM a log of the operations of the first one
    - The second VM is frozen when the first one runs and then the first is frozen and the second one is defrozen
    - Requires an intrusion detection infrastructure because it is activated by intrusion detection

Virtual Reincarnation = Moving Target Defense

- The potential target of an attack is reconfigured in a continuous way to prevent the attacker from collecting the information it needs for its attacks
- **Adversaries have an asymmetric advantage**: They have the time to study a system, identify its vulnerabilities, and choose the time and place of attack to gain the maximum benefit
- **MTD imposes** the same asymmetric disadvantage on attackers by making systems dynamic and therefore harder to explore and predict
- Network level MTD: Changing the topology, including IP-hopping, random port numbers, extra open or closed ports, fake listening hosts, and obfuscated port traffic, fake information about the host and OS type and version.
- Host level MTD: Changing the host and OS level resources, naming and configuration.
- Application level MTD: Changing the application environment: randomly arranging memory layout, changing application type and versioning and routing them through different hosts, or changing setting and programming languages to compile the source code, altering the source code at every compilation.

Overall Architecture

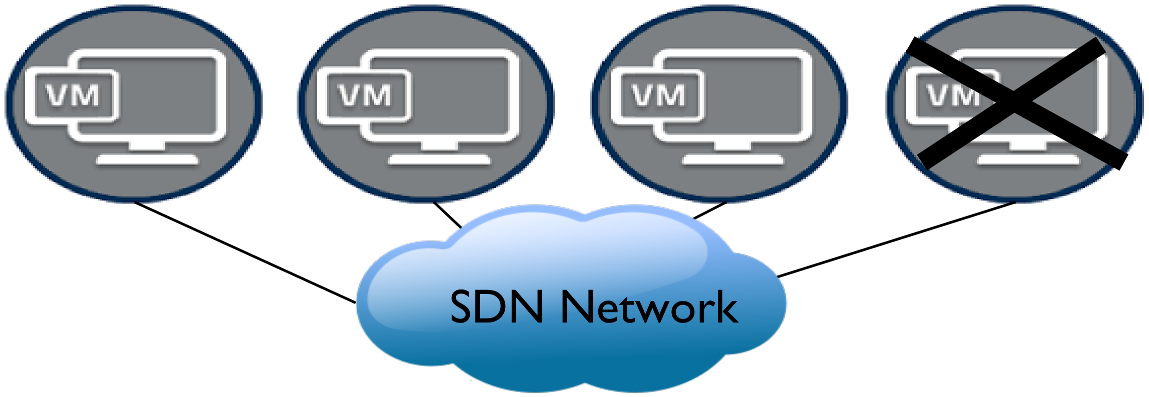


Components:

(1) Virtual Reincarnation (ViRA)
(2) Proactive Monitoring
(3) SDN Network Dynamics
(4) Systems States and Application Runtime

Overall Architecture: Details

- Nodes run a distributed application on a given platform for a controlled period of time
- The running time is chosen in a way that successful ongoing attacks become ineffective
- The new fresh machine will integrate to the system and continue running the application after its data is updated



Virtual Reincarnation

- Randomization and diversification technique where nodes (virtual machines) running a distributed application vanish and reappear on a different virtual state with different guest OS, Host OS, hypervisor, and hardware .

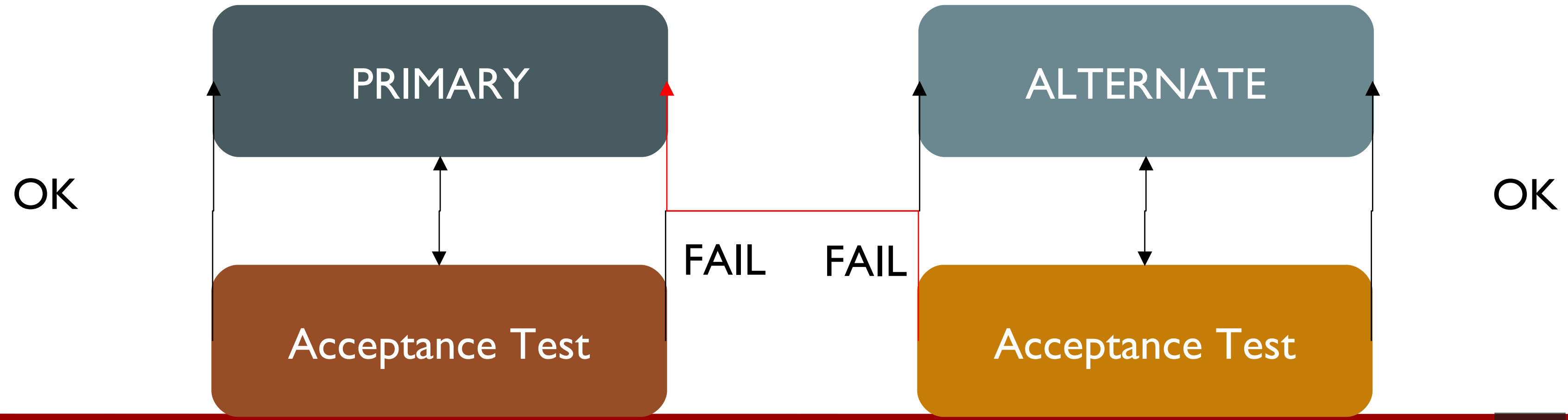
Improve Resiliency

Improve Anti-Fragility

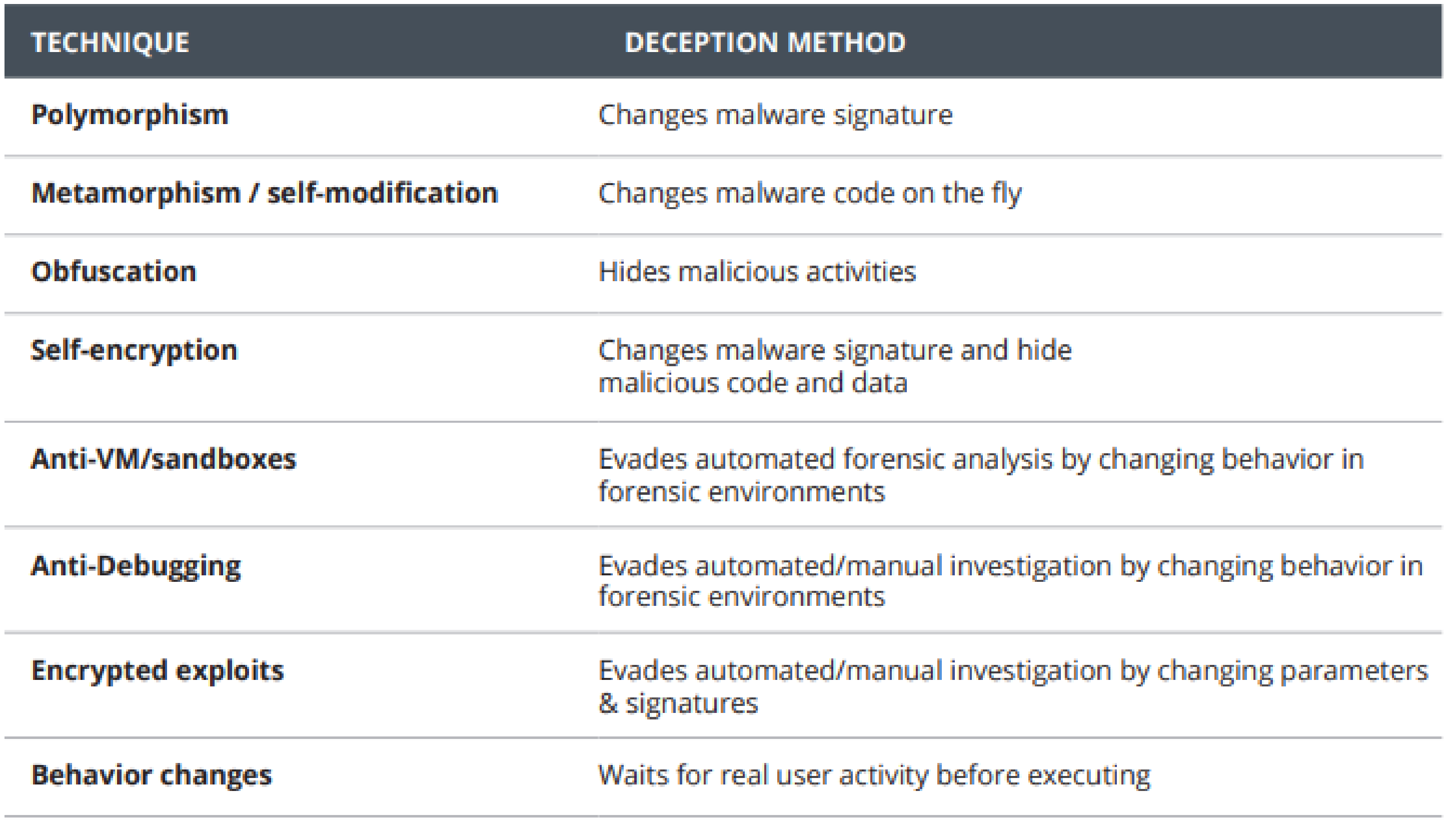
Virtualized Environment

How do we create replicas?

- Primary VM runs as no failures are detected.
- Alternate VM takes place when a failure occurs
- Acceptance tests are adjusted independently to guarantee system operation
- Alternate learn from Primary and become more robust to failures/attacks experimented by primary



Attacks are non monotone

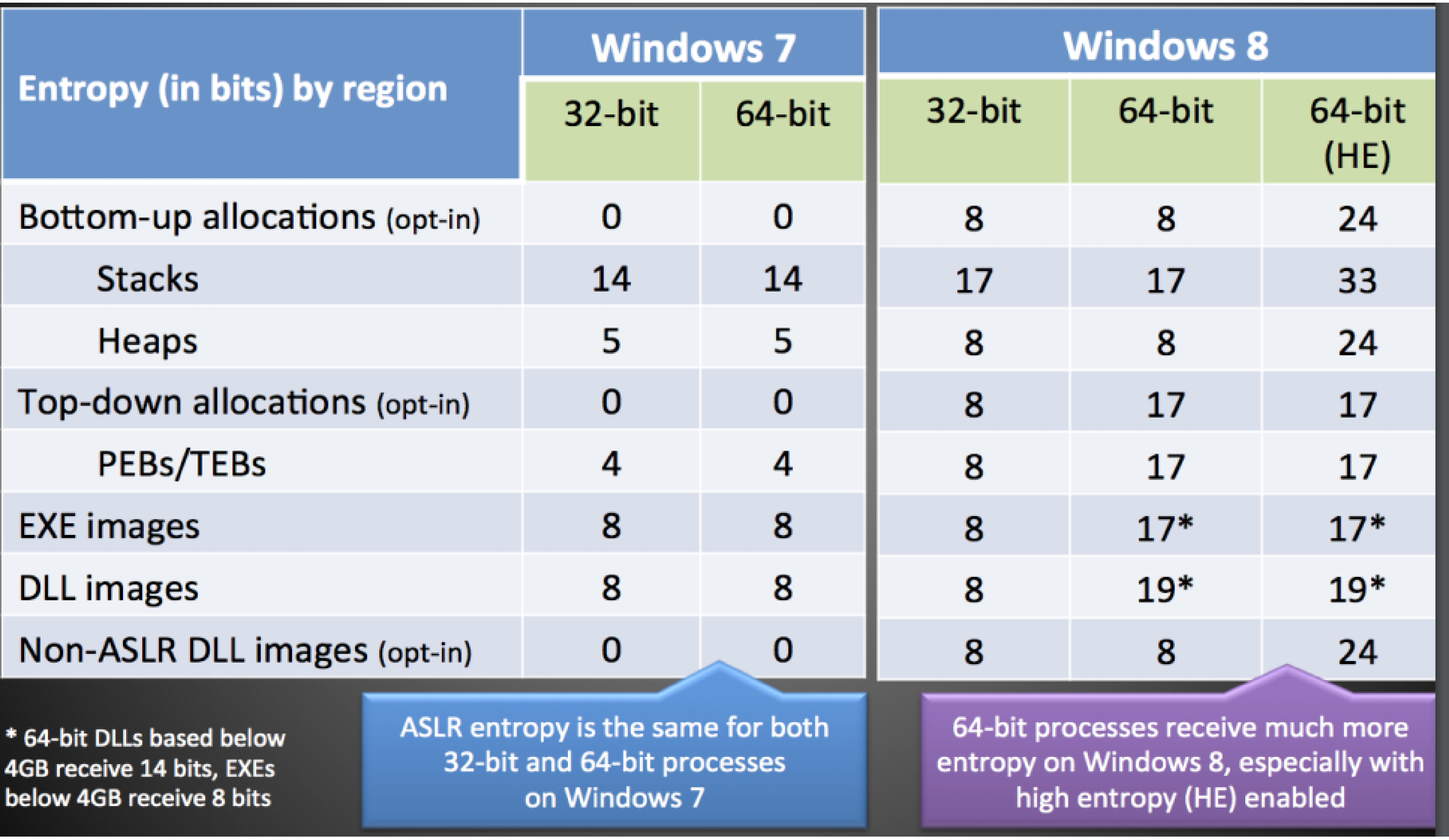


TECHNIQUE	DECEPTION METHOD
Polymorphism	Changes malware signature
Metamorphism / self-modification	Changes malware code on the fly
Obfuscation	Hides malicious activities
Self-encryption	Changes malware signature and hide malicious code and data
Anti-VM/sandboxes	Evades automated forensic analysis by changing behavior in forensic environments
Anti-Debugging	Evades automated/manual investigation by changing behavior in forensic environments
Encrypted exploits	Evades automated/manual investigation by changing parameters & signatures
Behavior changes	Waits for real user activity before executing

Address Space Layout Randomization

- It assigns random addresses to program components in the memory space. OSs use ASLR to reduce the risk due to vulnerabilities that grant access to memory locations, such as buffer overflow.
- Whenever a new process is loaded in the main memory, the OS loads different areas of the process (code, data, heap, stack, etc.) at random positions in the process virtual memory. Attacks that rely on precisely knowing the absolute address of a library function are very likely to crash the process, thus preventing a successful intrusion.
- It makes it harder for an attacker to determine the position of a specific block of a target program in RAM, reducing the likelihood of a successful attack.
- Security increases with the search space. It is more effective when more entropy is present in the random offsets.

Entropy in Randomization



Entropy (in bits) by region	Windows 7		Windows 8		
	32-bit	64-bit	32-bit	64-bit	64-bit (HE)
Bottom-up allocations (opt-in)	0	0	8	8	24
Stacks	14	14	17	17	33
Heaps	5	5	8	8	24
Top-down allocations (opt-in)	0	0	8	17	17
PEBs/TEBs	4	4	8	17	17
EXE images	8	8	8	17*	17*
DLL images	8	8	8	19*	19*
Non-ASLR DLL images (opt-in)	0	0	8	8	24

* 64-bit DLLs based below 4GB receive 14 bits, EXEs below 4GB receive 8 bits

ASLR entropy is the same for both 32-bit and 64-bit processes on Windows 7

64-bit processes receive much more entropy on Windows 8, especially with high entropy (HE) enabled