Security in Virtualized Systems

Ophir Rachman

Engineering Director - Security Products

VMware, Inc.

4/2008

vmware

PART I – Security Basics

What is security?

se·cu·ri·ty : freedom from care, anxiety, or doubt; well-founded confidence

Information Security

- Confidentiality - limiting information access and disclosure to authorized users
- Integrity - trustworthiness of information
- Availability - availability of information resources

Overview

Motives

- Follow the money trail..

Vulnerabilities & Exploits

- Human & software vulnerabilities. High picks

Popular Malware

- Spyware, rootkits, and Bots

Motive

Network & Computational Resources

- Provide anonymous tunnel, so attacker can:
  - Launch attack on high profile targets
  - Install spam \ phishing server
  - Run terrorist network
- Launch distributed DoS
- Maintain backdoor for future data access

Data

- Corporate – Intellectual property, Business plans, Insider information, employee data
- Government – Discover weaknesses, intelligence ..
- Personal – Identity, personal communications, ..

Vulnerability Picks

Social Engineering

- The Human factor. Security = Trust in strangers
- Not going to go away. Ever.

SQL Injection

- The simplest way to get the Data you need without even compromising any web application

Unsafe Languages

- Why I started to like C#

Social engineering

Phishing

- Attacker send message with link to a spoofed site as the bait
- Many techniques to make site look and feel like the original
- Many techniques to make the link look innocent
- User is likely under pressure to fix some “problem”
- User will give away credentials

Protection

- Never follow suspicious link!
- Manually go to the main site in question and look for more details from there

Social Engineering

Spear Phishing

- Advanced Phishing. Targeting a smaller audience
- Use internal knowledge of internal lingo or messages format

Protection

- Be cautious with internal mail.
- Validate that internal mail comes from internal sender
- Examine the link you follow

Social Engineering

Pharming

- Bypass the need to social engineer a user to follow a link
- Attack DNS server that are either vulnerable or miss-configured and poison their cache with bogus records
- When user use legitimate link, we will be redirected to spoof site

Protection

- Use only authoritative DNS
- Validate that internal mail comes from internal sender
- Examine the link you follow

Social Engineering

Viruses

- Started as a hobby for geeks, recently became motivated by financial gains
- Original viruses simply propagated to more system. Today, most viruses include some malicious payload that will leave a backdoor in the system
- Viruses rely heavily on social engineering
- Virus writer also pioneered a few evasion techniques
  - Polymorphism – Many binary representation with identical logic
  - Packing – Encryption at rest and decryption at runtime

Protection

- Be aware of what you run, and open. There is no harmless file format left
- Run Anti-Virus, Patch your system

Social Engineering

Trojan Horses

- Unlike viruses they don’t replicate by themselves.
- The user knowingly install them
- Once installed, the Trojan will establish connection with its creator and surrender control of the attacked system

Protection

- Use Anti-Virus / Anti-spyware
- Be aware of shareware, ActiveX, game mods, P2P network content

Software Vulnerabilities - SQL Injection

How It Is Done?

- Attacker find an interface in the application (e.g web application) that is implemented by a Database (e.g. login page)
- Vulnerable application takes user input and use it to create an SQL query
- Attacker carefully craft its input, to inject is own SQL commands

Example

Select * from users
Where user_name =‘$user’ and
password = ‘$password’

Attacker input:
‘ or 1=1 --

Select * from users
Where user_name =‘’ or 1=1 --’ and
password = ‘$password’

Software Vulnerabilities - SQL injection

Blind SQL Injection

- When the application SQL query string is not known to the attacker, the attacker need to work its way to properly craft his injection
- With a sequence of planned injections the attacker will produce application errors that will provide:
  - Evidence of SQL injection vulnerability
  - Table names
  - Fields names

Protection

- Fix the application – easy in comparison to other software bugs
- Install Database firewalls / Intrusion prevention

Software Vulnerabilities - Unsafe Languages

Definition

- unsafe languages have no built-in memory management support which lead to many possible memory corruption scenarios, the most famous of which are buffer overflows

Overview

- Class of unsafe language vulnerabilities
- Example - The lifetime of buffer overflow exploit
- Protection technologies

Unsafe Languages - Popular picks

Buffer Overflow

- User input is copied into a buffer too small to contain the full length of the input. Result in corruption of memory adjacent to buffer
- No algorithm exists that can scan C\C++ and as a result find all possible buffer overflow bugs

Double Free()

- Calling free() on an already free memory can allow an attacker to reinitialize the buffer backed by the recently freed memory and control the heap so it will override arbitrary memory

Un-initialized Memory Access

- Attacker can force an application into a state in which an attacker control the value of an “un-initialized” variable and force the application to use it to its will.

Unsafe Language – Exploit breakdown

The Lifetime Of A Buffer Overflow Exploit

- Attacker send input to target software
- Vulnerable code executes and a variable gets overflowed
- Depending on variable location, either stack, heap, or .data section are now containing attacker controlled data
- Attacker gain CPU control by overriding return-address , exception-handler , or a function pointer
- Assembly shellcode starts executing in the software context
- Shellcode connect to attacker , steal data on the system, or install a backdoor for future use

Attacks – Unsafe Language

Detection Of Buffer Overflow Exploits

Stage	Protection
Malicious input sent	Network intrusion prevention signature (specific or heuristic)
Vulnerable code executes	Run time patching of vulnerability
CPU control hijack	Security cookies instrumentation & policy Address space randomization (ASLR)
Shellcode executes	Non execute memory (NX)
Shellcode execute systemcalls	Host intrusion prevention signature

Popular Malware

Spyware

- Designed to retrieve information from its host system
- Most Spyware will install a “key” logger to capture user credentials as they are being typed in
- Its less dangerous version “Ad-ware” will only collect generalized information about web usage, doing so by installing browser extension
- Often show rootkit capabilities to maintain stealth in the system

Popular Malware

Rootkit

- Designed to hide its presence in the system.
- Typical rootkit will hide files, process, and network connection.
- More advanced rootkit include covert network channels, and shadowed memory
- Though a well designed rootkit runs in the OS kernel, there are multiple common rootkit that operate from user mode
- Rootkits have became increasingly popular recently, possible due to the common practice now days of bundling rootkit component in traditional malware
- System that have hardware support for virtualization which are not utilizing it are in greater risk for virtualization assisted rootkits.

Detection Of Rootkits

- Hooking point enumeration (syscall tables, import tables etc)
- Crosscheck of object enumeration from multiple view points

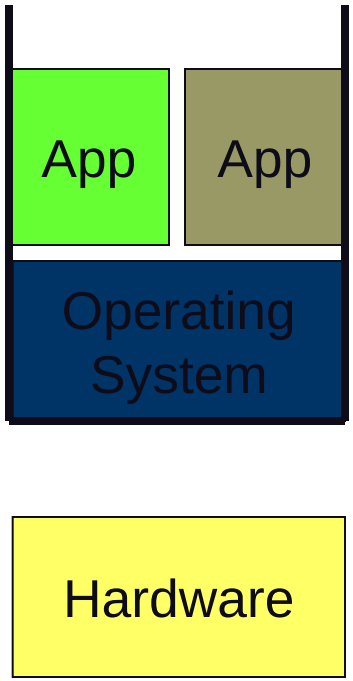
Popular Malware

BotNet

- Designed to convert compromised systems into a bot-net drone
- Bot-net master is in full control over the bot-net activities and size
- Commonly found to contain multiple exploits for controlled propagation, Bots will also propagate using traditional vectors
- Commonly bundled with rootkit components
- Swiss knife of remote control. Example of available commands
  - Upload / Download / Execute files
  - Launch network scan / DDOS / propagation
  - Scan local system for serial keys, license files, and mail addresses
- And above all, it is open source !!

PART II – Virtualization Basics

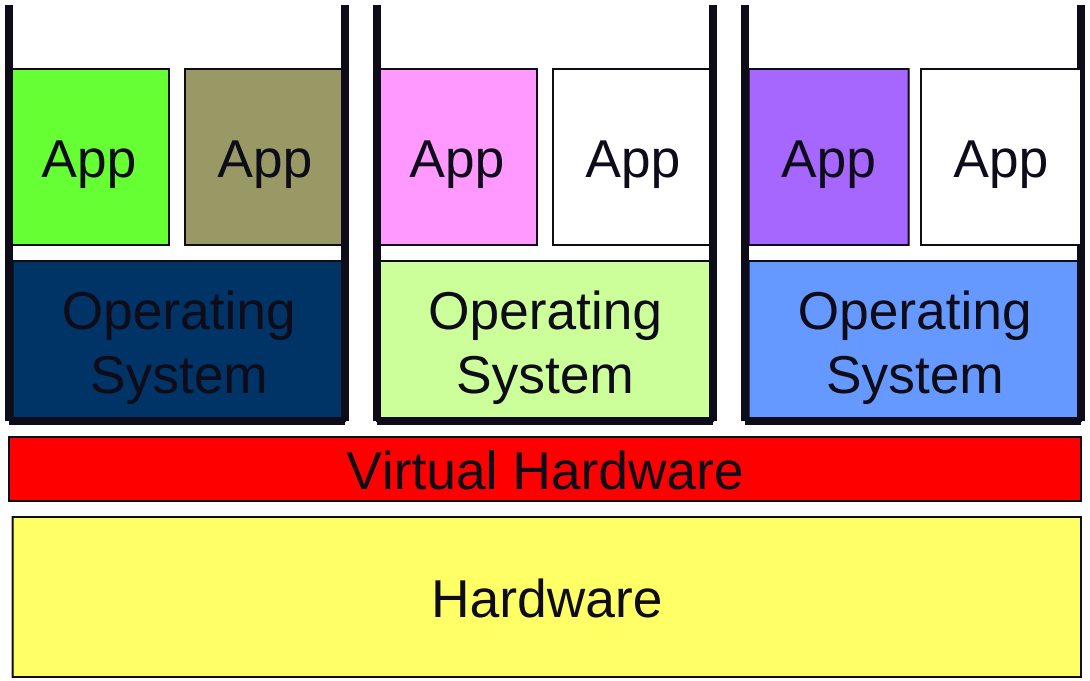
Virtualization – Architecture Basics



Traditional Architecture

Hardware -> OS (HAL)-> OS (Kernel)-> OS (user) -> Applications -> Processes -> Threads

Virtualization – Architecture Basics



Virtualization Architecture (one more abstraction layer)

Hardware -> Virtual Hardware -> Operating systems -> ….

Virtualization – Architecture Options

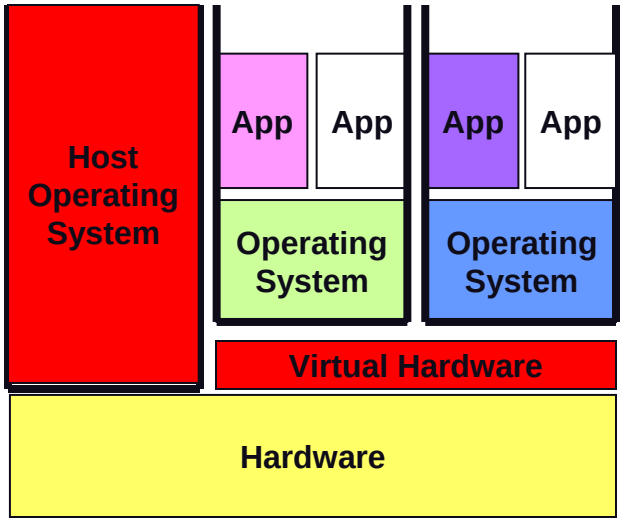
- **What hardware is virtualized?**
  - CPU
  - Memory
  - Storage
  - Network
  - Other devices
- **What is virtualized in software vs. hardware?**
  - All in software
  - CPU/memory in hardware
  - Graphics in hardware
  - Etc.

Virtualization – Architecture Options

- **Operating system supports?**
  - Native
  - Paravirtualized
- **Usage of a traditional host OS?**
  - For everything
  - For boot and launch
  - For drivers
- **And more to come!**

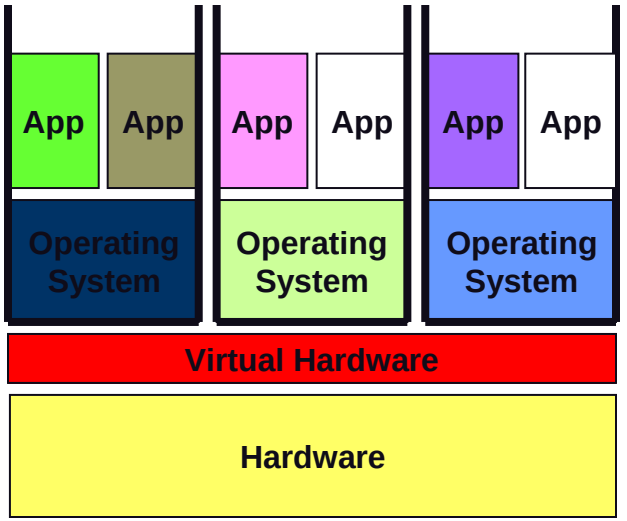
TOOOOOO Many Variants – Maturing High-Potential Architecture/Concept/Technology

Virtualization – VMware Architecture



Hosted Architecture

- **Virtual hardware is an extension of the traditional operating system.**
- **Using the traditional operating system to support a large variety of physical devices.**



VI/ESX

- **Virtual hardware is independent of any traditional operating system**
- **Virtual hardware optimized for:**
  - **Performance**
  - **Data-center storage and networking**
- **Data-center management**
- **Serves as the 'new operating environment' for data-centers**

Virtualization – Capabilities

Basic

- Run different operating systems on the same box
- Virtual networking enables multi-tier environment running on the same box
- Virtual machine encapsulation: copy, suspend, resume, store, etc.
- Hardware independence

Advanced

- Controlled access to machines resources (cpu, networking, storage, etc)
- Mobility – the ability to move running machines from one hardware box to another
- Record and replay – the ability to record the activity of a virtual machines and repeat it later on potentially a different hardware.
- Virtual networking will create networking layer independent of the physical network
- More to come…on top of the basic concept

Virtualization – Use Cases

Initially

- **Development environments (Test/Dev)**
  - Creating encapsulated run time environment for development and testing
  - Ease the maintenance of multiples target machines
  - Repeatability of execution scenarios
  - Inventory of machines in known scenarios
  - Became the de-facto infrastructure for development

- **Home users**
  - Ease of exposure for new environments
  - Ease of penetration of Linux and open source
  - Cool

Virtualization – Use Cases

Later

- **Data centers**
  - Take advantage of unutilized server world for server consolidation (reduces data centers cost)
  - Data center management capabilities:
    - Backup
    - High-availability
    - Survivability
    - Software life cycle
    - Remote desktop innovations
    - More and more
- **Test/Dev and home**
  - Same

Virtualization – Use Cases

Where is it going?

- **Data centers**
  - Become the standard in any data-center
  - Complete separation of hardware from software
  - Fine-tune the role of the traditional operating systems
- **Enterprise desktops**
  - Isolation used to separate roles and functionalities
  - Used as a client-side management agent for desktops
  - Enriches the mobility and independence of functional units
- **Home users**
  - Challenging environment – high paced and rich in devices
  - Probably best use to separate functionalities controlled by a management machine
- **Test/Dev and home**
  - No competition there…

PART II – Security and Virtualization

Virtualization and Security – Challenges

Ease of virtual machines deployments

- Easy as copying a file
- Increases the number of machines dramatically
- Machines are offline for long periods of time
- Complex virtual machines life cycle
- Management overhead and complexity
- No management methodologies: creation, deployment, updates, EOL, etc.
- Propagation surface of malware

Virtual machines attributes

- No strong identity
- No constant location
- No audit for transition
- Mobility in and out the data-center

New components and scenarios ALWAYS present new threats

Remember the Internet??

Virtualization and Security - Opportunities

New infrastructure – new capabilities and opportunities

- Starting from scratch – build for better security
- New ‘privileged’ core
  - Small
  - Attestable
  - Isolated
- Use machine isolation
  - Separate functionalities
  - Create external machine services:
    - Security
    - Patching
    - Backup
- Use new capabilities
  - Introspection (external monitoring of machines)
  - Virtual networking for better granularity
  - Machine mobility for remediation and control

Platform Security

Enhance the security ecosystem solutions to cover virtual infrastructure

Maturing Virtualization Security - Platform

The platform itself introduces new concepts

- Virtual layer itself
- New communication channels
- New protocols and semantics
- New attack interfaces
- Use scenarios

Need to encourage security solutions enhancements

- Virtual infrastructure network and host scanning and remediation
- Virtual infrastructure Patching
- Virtual infrastructure audit and compliance coverage
- Virtual infrastructure access controls
- Virtual infrastructure authentication
- Virtual infrastructure Integration with security management consoles

Maturing Virtualization Security - Platform

What can we do

- Good security practices
- Keep design safe and simple
- Use opportunity for a small core code base
- Use opportunity to re-design the core platform (vs. current operating systems)

Make virtual infrastructure a ‘good security citizen’

Workload Security

Enhance the virtualization platform to enable best-of-breed security solutions (“virtualization aware solutions”)

VMsafe – VMware Security API's Framework

Mission Statement
Enable security ecosystem partners to develop VMware-aware security products/agents.

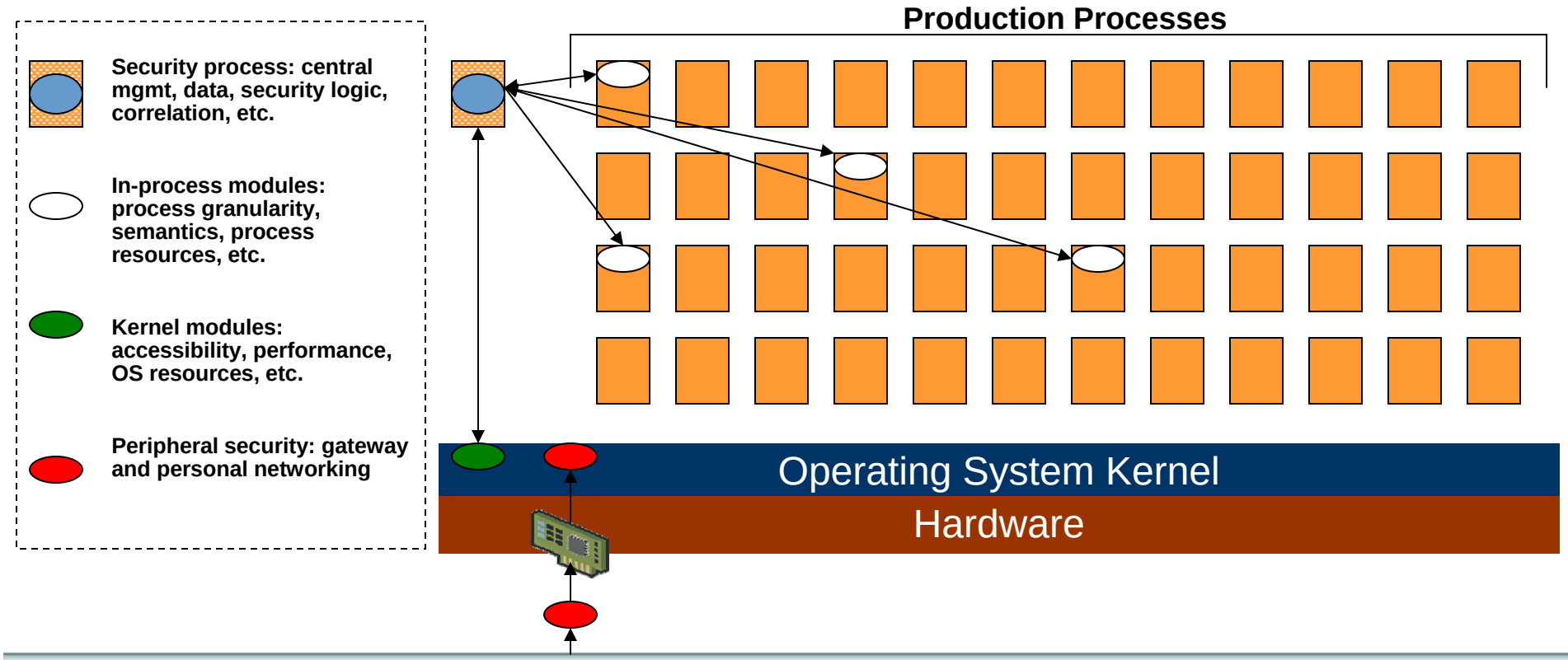
VMsafe high-level objectives

- Design a new, multi-tier security architecture for virtualized data-centers.
- Identify advantages of virtualization for improving data-center security.
- Expose above design and capabilities to enable security vendors to build for better security in VMware environment

VMsafe – VMware Security API's Framework

Mission Statement
Enable security ecosystem partners to develop VMware-aware security products/agents.

Standard Security Architecture



vmware

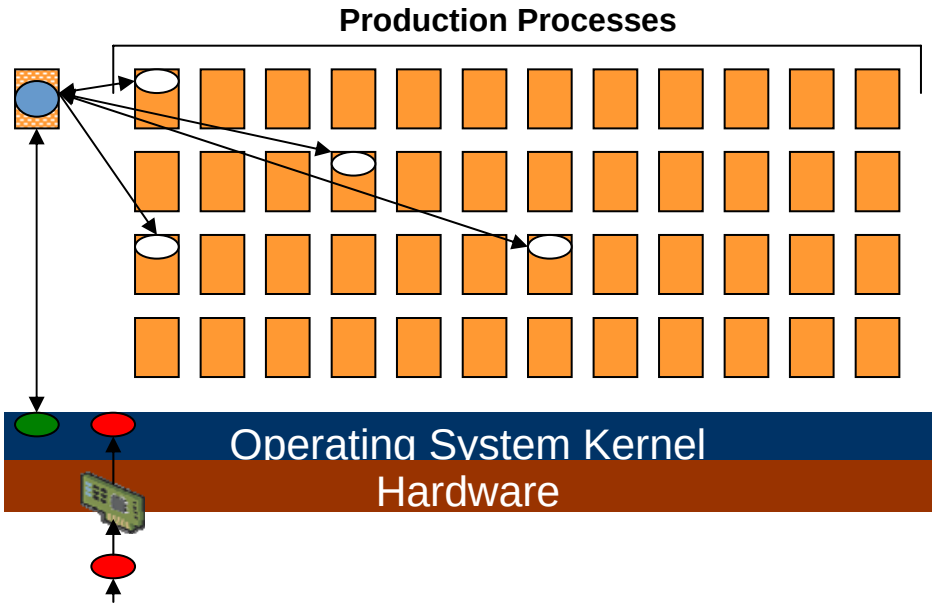
VMsafe – VMware Security API's Framework

Mission Statement
Enable security ecosystem partners to develop VMware-aware security products/agents.

Standard Security Architecture

Concepts:

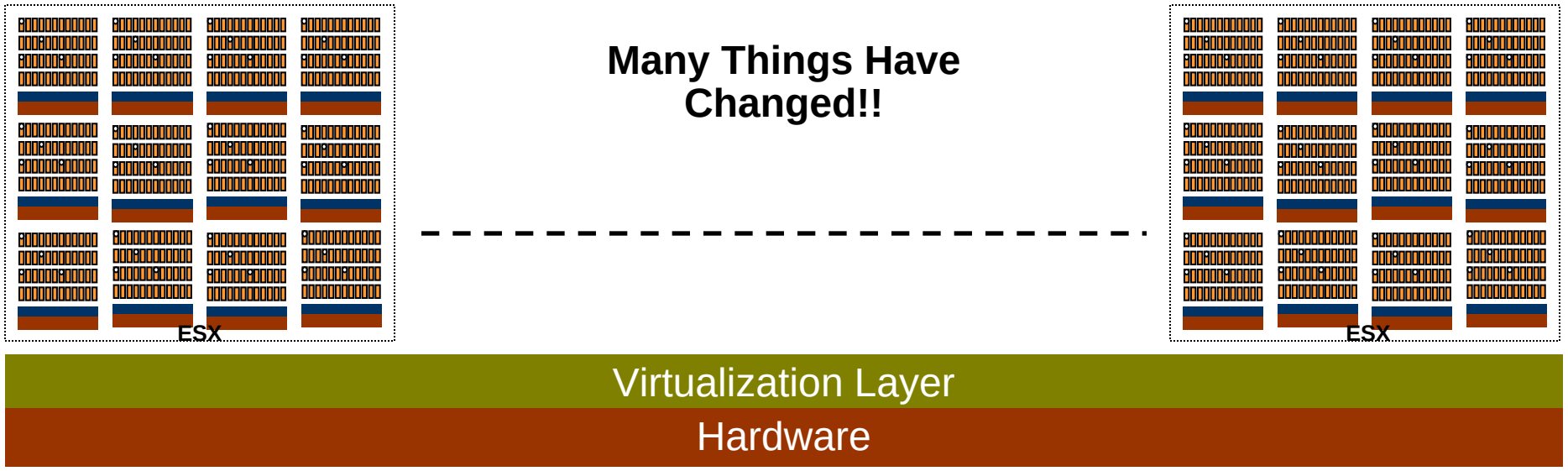
1. **Multi-layered**
2. **Monitor everything**
3. **Monitor everywhere**
4. **Be able to act early**
5. **Get control quickly**
6. **Protect yourself**
7. **Performance-granularity-accuracy tradeoffs**



VMsafe – VMware Security API's Framework

Mission Statement
Enable security ecosystem partners to develop VMware-aware security products/agents.

Virtualization Architecture



VMsafe – VMware Security API's Framework

Mission Statement
Enable security ecosystem partners to develop VMware-aware security products/agents.

Virtualization Security – What Have Changed?

New entities

- Virtual machines
- Virtual networks
- Hosts/Guests
- Virtual devices

New scenarios

- VM's has complex life cycle (clones, snapshots, etc)
- VM's are easy to create and destroy (hard to track)
- Virtualized data center introduces new scenarios (VM mobility, Record&Replay, Virtual appliances, and more)

VMsafe – VMware Security API's Framework

Mission Statement
Enable security ecosystem partners to develop VMware-aware security products/agents.

Virtualization Security – What Have Changed?

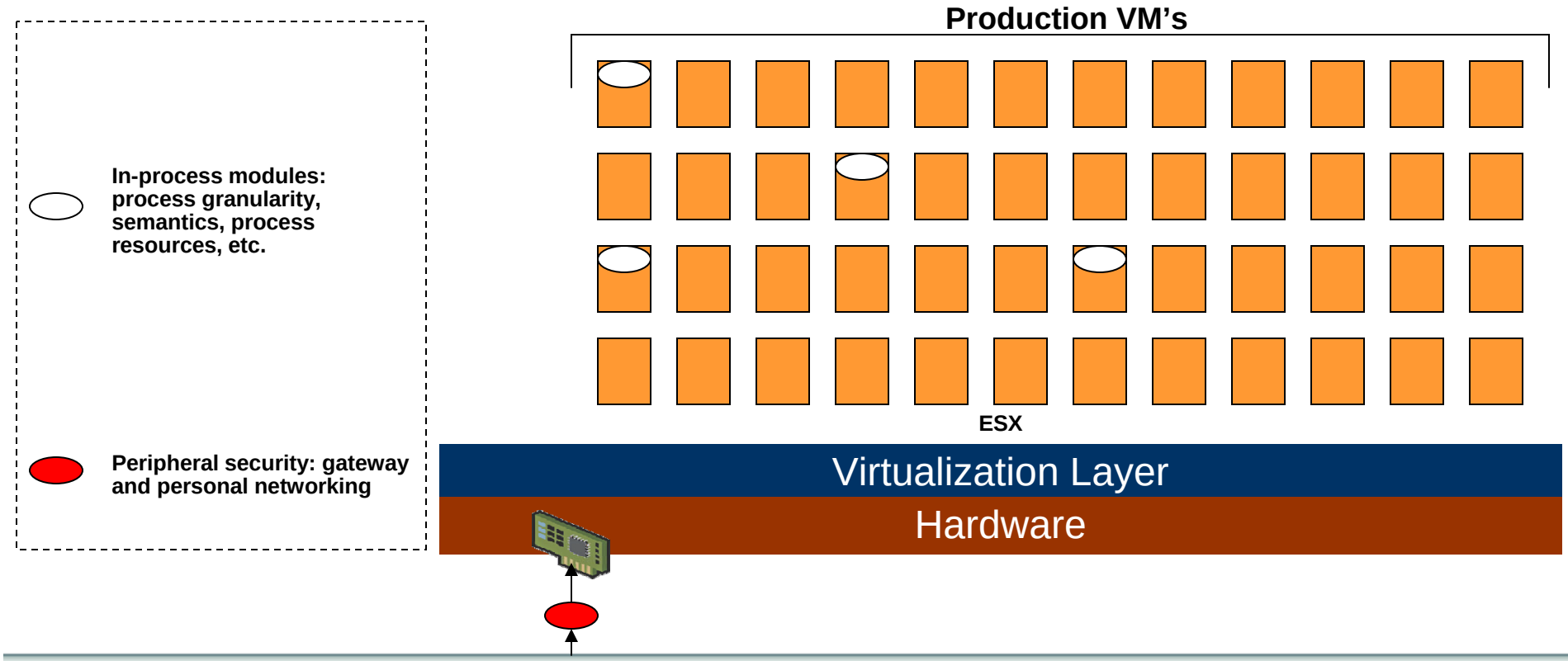
New capabilities

- New abstract layers
- New privilege levels
- Isolation
- Protection
- Introspection

VMsafe – VMware Security API's Framework

Mission Statement
Enable security ecosystem partners to develop VMware-aware security products/agents.

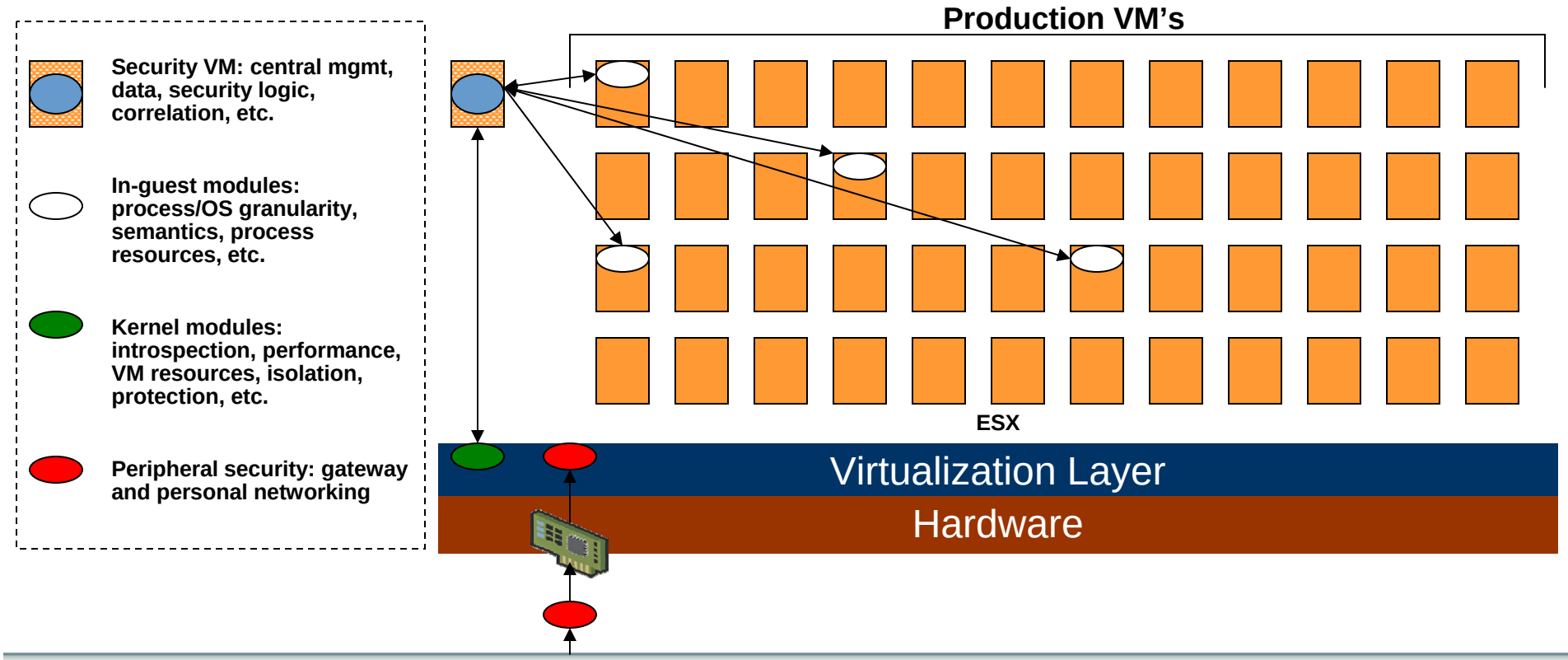
Virtualization Security Architecture



VMsafe – VMware Security API's Framework

Mission Statement
Enable security ecosystem partners to develop VMware-aware security products/agents.

Virtualization Security Architecture

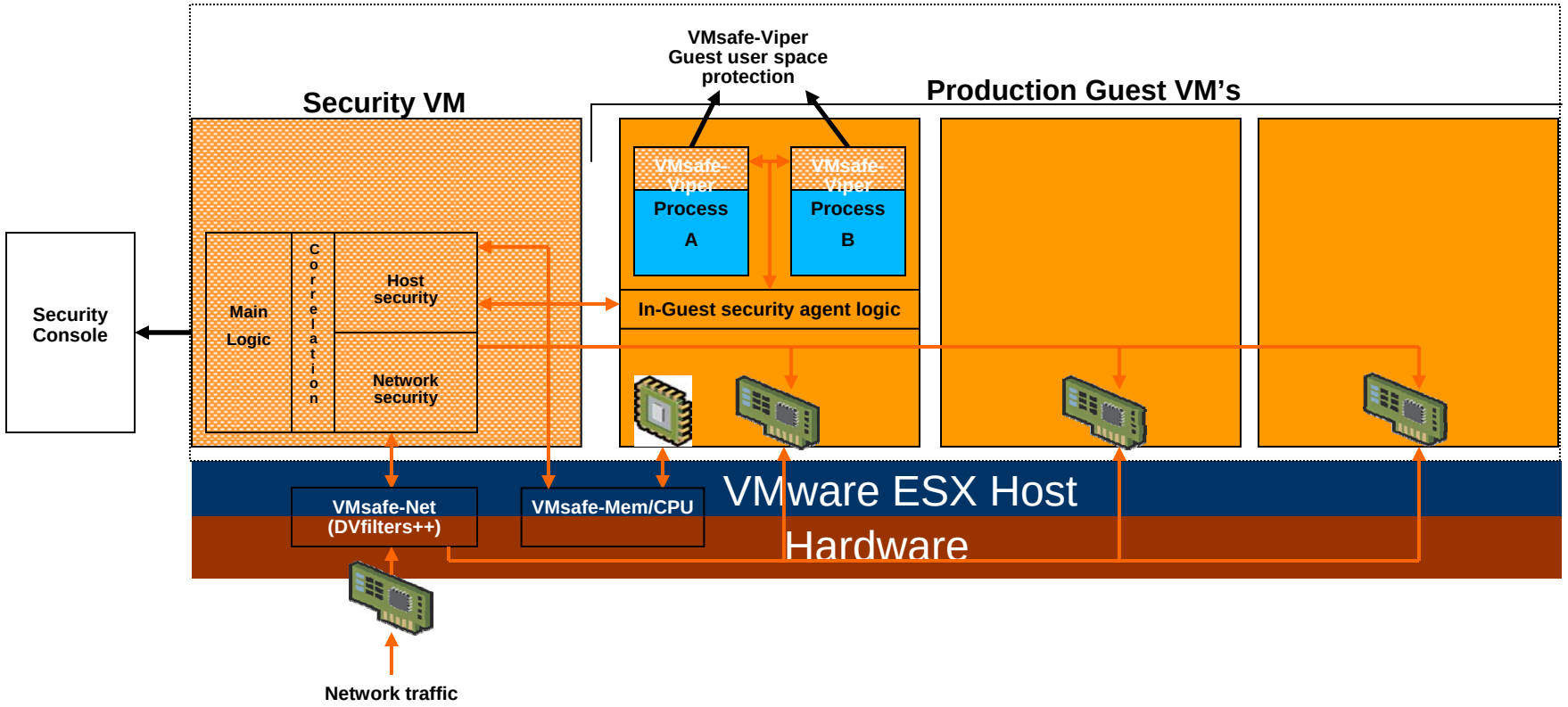


vmware

VMsafe – Basic Architecture

Mission Statement

Enable security ecosystem partners to develop VMware-aware security products/agents – better than physical, better than MS



vmware

VMsafe – VMware Security API's Framework

Mission Statement
Enable security ecosystem partners to develop VMware-aware security products/agents.

VMsafe Basic Components

VMsafe-Net

- In-hypervisor packet filtering modules (monitoring, in-line, NIPS)
- Efficient ability to re-direct specific traffic (or aggregation) to higher-level security VM.

VMsafe-Introspect (CPU-MEM)

- Introspection capabilities of VM state (memory and CPU) from the external security VM.
- Events based monitoring:
  - Ability to define transitions of memory and CPU and define callback handlers.
  - External handling; isolated and protected handling of security events in the security VM
  - Internal handling: fast handling of events inside the guest

VMsafe – VMware Security API's Framework

Mission Statement
Enable security ecosystem partners to develop VMware-aware security products/agents.

VMsafe Basic Components

VMsafe-Viper (Ex Determina Tech)

- In-guest in-process binary translation module for guest processes.
- State-of-the-art application protection infrastructure.
- Canned policies (Determina Memory Firewall HIPS)

Planned

- Disk introspection (I/O blocks level and file system level)
- Keystrokes introspection
- Introspection of graphic stream and other data path devices
- Cloaking technologies to reduce guest OS trust level
- More…

VMsafe – VMware Security API's Framework

Mission Statement
Enable security ecosystem partners to develop VMware-aware security products/agents.

VMsafe Basic Components

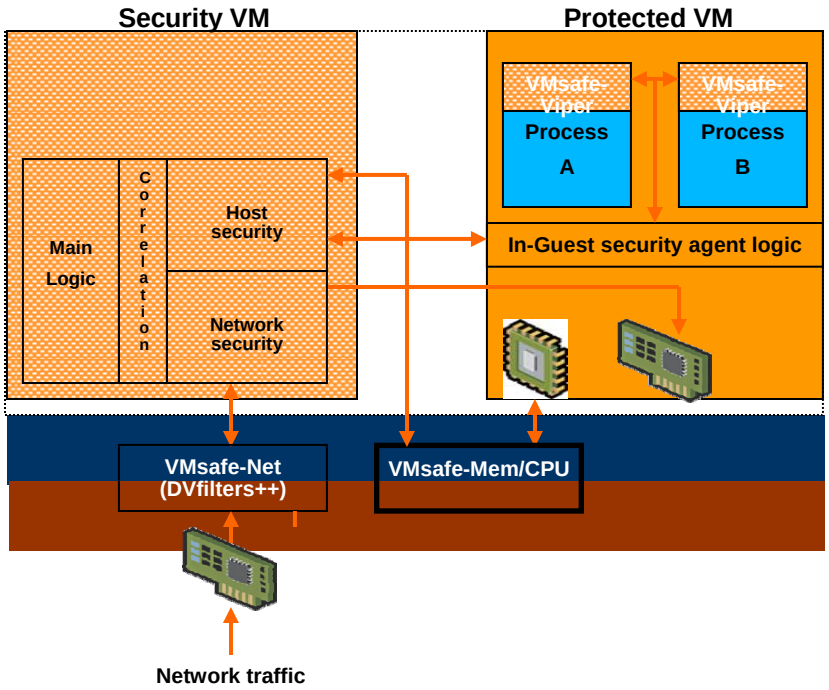
General

- Out-of-the box support for VMware current and future scenarios:
  - Vmotion
  - DRS
  - Record&Replay
  - Notifications of major VM events
  - In-guest in-process binary translation module for guest processes.

VMsafe – VMware Security API's Framework

Mission Statement
Enable security ecosystem partners to develop VMware-aware security products/agents.

Sample Use Cases – Verify Before Execute



Flow

- From the security VM install VBE trigger in for the protected VM
- Automatically get calls into the security VM whenever a 'new' page is being executed.

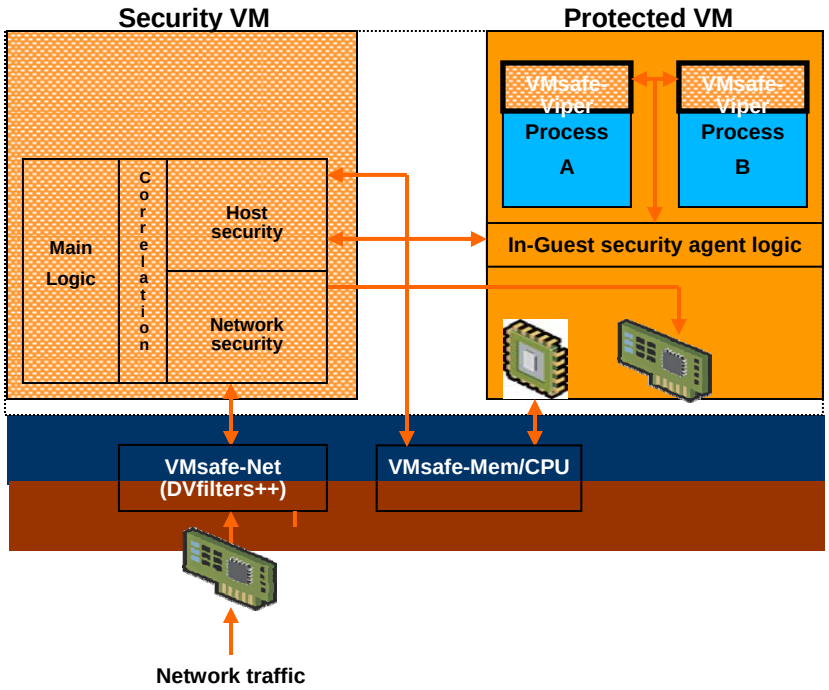
Benefits

- External control/inspection
- Memory level inspection (no disk)
- Packing-based kernel rootkits gone

VMsafe – VMware Security API's Framework

Mission Statement
Enable security ecosystem partners to develop VMware-aware security products/agents.

Sample Use Cases – Hot Patching



Flow

- Install breakpoints on buggy kernel components entry points.
- Install VMsafe-Viper into buggy processes
- Replace buggy code with hot-patches on the fly.

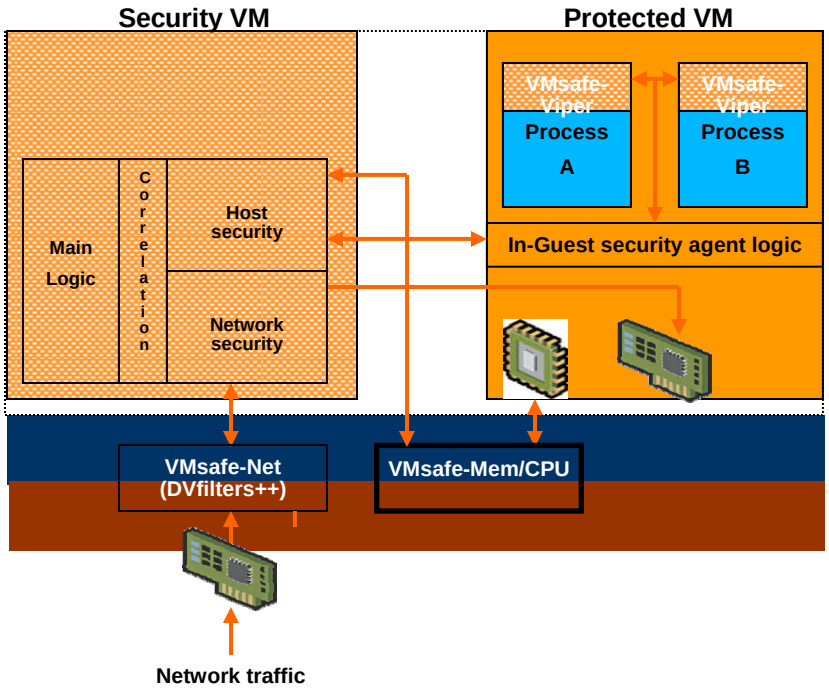
Benefits

- Kernel and user space hot patching with no reboots.
- OS agnostic

VMsafe – VMware Security API's Framework

Mission Statement
Enable security ecosystem partners to develop VMware-aware security products/agents.

Sample Use Cases – Bypass OS limits



Flow

- Install transparent breakpoints at system calls entry points.
- Monitor system calls invocation.

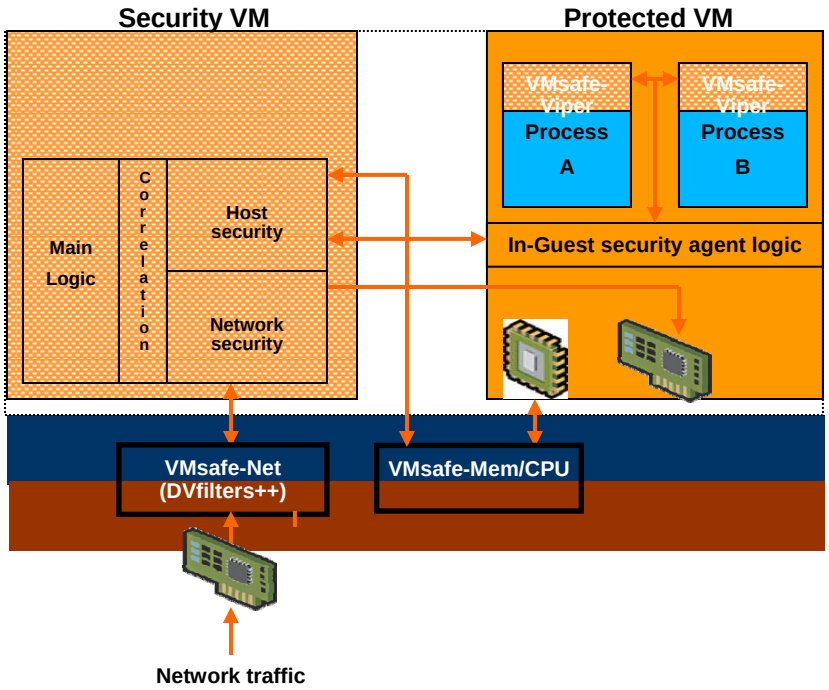
Benefits

- OS-agnostic system call interception engines.
- Bypass PatchGuard and similar in-OS road blocks.

VMsafe – VMware Security API's Framework

Mission Statement
Enable security ecosystem partners to develop VMware-aware security products/agents.

Sample Use Cases – Correlate Network/Host



Flow

- Install in-guest hooks to obtain dynamic guest networking information (applications, sessions, ports, protocols, etc)
- Transfer the information in real-time to the Security VM and to the networking security engine.

Benefits

- Correlated real-time guest and network protection.
- Efficient monitoring of guest application on the network based on real guest data.

VMsafe – VMware Security API's Framework

Mission Statement
Enable security ecosystem partners to develop VMware-aware security products/agents.

Sample Use Cases

More

- Instruction level profiling of guest applications
- First, security – launch security engines before the OS and even before BIOS loads.
- Protection of important OS kernel structures
- Scanning of offline machines (disk/fs api's needed)
- In general, building a layered approach for protection:
  - Hypervisor
  - Security VM
  - OS kernel
  - OS user space
  - OS processes

Questions?