Comprehensive Virtual Appliance Detection at Web Script, Application, and Kernel Levels

Kang Li  
kangli@uga.edu

Xiaoning Li  
lipatchguard@gmail.com
About us

- Kang
  - College Educator

- Xiaoning
  - Security Researcher
The Virtualization World

- Physical Machine
- Hypervisor / VM Platform
- Guest OS
- Applications
The Virtualization World

- Popular Virtual Machine Platforms

- VMware
- VirtualBox
- KVM
- Parallels
- Xen
Virtual Machines versus Appliances

- **Virtual Machines**
  - Generic guests
  - Often run multiple Apps in one guest

- **Virtual Appliances**
  - Specially built guests
  - Optimized to specific applications
    - Pre-configured OS and App stack
    - Isolations
Why Detecting Virtual Appliances?
In the past

Motivation to detect **VMs** ...
Why Detecting **Virtual Appliances**?

- Now, the Emerging Needs are:
  - Avoid Malicious VM Emulators
    - Is my program running on hardware or a VM rootkit?
  - Evade Detection
    - Is the program being evaluated in a specific VM environment?
How to Detect Virtual Appliances?
Detection of Virtual Appliances

- For Virtual Appliances that allow to run arbitrary programs
  - Conventional VM detection technique applies
  - A rich set of prior works

http://strategictrends.ca/
http://comictan.com/
Detection of Virtual Appliances

- But what if the appliance is for a specific application?
  - For example, Browser, PDF, Office …
  - The detection has to be done through application scripts.
  - Most of the prior works on VM detection do not apply.

Solution:

Appliance Detection through Display Properties
Why Focusing on Display Properties

- Display needs to be exported to the “sandbox-ed” applications.

- But, lack of hardware support for graphic virtualization
  - Many appliance instances running in one physical box
  - Difficult for PCI Passthrough to multiple VMs
Virtual Appliance Detection Overview

- **Goal:**
  - Detecting the use of VM by Display Properties

- **Sample Approaches:**
  - **Screen Properties**
    - Resolution, color depth, and refresh rate
  - Support of “Advanced” Display Functions
    - HTML5/WebGL
  - Performance Measurement
    - 3D rendering capability
Scenario #1: Detection Using Resolution

- Using Java to obtain list of display properties
  - Screen Size, Refresh Rate, Color depth

Example: Windows 8 on a Lenovo PC, VMware, and VirtualBox

<table>
<thead>
<tr>
<th></th>
<th>Host</th>
<th>VMware Player</th>
<th>VirtualBox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Resolution Supported</td>
<td>38</td>
<td>25</td>
<td>4</td>
</tr>
<tr>
<td>Refresh Rate</td>
<td>56, 59, 60, 70, 72,75</td>
<td>64</td>
<td>60</td>
</tr>
<tr>
<td>Unique Screen Size</td>
<td></td>
<td>1041x1041</td>
<td></td>
</tr>
</tbody>
</table>
Scenario #1: Detection Using Resolution

- How to Get Screen Properties

Example: the Monitors methods in Adobe Javascript API

- **Monitors Method**
  - Primary
  - Secondary
  - Tallest
  - Widest
  - ...

- **Monitor properties**
  - colorDepth
  - isPrimary
  - **Rect** (boundary of virtual display)
  - workRect

Scenario #1: Detection Using Resolution

- How to Get Screen Properties

Example: the Monitors methods in Adobe Javascript API

- **Monitors Method**
  - Primary
  - Secondary
  - Tallest
  - Widest
  - ...

- **Monitor properties**
  - colorDepth
  - isPrimary
  - Rect (boundary of virtual display)
  - workRect

More properties are available via the Monitors Method.

Scenario #2: Detection Using the Support of HTML5 Features

- Virtual Appliance might Not bother to implement “advanced” graphic support for web scripts.

- Example: Firefox 27.0.1 running on a Windows Host, VMware, and VirtualBox
Scenario #2: Detection Using the Support of HTML5 Features

Same version of Firefox Browser in Host and VM

Firefox 27.0.1 on Win 7
Physical Host

Firefox 27.0.1 on Win 7
Virtualbox (with add-on)
Scenario #2: Detection Using the Support of HTML5 Features

- Virtual Appliance might Not bother to implement “advanced” graphic support for web scripts.

- No Support WebGL ➔ Virtual Appliance
  - Possible False Positives …
Scenario #3: Detection based on 3D Performance

Observations:

- Smooth 3D graphic display heavily relies on hardware support.

- Even with 3D support, browser performance in VM falls far behind modern host systems.

- Often, multiple instances of Virtual Appliances are running on the same physical box.
Performance of Running this Script

Quake 3 in WebGL

http://media.tojicode.com/q3bsp/
Observing Frame Rate of Quake 3

Frame Rate for running the same web script
Ability to Render Concurrent 3D Objects

Number of 3D objects rendered in a fixed frame rate (15 FPS)
Scenario #3: Detection by 3D Performance

- **Strategy**
  - Dynamically increase the number of 3D objects and detect the frame rate
    - 500 ~ 5000 Objects
    - Expect to have FPS > 30 FPS
  - Disguise test under popular games in WebGL
    - E.g. Quake3 in WebGL
Scenario #3: Detection by 3D Performance

- False Positives
  - Slow Physical Machines
    - Atom Box < 500 objects, 15 FPS>
    - Qualcomm Quad-core Table < 500 objects, 30 FPS>

- Implication to Virtual Appliance Detection
How to Get More Accurate VM Information?
If VA Chooses to Enable Native Code

- Example: ActiveX
  
  - Close to run a user-level application
  
  - Although not a good idea for security-aware users to enable,
    
      - likely enabled in those appliances that analyze malware
      
      - Likely lowered the Trustzone levels to enable automatic analysis
Peek Host Info from ActiveX

- Using Predefined APIs
  - WMI (windows management instrumentation)
  - WBEM (web-based enterprise management)
    - Allow to grab systems, networks, devices information.

- Using Known Vulnerabilities in IE/ActiveX
  - Beyond the scope of this talk
Getting Host Device Info from ActiveX

- Using the SWbemLocator
  - Scripting API from WMI

```javascript
... var locator = new ActiveXObject("WbemScripting.SWbemLocator"); var service = locator.ConnectServer("."); var properties = service.ExecQuery(
"SELECT * FROM Win32_DeviceMemoryAddress"); var e = new Enumerator(properties); var p = e.item(); ...

// Available Properties: Device Description, Name, Memory Address ...
```
### Sample Output from Using SWbemLocator

#### Processor

<table>
<thead>
<tr>
<th>DeviceID</th>
<th>Description</th>
<th>Name</th>
<th>SystemName</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU0</td>
<td>x86 Family 6 Model 42 Stepping 7</td>
<td>Intel Pentium III Xeon processor</td>
<td>VIRTUALBOX-XP</td>
</tr>
</tbody>
</table>

#### BaseBoard

<table>
<thead>
<tr>
<th>Name</th>
<th>Manufacturer</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Board</td>
<td>Oracle Corporation</td>
<td>VirtualBox</td>
</tr>
</tbody>
</table>

#### Video

<table>
<thead>
<tr>
<th>DeviceID</th>
<th>Caption</th>
<th>DriverVersion</th>
<th>VideoMode</th>
<th>VideoProcessor</th>
</tr>
</thead>
<tbody>
<tr>
<td>VideoController1</td>
<td>VirtualBox Graphics Adapter</td>
<td>4.2.18.r88780</td>
<td>1024 x 768 x 4294967296 colors</td>
<td>VBOX</td>
</tr>
</tbody>
</table>
## BIOS

<table>
<thead>
<tr>
<th>Caption</th>
<th>Manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default System BIOS</td>
<td>innotek GmbH</td>
</tr>
</tbody>
</table>

## DeviceMemoryAddress

<table>
<thead>
<tr>
<th>Description</th>
<th>CSName</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA0000-0xBFFFF</td>
<td>VIRTUALBOX-XP</td>
</tr>
<tr>
<td>0xE0000000-0xFFDFFFFF</td>
<td>VIRTUALBOX-XP</td>
</tr>
<tr>
<td>0xF0000000-0xF0000FFF</td>
<td>VIRTUALBOX-XP</td>
</tr>
<tr>
<td>0xF0080000-0xF00FFFFF</td>
<td>VIRTUALBOX-XP</td>
</tr>
<tr>
<td>0xF0400000-0xF07FFFFF</td>
<td>VIRTUALBOX-XP</td>
</tr>
<tr>
<td>0xF0800000-0xF0803FFF</td>
<td>VIRTUALBOX-XP</td>
</tr>
<tr>
<td>0xF0804000-0xF0804FFF</td>
<td>VIRTUALBOX-XP</td>
</tr>
<tr>
<td>0xF0805000-0xF0805FFF</td>
<td>VIRTUALBOX-XP</td>
</tr>
</tbody>
</table>

## NetworkAdapter

<table>
<thead>
<tr>
<th>DeviceID</th>
<th>Description</th>
<th>Manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AMD PCNET Family PCI Ethernet Adapter</td>
<td>Advanced Micro Devices (AMD)</td>
</tr>
</tbody>
</table>
**Processor**

<table>
<thead>
<tr>
<th>DeviceID</th>
<th>Description</th>
<th>Name</th>
<th>SystemName</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU0</td>
<td>Intel64 Family 6 Model 23</td>
<td>Intel(R) Core(TM)2 Duo CPU L9400 @ 1.86GHz</td>
<td>X200S-GUODONG</td>
</tr>
</tbody>
</table>

**BaseBoard**

<table>
<thead>
<tr>
<th>Name</th>
<th>Manufacturer</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Board</td>
<td>LENOVO</td>
<td>7465CTO</td>
</tr>
</tbody>
</table>

**Video**

<table>
<thead>
<tr>
<th>DeviceID</th>
<th>Caption</th>
<th>DriverVersion</th>
<th>VideoMode</th>
<th>VideoProcessor</th>
</tr>
</thead>
<tbody>
<tr>
<td>VideoController1</td>
<td>Mobile Intel(R) 4 Series Express Chipset Family (Microsoft Corporation - WDDM 1.1)</td>
<td>8.15.10.2702</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>VideoController2</td>
<td>Mobile Intel(R) 4 Series Express Chipset Family (Microsoft Corporation - WDDM 1.1)</td>
<td>8.15.10.2702</td>
<td>1280 x 800 x 4294967296 colors</td>
<td>Mobile Intel(R) 4 Series Express Chipset Family</td>
</tr>
</tbody>
</table>

**BIOS**

<table>
<thead>
<tr>
<th>Caption</th>
<th>Manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ver 1.00PARTTBLX</td>
<td>LENOVO</td>
</tr>
</tbody>
</table>

**DeviceMemoryAddress**

<table>
<thead>
<tr>
<th>Description</th>
<th>CSName</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xE0000000-0xEFFFFFFFF</td>
<td>X200S-GUODONG</td>
</tr>
<tr>
<td>0xE0000000-0xEF01FFFFF</td>
<td>X200S-GUODONG</td>
</tr>
<tr>
<td>0xE0000000-0xEF013FFF</td>
<td>X200S-GUODONG</td>
</tr>
<tr>
<td>0xE0000000-0xEF018FFF</td>
<td>X200S-GUODONG</td>
</tr>
<tr>
<td>0xE0000000-0xEF019FFF</td>
<td>X200S-GUODONG</td>
</tr>
<tr>
<td>0xE0000000-0xEF04FFF</td>
<td>X200S-GUODONG</td>
</tr>
<tr>
<td>0xE0000000-0xEF145000</td>
<td>X200S-GUODONG</td>
</tr>
</tbody>
</table>
What if Appliance choose to emulate none-VM specific devices?
What if Appliance choose to emulate none-VM specific devices?

No Hypervisor names shown in Device and Drive info.

Real device drivers are used.
Popular Virtual Devices in VM

- **Virtualbox**
  - **NIC:** Intel PRO/1000 MT (82540EM)
  - **Audio:** ICH AC97
  - **IDE:** Intel 82371 PIIX4 IDE
  - **SATA:** Intel 82801HBM/HEM

- **Vmware Fusion**
  - **NIC:** AMD PCnet32 LANCE
  - **Audio:** Ensoniq ES1371
  - **IDE:** Intel 82371 PIIX4 IDE
  - **SCSI:** LSI Logic 53c1030
Difficult to make the behavior of Virtual Device == Physical Device
Virtual-Physical Inconsistency Example

- Intel PRO/1000 MT (82540EM)
  - Popular Virtual NIC

- After the following I/O event
  `mmio_write (ICS [0xC8], 0x4)`

Real NIC
- Register ICS: 0x00000004

Virtual NIC
- Register ICS: 0x80000004
Virtual-Physical Inconsistency Example 2

- Intel PRO/1000 MT (82540EM)
  - Popular Virtual NIC

- After the following I/O event
  `mmio_write (MDIC [0x20], 0x8000)`

Real NIC
- Register ICS: 0x0

Virtual NIC
- Register ICS: 0x200
Virtual-Physical Inconsistencies

Almost all virtual devices contain differences to their physical peers.

How to detect the inconsistencies between physical and virtual devices? [Blackhat Briefing USA 2013]
What about PCI Passthrough?

Not Commonly used in Virtual Appliances.
- Current practice often limited device passthrough to one VM.
  - Things might change in the future with devices like NVIDIA GRID K2

Some Passthrough Implementation Can still be Detected
- E.g. Intel NIC Virtualization
  - Host runs PF functions driver, VM runs VF function driver.
How to resist VM/VA detection?
Prevent Virtual Appliance Detection

- **Difficult to Resist Root and Application Level Detection**
  - Virtual devices have many inconsistencies with physical ones
  - Hardware virtualization support helps but often still leaves clues

- **To Resist Detection at Web/Application Scripts Level**
  - Not too hard to fake simple display properties
  - To resist timing/performance based detection
    - Pretend to be a low-end/old device
  - Challenging for protect against targeted attacks
Summary

- Detecting Virtual Appliances
  - Using Display Properties through Web Scripts
  - Using System Information through ActiveX
  - Using Device Inconsistencies

- The Specialty of Display Devices in VA Detection
  - Have to expose to sandboxed applications
  - Many properties are useful for detection
  - Timing and performance aspects are hard to fake
Thanks!

kangli@uga.edu
ldpatchguard@gmail.com