2008

Preventing the Acquisition of Data from Virtual Machine based Secure PortableExecution Environments

Peter James
Secure Systems Ltd

DOI: 10.4225/75/57b26f7f40cbb
Originally published in the Proceedings of the 6th Australian Digital Forensics Conference, Edith Cowan University, Perth Western Australia, December 3rd 2008.

Preventing the Acquisition of Data from Virtual Machine based Secure Portable Execution Environments

Peter James[1]
Secure Systems Ltd
pjames@securesystems.com.au

Abstract

A Virtual Machine (VM) based secure Portable Execution Environment (PEE) provides a safe and secure environment that can be loaded into a host PC and an application executed with a degree of confidence that the application is separated, protected and little or no forensic evidence remains after the application has executed. A VM based secure PEE is characterised as a USB storage device containing a VM with a trusted guest operating system and application(s) which is stored in a protected partition, strong authentication to only allow an authorised user to load the VM into the host PC, and full storage device encryption to protect the confidentiality of the contents of the device. Secure PEEs provide an opportunity for organisations to issue a portable device to an individual (to perform a secure transaction on an available host PC) with the reduced risk to the organisation that neither malicious software (resident on the host PC) will infect the secure PEE device, nor sensitive data remnants (resulting from the transaction) will remain on the host PC hard disk drive after the secure PEE device has been removed.

A VM based secure PEE significantly reduces the opportunity to use dead forensic analysis techniques to acquire evidence of the occurrence of a transaction. However, VM based secure PEEs are susceptible to the acquisition of data through monitoring software and live forensic techniques. This paper considers the mechanisms that can be used to prevent various monitoring and live forensic techniques acquiring data from a VM based secure PEE. An attack scenario is presented to provide the context for the analysis of VM based secure PEE device vulnerabilities and why it is important that such a device would be required to counter hostile monitoring and forensic analysis. An overview is given of the security mechanisms provided by the type of VM based secure PEE under consideration and how those mechanisms combine to limit the opportunity for data acquisition through dead forensic techniques. The vulnerabilities of VM based secure PEEs with respect to malicious software and live forensic techniques are enumerated and discussed. A comprehensive set of countermeasures are proposed and analysed. The paper concludes by considering the most appropriate countermeasures to include in a VM based secure PEE to prevent the live acquisition of data...

Keywords

Secure Portable Execution Environments, Virtualisation, Virtual Machine Vulnerabilities, Securing Virtual Machines, Digital Forensics.

INTRODUCTION

The convenience provided by public Internet access centres has increasingly resulted in such centres being used to perform sensitive Internet transactions by individuals unaware of the associated risks. Given the totally open access to the PCs in these Internet access centres no level of confidence can be assumed in PC security; it is readily conceivable that such PCs have been compromised by malicious software which can exploit any Internet transactions. In addition PCs often "considered safe" (e.g. the home PC), which are used for a variety of tasks and by a number of different users often lack best practice security (e.g. anti-virus & anti-spyware software and modem/router enabled firewall capabilities). These "considered safe" PCs may contain malicious software unbeknown to the user which may be able to exploit Internet transactions. An option considered by some organisations is to issue employees and/or customers with secure portable execution environments (secure PEEs) to be used to perform sensitive Internet transactions on PCs for which no level of trust can be assumed. Secure PEEs provide trusted functionality to reduce the opportunity of malicious software exploiting sensitive data processing or an Internet transaction.

[1] Peter James is registered on a Professional Doctorate programme at the School of Computer & Information Science at Edith Cowan University. Peter is the Managing Director of Secure Systems Ltd.

In this paper a secure PEE device is considered to be a portable Universal Serial Bus (USB) storage device containing a trusted operating system and application that can be uploaded into a PC and used to perform a transaction with a high degree of confidence that the transaction will not be exploited. A secure PEE device will also provide space to store data, strong authentication to prevent unauthorised access, device encryption to protect the confidentiality of information on the device and partitioning with differentiated access rights to separate and protect the secure PEE.

In *"Secure Portable Execution Environments: A Review of Available Technologies"* (James 2008) a comprehensive review and analysis of secure PEE technologies and products is given. The paper finds that a secure PEE that utilises a bootable OS provides a strong secure PEE that is resistant to malicious software. However, booting a USB device often requires a user to change the PC Basic Input Output System (BIOS) boot order which can be an unfriendly and sometimes a non-trivial activity. An alternative to a bootable OS that requires no interaction with a PC BIOS is for a secure PEE to utilise a virtual machine (VM) with a guest OS, i.e. a VM based secure PEE.

A VM based secure PEE is simpler to load and execute than a bootable OS based secure PEE because the user can load and execute the VM when the secure PEE device is plugged into a host PC running a fully booted OS. A VM provides an abstract execution environment separate from the physical PC. There are a number of different types of VM (Smith 2005). The type of VM considered in this paper is one that runs within (on top of) the PC operating system; often referred to as a type 2 hosted VM. A "guest" OS is hosted within the type 2 VM and the application is executed within the guest OS. However, a VM based secure PEE may be susceptible to any malicious software that maybe resident on the host PC OS and also susceptible to certain data acquisition techniques, e.g. live forensics and monitoring software. In this paper techniques are considered to limit the opportunities of malicious software, monitoring software and live forensic techniques to acquire data from a VM based secure PEE.

The optimal features for a secure PEE device are considered in *"Secure Portable Execution Environments: A Review of Available Technologies"* (James 2008). It will be assumed that the type of secure PEE considered in this paper will have all of the protection measures of the "secure PEE device 4" defined in James 2008; in summary these protection measures are:

- **Authentication:** The device will prevent access to its contents and the VM can not be loaded until an authorised user has entered the correct authentication credentials. The most convenient approach to authenticate a secure PEE device is to plug it into a PC (that has a booted and executing OS) and an authentication application is uploaded from the secure PEE device. Through the authentication application a user authenticates with the secure PEE device.
- **Device Encryption:** The secure PEE device will be fully encrypted using on-the-fly encryption to preserve the confidentiality of the secure PEE and data residing on the USB device. In this paper it will be assumed that on-the fly-encryption will be implemented in hardware.
- **Swap space (virtual memory/page file) and space for temporary files:** The VM guest OS will require swap space. The usual default approach is to create the swap space on the host PC hard disk drive (HDD). Also applications executing on the guest OS may write temporary files to a 'temp' directory/folder on the host PC HDD. To prevent data remnants residing on the host PC HDD, following the use of a secure PEE, the device will be configured to:
    - provide swap space (virtual memory/page file) for the secure PEE VM guest OS; and
    - ensure the secure PEE VM guest OS and application(s) write all temporary information to available allocated space on the secure PEE device.
- **Partitioning with Differentiated Access Rights:** The secure PEE will support storage partitioning and role based differentiated access rights to partitions to preserve the integrity of the VM and any stored data on the secure PEE device. Such partitioning allows separation and isolation to be achieved. In addition to the provision of a partitioning capability the secure PEE device will also allow a partition to be defined as Read-Only. A Read-Only partition will be used to protect both the integrity of the VM (from malicious software) and 'valued' data.

Figure 1 presents a conceptual model of the configuration for the secure PEE device to be considered in this paper. The secure PEE device will be configured with three partitions. The first partition containing the VM will be set to Read-Only, to protect the VM's integrity. The second partition will have Read-Write access and will be used as swap space for the VM guest OS and for any temporary files created by applications. A separate third partition with Read-Write access will be used to separate and protect any user generated data.

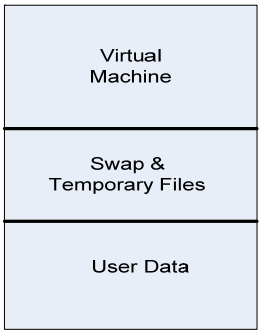

Figure 1

The secure PEE configuration presented in Figure 1 provides strong security to protect against the acquisition of data from dead forensic analysis techniques. This configuration and the aforementioned protection measures are however, unable to protect a VM based secure PEE from malicious software once the VM has been loaded into a PC.

In this paper techniques are considered to limit the opportunities of malicious software, monitoring software and live forensic techniques to acquire data from a VM based secure PEE. Through the presentation of a threat model and an attack scenario the proposition for the identification of additional protection measures for a VM based secure PEE is given. The vulnerabilities of VMs are enumerated and discussed. A set of VM protection measures are presented and an improved VM based secure PEE proposed.

The following terms are defined in James 2008 and are restated below as they are used throughout this paper:

- **secure PEE device:** the secure platform/infrastructure consisting of a USB mass storage device configured with a secure PEE, secure storage space and possibly hardware based security mechanisms/technologies (e.g. encryption and secure partitions if available).
- **secure PEE:** the trusted OS, trusted application(s), security technologies (e.g. authentication and software encryption) and the appropriately configured hardware security mechanisms (if any) of the secure PEE device.
- **secure PEE OS:** the trusted OS component of the secure PEE.
- **VM based secure PEE:** a secure PEE that utilises a VM to host the secure PEE OS.
- **trusted OS:** an OS that has been acknowledged as secure by the supplier and users. To be considered trusted the OS may have a reduced set of hardened functionality and/or been subjected to independent rigorous evaluation and testing.
- **PEE:** a portable execution environment that does not necessarily have any security technology nor has been specifically configured to be secure.
- **portable storage device:** a USB flash device (often know as a thumb drive or pen drive) or a USB HDD packaged in a portable enclosure.

THREAT MODEL AND ATTACK SCENARIO

Threats

The secure PEE protection measures discussed above provide countermeasures to the following threats:

- Acquisition of sensitive data remnants (resulting from an application storing temporary information) residing on a host PC's HDD following the completion of a network transaction through the use of dead forensic techniques; and
- As a result of loss or theft, unauthorised access is gained to the VM and sensitive data held on the device.

However, a VM based secure PEE is susceptible to the following threat:

- Malicious software or monitoring software (inserted by a hostile user or organisation) is able to capture information through the application of
    - memory acquisition including the use of live forensic techniques
    - keyboard logging
    - screenshot capture; and
    - reverse engineering.

Attack Scenario

Numerous different attack scenarios are possible. The attack scenario presented in this paper provides the context and rationale for the analysis of VM vulnerabilities and the identification of tools and techniques to be applied to limit the exploitation of the identified vulnerabilities when a VM is used in a secure PEE. Rather than consider attack options possible from an easily accessible and open Internet access centre, the attack scenario presented below is focussed on a very specific area of use.

> *VM based secure PEE devices are issued by an organisation to its employees to enable work to be performed remotely using a virtual private network (VPN) connection over the Internet. The VM based secure PEE will connect (through the VPN) to remote servers that contain sensitive/classified information, also the work performed by the employee using the VM based secure PEE may result in the generation of sensitive/classified data.*
>
> *An employee will use the VM based secure PEE device and its applications on remote host PCs that the employee considers to be safe and secure. The host PCs are executing the Windows OS. However, due to the nature of the business conducted by the organisation it is likely that technically sophisticated and capable hostile organisations will attempt to acquire data through:*
>
> - *Embedding monitoring and data acquisition software on the remote PCs that can transmit data to the hostile organisation.*
> - *Performing dead forensic analysis on the remote host PC subsequent to the use of the VM based secure PEE.*
> - *Performing live forensic data acquisition either (possibly) during or subsequent to the use of the VM based secure PEE.*

The attack scenario may appear infeasible or extreme but for certain organisations (e.g. government agencies) considering home and/or remote working for employees such an attack scenario needs to be considered as hostile organisations (e.g. foreign intelligence agencies) would use any means available to gain access to sensitive/classified data. It is conceivable that such hostile agencies will have access to sophisticated monitoring and dead & live forensics tools and techniques to enable the analysis of:

- the host PC used for processing (assuming access to the PC by the attacker is possible); and
- a captured VM based secure PEE device and the host PC used for processing.

HOW THE VM BASED SECURE PEE LIMITS THE OPPORTUNITY FOR DEAD FORENSIC ANALYSIS

The protection measures of the VM based secure PEE device considered in this paper provide strong mechanisms to protect the confidentiality of data and the integrity of the VM based secure PEE. These protection measures limit both the opportunity to acquire data using dead forensic analysis techniques when the device is at rest and to compromise the integrity of the VM when the device is operational.

Dead forensic analysis is the analysis of a digital device once processing has stopped and power has been removed. The analysis involves the examination of any aspect of the digital device's storage media to acquire data using a tool set that does not require the OS of the device to be booted. The protection measures of the VM based secure PEE device limit the opportunity to acquire data from both the device itself and the host PC HDD.

To assist the reader to appreciate the capabilities of the type of VM based secure PEE proposed in this paper an overview is given below of how the device's protection measures limit both the opportunity to:

- acquire data using dead forensic analysis techniques; and to
- compromise the integrity of the VM.

Encryption

Full hardware based device encryption prevents the acquisition of meaningful data from the VM based secure PEE storage medium. Access to decrypted data is only possible once the device has been powered and successfully authenticated. Even if the device can be successfully dismantled to bypass the authentication mechanism and allow access to the storage medium, time consuming brute force attacks would be required. Whilst brute force attacks are possible, if a strong cryptographic algorithm like AES is used then significant computing resources (i.e. very high end supercomputing centres) and large amounts of time are required to identify the cryptographic key(s) and thus acquire data.

Unlike software based encryption, hardware based encryption does not store any encryption key(s) in the host PC's random access memory (RAM) and therefore it is not possible to acquire the secure PEE device key(s) from memory. Recent research has shown it is sometimes possible to acquire keys from a PC's RAM (Haldermany et al 2008) after the PC has been powered off, however in practice such key recovery is difficult in the extreme (Hannay et al 2008).

Strong Authentication

The authentication component of a VM based secure PEE prevents access to the contents of the device and the loading of the VM into the host PC until a user has entered the correct authentication credentials. Once successful authentication has occurred access to the contents of the device is possible with the device performing on-the-fly encryption/decryption so the user can read all available data (subject to partitioning access controls). The strong authentication prevents the application of dead forensic tools (running on the host PC) accessing any data on the VM based secure PEE; as access to data is blocked until the authentication process has been completed.

Swap Space/Virtual Memory

Configuring the VM guest OS of the secure PEE to write its swap file direct to the device and also to configure applications that run under the control of the VM guest OS to write any temporary data to the device prevents data remnants residing on the host PC HDD after completion of a VM based secure PEE session. Any analysis of the host PC HDD will not reveal data generated during the use of a VM based secure PEE; although the host PC OS logs may indicate a VM was loaded and executed.

VM & Guest OS State Information

The VM and guest OS may generate data on their respective states that is required to be stored so that when another VM based secure PEE session is initiated the system state from the previous session is restored. Like the swap space and other temporary data, the files that store the VM and guest OS system state can be stored on the VM based secure PEE device; to avoid leaving data remnants on the host PC HDD.

Partitioning

Partitioning allows the separation and protection of software and data; a partition need only be mounted as required and differentiated access rights ensure only authorised access is permitted to a partition. As shown in Figure 1 the VM based secure PEE considered in this paper utilises three partitions. The first partition containing the VM will have Read-Only access to prevent any malicious software running on the host OS compromising the integrity of the VM.

The second partition will have Read-Write access and will be used as swap space for the VM guest OS, for any temporary files created by applications and system states. As identified above using a dedicated partition for all temporary data generated by the VM guest OS significantly limits the opportunity to acquire data remnants from the host PC HDD upon completion of a VM based secure PEE session.

A separate third partition with Read-Write access will be used to separate and protect any user generated data. A partition dedicated to storing user data ensures the partition containing the VM can be set to Read-Only without preventing user generated data being stored on the device.

ACQUIRING DATA FROM A VM BASED SECURE PEE

The type of VM considered in this paper utilises the OS of the host PC as an execution platform. The host PC OS can provide a platform for malicious/monitoring software. For a host PC for which no level of trust can be assumed the possibility of the host PC OS supporting attacks (through embedded malicious software) on the VM and its guest OS (& applications) is feasible. Techniques like keyboard logging and screenshot capture are typical of the type hostile acquisition methods used.

Attackers will, if logistically possible, utilise live forensic techniques to acquire sensitive data. Live forensic analysis involves the acquisition of data from a PC whilst it is still executing and therefore enables data to be acquired from the PC's memory. Another advantage of live forensic analysis is that any mounted HDD partitions that are encrypted when the data is at rest (i.e the PC is powered off) will provide full access to the plain text data when the PC is executing.

The work performed by (CSIRO 2008), (Ferrie 2007) and (Ormanday 2007) on VM vulnerabilities have provided excellent information sources to support and supplement the author's experience and knowledge.

Detecting the Presence of a VM Based Secure PEE

An attacker or embedded malicious software needs to avoid detection and therefore should only launch an attack if a VM based secure PEE has been identified. The following methods could be used to identify if a VM has been loaded and is executing:

- ***Using Standard OS Features:*** Malicious software could query any or all of the standard Windows Autorun, Task Bar, Task Manager or Windows data structures to identify if a VM is executing.
- ***Tools to Detect Processes:*** A VM could be detected by a range of specialist detection (e.g. Process Explorer) that have been created to identify executing processes; particularly if the processes are using concealment techniques.
- ***Checking OS Capabilities:*** Malicious software could search:
    - the Windows registry for entries that may identify the presence of a VM.
    - the list of loaded Windows DLLs to identify DLLs used by a VM.
    - the list of installed drivers to identify drivers used by a VM.

Memory Acquisition

The memory management function of an OS:

- controls access to, and the allocation of, the PC RAM
- implements a virtual memory system (often known as swap space or paging) which extends and optimises the use of the PC RAM by using part of the PC HDD; and
- provides memory isolation for processes.

As outlined above, to prevent virtual memory pages being placed on the host PC HDD, a VM based secure PEE device allows the VM guest OS to be configured to utilise a dedicated partition on the device for virtual memory pages. Whilst the VM based secure PEE device prevents the VM's guest OS leaving sensitive data remnants (held in virtual memory pages) on the PC HDD, the device can do nothing to protect the guest OS virtual memory from a live attack when the VM based secure PEE is executing.

As the host PC OS provides the execution environment for the VM, the PC memory (RAM and virtual memory) utilised by the VM will therefore be managed and controlled by the host PC OS. Although OS' are designed to provide process isolation (i.e. a process should not be able to interfere with the memory space of another process) it is possible for a hostile OS to interfere with a process' allocated memory; the hostile OS (or an executing hostile process) could access and modify or acquire the contents of the memory. To be successful in modifying or acquiring the contents of the memory used by the VM the attacker would need to have both a detailed knowledge of how the OS implements memory management and the design and structure of the VM and the executing application. Information on how to modify and acquire memory is becoming increasingly available due to gamers publishing details on memory hacks for games.

Techniques to modify and acquire the contents of a PC's memory include:

- ***Inserting Malicious Memory Management Software:*** An attacker, who has access to the host PC OS and has a detailed knowledge of the OS memory management architecture and design, may be able to rewrite the memory management software to allow access to the VM's allocated memory. For such an attack to be successful a detailed knowledge of the use of memory (RAM and virtual memory) by the VM is required. The advantage of embedding malicious memory management software in the OS kernel is that the user will be unaware of its existence, the disadvantage is that the attacker needs undetected access to the host PC and a highly sophisticated knowledge of the OS memory management and VM used in the secure PEE.
- ***Utilising the Memory Management API:*** The OS virtual memory management API, which allows developers to allocate, release and modify virtual memory could be used by malicious software to capture the contents of the VMs allocated memory. The advantage of using the virtual memory API is that it is a published interface that malicious software can utilise, the disadvantage is that a well written VM or guest application can use access rights on virtual memory pages to prevent malicious software acquiring data.
- ***Memory Hacking Tools:*** Generated predominately by gamers, the increasing number of available memory hacking tools can be used to gain access to memory; such tools include:
    - Memory Hacking Software – MHS (Spiro 2008): allows a user to search and change data via a graphical interface.
    - Tsearch (Corsica Productions 2008): provides a similar capability to MHS.
    - FU Rootkit (FU Project 2008): allows kernel data structures to be accessed via a device driver that gets installed.

  The advantage of memory hacking tools is that they are readily downloadable from various web sites, however the main disadvantage is that it is difficult for the attacker to use these tools without detection as concurrent access (with the VM based secure PEE user) to the host PC is required.
- ***Memory Dump:*** The standard OS memory dump upon process termination can reveal sensitive data being processed by the VM. Malicious software can cause the VM process to terminate and a process memory dump to occur. An advantage of causing memory dumps to occur is that relatively simple malicious software can cause a dump, however the main disadvantage is knowing when to trigger the memory dump to acquire sensitive data; causing multiple dumps to occur (i.e. numerous memory dumps will increase the probability of acquiring sensitive data) will make the user suspicious and/or end the VM based secure PEE session due to the multiple process terminations.
- ***Firewire Access:*** Although an overt action, it is possible to acquire the contents of a PC's RAM using a Firewire direct memory access function (Woodward et al 2008). The technique involves connecting another PC to the target PC via a Firewire port and executing a Firewire memory access tool. The tool will allow the contents of the target PC's memory to be copied to the PC running the Firewire memory access tool. The advantage of this approach is that the whole contents of the RAM can be acquired. The disadvantage is that it is an overt action that would be extremely difficult to conceal from the user.
- ***Cold Boot Memory Access:*** As discussed above, recent research (Haldermany et al 2008) has shown it is possible to acquire the contents of a PC's RAM after the PC has been shutdown if the acquisition is performed soon after the poweroff has occurred. To work successfully the memory needs to be kept cold. The advantage of this approach is that it is possible to analyse the PC after the VM based secure PEE has been used and in theory acquire potentially sensitive information. The disadvantages include being able to gain access to the PC in a timely fashion to acquire meaningful data and the need to have cold RAM.

Keyboard Logging

Keyboard logging is most often used to obtain authentication credentials and is implemented by malicious/monitoring software executing on the host PC OS and intercepting keyboard input which is either:

- stored in an unused area of the host PC HDD and retrieved by the attacker at a later time; or
- transmitted directly to the attacker as the keyboard input is received from the malicious/monitoring software.

Attackers are able to embed malicious software into a process and receive input from a keyboard due to the way OS's allow the interception of messages (using hooks) sent by concurrently executing processes. Attackers also use the information gathering capabilities available in application programming interfaces (APIs) to understand the capabilities of executing processes. Attackers are able to abuse the process communication and API features of OS' to introduce keyboard logging capabilities. Keyboard logging can be implemented in the following ways:

- ***Changing Device Drivers:*** A device driver is part of the OS kernel and has unrestricted access to the host PC hardware. A device driver based keyboard logger is able to communicate directly with the keyboard and capture all input. An advantage of this type of key logger is that it is very hard to detect its presence within the OS kernel. However, disadvantages include:
    - the requirement for administrator privilege and access to the OS kernel to be able to install the device driver (into the kernel); and
    - due to the low level nature of the key logger it is not possible to determine the context of the input, i.e. all input is captured and filtering authentication credentials (or other sensitive information) from other input at the device driver level is not possible.
- ***Utilising Application Programming Interface Hooks:*** The Windows OS provides the capability, through API hooks (Ivanov 2002), to intercept inter-process messages. The API hook allows a malicious/monitoring dynamic link library (DLL[2]) to be inserted into a process which can intercept and forward messages to a keyboard logger process. The large amount of documentation on the use of API hooks, available on the Internet, enables an attacker to readily implement this type of key logger. Another advantage is that under certain circumstances it is possible to introduce an API hook based key logger without access to the host PC. A disadvantage in the use of an API based key logger is that it can be detected as it is an executing process.
- ***Exploiting Debug Code Injection:*** Debug code injection is a Windows facility provided to enable debugging of executing programs (where the source code is not available). The technique involves modifying the program's binary code (in the PC RAM) through the injection of interceptor code to output specific information. Debug code injection can be exploited by attackers to inject key logging code. Like API hooks an advantage of debug code injection based key loggers is that lots of documentation exists to enable an attacker to build such a key logger; Microsoft supports the Detours DLL to enable code injection. An important disadvantage is that the attacker is required to understand the executing process into which the key logger code is to be injected.
- ***Replacing Dynamic Link Libraries:*** Replacing a DLL with a DLL with the same name is another approach to installing a key logger. The key logger is implemented within a DLL function that a process utilises in 'good faith'. An advantage is that a malicious substituted DLL is hard to detect, however a disadvantage for the attacker is that every function in the original DLL needs to be present in the malicious replacement DLL.

The following techniques can be used to prevent a key logger process/application from being detected:

- ***Covert Existence:*** Methods include removing the process from the taskbar, hiding the application window, hiding tracing information of the application installing and preventing the process being listed in the task manager.
- ***Alternative PC:*** The key logger can be installed on an alterative networked PC.
- ***Renaming:*** Methods include modifying the key logger process' signature and renaming files used by the key logger.

[2] A DLL provides a capability to allow multiple programs to share a set of library functions. In the case of a VM based secure PEE the library function is linked to the VM at run-time with the host PC OS performing the binding.

Screenshot Capture

A captured image of the screen can be used to acquire data from a VM based secure PEE. Windows provides a number of capabilities to capture and store screen images, also it is possible to use the processing capabilities of a PC graphics card to acquire screenshots. Screenshot capture can be implemented in the following ways:

- ***Changing Graphics Card Device Drivers:*** A PC uses a specialist graphics card to manage and present the output on the PC screen. A graphics card device driver based screenshot capturer can capture all screen output and save as bit streams in a file(s). To successfully implement such a malicious device driver requires the attacker to have a detailed knowledge of the host PC's graphic card design. An advantage of this type of screenshot capture is that it is very hard to detect its presence within the OS kernel. However, a major disadvantage is that due to the low level nature of the device drivers it is not possible to determine the context of the output and therefore large volumes of screen displays will be captured.
- ***Standard Print Screen Capability:*** Windows provides a capability that allows the print screen key (present on almost all keyboards) to be pressed and an image of the screen is captured and placed on the Windows 'clipboard'. Malicious software can use the print screen capability by emulating the key press, once a screenshot image has been written to the clipboard the malicious software can use the standard clipboard API to move the image to a file. An advantage in using the print screen capability is that it is a reliable existing feature that can be easily implemented. A disadvantage is that the capability can be disabled.
- ***Graphics Device Interface (GDI+):*** The Windows GDI+ API is responsible in the Windows OS for managing the presentation of output to the screen. GDI+ presents each screen as a pixel map. The GDI+ API can be used by malicious software to make copies of the pixel maps and move the map to an alterative part of the PC RAM and then be exported as an image to a file. An advantage of using the GDI+ based screenshot capture is that it cannot be disabled. A disadvantage is that non-trivial malicious software needs to be developed (in comparison with the print screen approach) to exploit the capability.
- ***DirectX API:*** The DirectX multimedia library is a comprehensive capability used for presenting videos and graphics. DirectX uses buffers to store the screen display before sending it to a graphics card for display on the screen. Malicious software can use the DirectX API to retrieve the contents of the buffers, convert into a bitmap and export as an image to a file. Like GDI+, the DirectX capability cannot be disabled, but similarly it requires relatively complex malicious software to produce useful output and of course can only be exploited if the VM guest application utilises DirectX.

Reverse Engineering

Reverse engineering a VM based secure PEE will enable an attacker to understand how the VM (and its OS and applications) work by reconstructing the source code from binary code. Once a VM (and its OS and applications) has been reconstructed into source code the attacker can identify vulnerabilities and compile a plan of attack. Reverse engineering a VM based secure PEE will obviously require a highly skilled and experienced attacker.

Reverse engineering is generally performed by a range of techniques including:

- debug software to step through the VM code.
- emulation software which will allow snapshots of the state of the VM to be taken and analysed
- process memory dumps that can be analysed.
- forensic analysis and memory hacking tools to understand how the VM utilises the HDD and memory.
- packet sniffing software to analyse network traffic.

The advantage of reverse engineering is that it enables the attacker to build a comprehensive understanding of the VM based secure PEE implementation to enable attacks to occur. The disadvantages include the need to acquire a VM based secure PEE device to reverse engineer, and of course reverse engineering by itself does not allow data to be acquired; the technique needs to be used in collaboration with other techniques to mount a successful attack.

PREVENTING THE LIVE ACQUISITION OF DATA FROM A VM BASED SECURE PEE

It has been shown above that there are potentially a number of exploitable vulnerabilities in VMs that the protection measures of the secure PEE device cannot counter. However, a range of techniques exist that can be constructed into a set of countermeasures to address the vulnerabilities. The work performed by CSIRO (CSIRO 2008) provided valuable input in the identification of many of the techniques presented.

Preventing the Detection of a VM

If the presence of a VM based secure PEE executing on a host PC can be successfully hidden then an attacker and/or embedded malicious software will be ineffective. Most of the techniques to prevent detection counter the detection techniques identified above. Techniques to hide the presence of a VM include:

- ***OS Features Avoidance:*** The VM should avoid execution by Autorun and prevent an icon appearing in the Task Bar. Using an application and process naming convention that has no association to VMs may avoid detection by both the Task Manager and searching through Windows data structures.
- ***Hiding from Scanning/Monitoring Tools:*** Preventing the detection of a VM by sophisticated scanning/monitoring tools may be difficult and will depend upon the capabilities of each tool. For instance, the freeware tool Process Explorer (SysInternals 2006) provides detailed information about a process icon, command-line, full image path, memory statistics, user account and security attributes. Some of these process attributes would be hard to conceal by the VM.
- ***OS Capability Avoidance:*** The VM should be constructed to avoid placing entries in the Windows registry, and where possible not use the host PC OS DLLs. However, avoiding the use of installed OS drivers is extremely unlikely, for obvious reasons.

Memory Acquisition

The following techniques can be used to prevent or limit the opportunity from memory acquisition based attacks:

- ***VM and Memory Address Obfuscation:*** Obfuscation is a technique used to prevent the reverse engineering of software. Software obfuscation is implemented by creating hard to interpret code by masking the language syntax and grammar. The principles of obfuscation can be used to make it difficult for an attacker to locate sensitive data in memory. Obfuscation techniques can be applied as follows:
    - *Memory Address Obfuscation:* By changing (randomising) the address space of memory an attacker cannot acquire meaningful data from consecutive memory space. Address obfuscation can be implemented by the guest OS virtual memory management system. The disadvantages are the degraded performance (due to retrieving data from non-consecutive memory locations) and complexity to implement.
    - *VM Obfuscation:* By implementing a virtual memory management system within the VM (in addition to the host PC OS and guest OS virtual memory management systems) the level of complexity for the attacker and obscurity of data will increase. The disadvantages of VM obfuscation are both the complexity to implement and the performance degradation due to multiple virtual memory management systems operating simultaneously.
- ***Complex Data Structures:*** Approaches to making data structures complex and therefore difficult for an attacker to understand include:
    - *Mixing variables:* The methodology involves placing parts of one variable into another. Records are kept of where each of the various parts of the variables are located to enable reassembly of the correct value to occur. The disadvantage of this approach is that the attacker may be able to read the variables before they are mixed or obtain the record of mixing locations and then perform reassembly; there is also the degradation in performance due to mixing and reassemble of variables.
    - *Assigning the wrong data type:* By storing integers as strings, strings as integer arrays, etc, it is possible to make some memory hacking tools become confused and possibly crash. The disadvantage is that the sophisticated hacker is able to circumvent this technique.

- ***Memory Encryption:*** Encryption can be used to protect the contents of memory. Whilst encryption is probably the strongest mechanism available to prevent the attacker gaining meaningful data from memory, the cryptographic algorithm and encryption keys must be stored in plaintext in memory to enable execution. The attacker therefore may be able to acquire the algorithm and keys and reverse engineer the VM. Performance is also likely to be an issue.
- ***Preventing Overflow Attacks:*** Causing memory to overflow is a technique used by attackers to allow inserted malicious code to execute, i.e. code is inserted into available process memory space, then the code is able to execute when a buffer overflow is invoked. Overflow attacks can be prevented by setting the virtual memory pages with no execution rights. The advantage of this technique is that it is a well documented technique, however a disadvantage is that an OS has the capability to mark memory pages as "no execute".

Keyboard Logging

The following approaches can be used to prevent or limit the opportunity from keyboard logging based attacks:

- ***On Screen Keyboard:*** A very popular technique to counter keyboard logging is to use an on screen keyboard (the keyboard is a screen image) where keystrokes are entered by pointing to the respective character with the mouse pointer and "clicking"; other methods (e.g. a stylus) can be used to select the required character on the screen. An on screen keyboard does not mean that it has to be used for all keyboard input, for instance the on screen keyboard could be used just for the input of authentication credentials; a number of Internet banking applications use this approach. The on screen keyboard can be implemented as:
    - a feature in the VM.
    - a feature in the guest OS; or
    - part of an application.
- ***Embedded Authentication Credentials:*** A technique to protect authentication credentials from capture is to embed them in a string of random text (Herley et al 2006). This technique can be used with passwords and personal identification numbers (pin) as follows:
    - when the password/pin dialogue box appears on the screen then perform the following steps:
        1. move out of password/pin dialogue box focus
        2. type any random input
        3. move into focus for the password/pin dialogue box and type the next character of the password/pin
        4. repeat the above steps 1 to 3 until the password/pin has been entered.

  Any keyboard logger will gather a large string of characters, as the keyboard logger will capture all input, however the password/pin dialogue box will only receive the characters entered whilst in focus. Any parsing of the input by malicious software will prevent the password/pin being identified.
- ***Preventing the use of malicious API Hooks:*** A technique that works to counter an API hook based key logger is to enter as the very first hook in the hook list a trusted hook to a trusted DLL. The trusted DLL will be responsible for passing only input to the VM and/or its guest OS. The trusted hook will only work if it is the very first hook.
- ***Simple Encryption:*** To counter a keyboard logger an application expecting input can issue a simple encryption key to the user which is used to encrypt the input. The keyboard logger would intercept meaningless characters but the application would be able to decrypt the input because it has the encryption key. The encryption key would need to be very simple to enable the user to calculate and enter data in a timely manner.

Screenshot Capture

The following approaches can be used to prevent or limit the opportunity of data acquisition from screenshot capture:

- ***Disabling the Standard Print Screen Capability:*** Windows allows the print screen capability to be disabled. However, this approach will not prevent sophisticated attackers as disabling does not prevent GDI+ and DirectX based malicious software.
- ***Use of Graphic Card Overlays:*** An approach to counter GDI+ and DirectX based screenshot capture is to use the overlay capabilities available in most PC graphics cards. By using the graphics card API code can be written that bypasses the OS and directly modifies the screen space to produce an overlay. An overlay allows graphics to be placed over the graphics being displayed by an application but the overlayed graphics cannot be captured by screenshot loggers because the functions performing the overlay are not part of the OS.
- ***Scrambling Screen Output:*** To prevent a screenshot logger capturing useful information the output on the screen can be scrambled with the exception of a small restricted area. The user accesses only the restricted area of the screen for sensitive operations. Whilst a screenshot capturer will capture the restricted area, if this area is kept small it may be difficult to identify meaningful information amongst the scrambled data.

Reverse Engineering

The main countermeasure adopted to prevent reverse engineering is software obfuscation (Ogiso et al 2003). As outlined above obfuscation is the process making software hard to read. Software obfuscation techniques include software compression, keyword substitution and the use/non-use of whitespace to mask language syntax and grammer.

In theory encryption could be used to encrypt the whole VM executable with only the encrypt/decrypt algorithm in plaintext, however in practice such an approach would require the cryptographic algorithm to execute under the control of the host PC OS and could therefore be susceptible to attack and the cryptographic algorithm compromised.

Detection of Malicious Software

Performing checks to detect malicious software could be performed as an alternative, or as an additional measure to the use of the aforementioned complex countermeasures. The issue with using detection as the sole countermeasure is that only known malicious techniques will be identified; also kernel level malicious software is unlikely to be detected. The best strategy is to use detection techniques to support other countermeasures. The following detection techniques could be used:

- ***Detection of Malicious DLLs:*** The host PC OS DLLs and API hooks (used by the VM based secure PEE) could be monitored by checking the modules loaded prior to execution and comparing against known modules for the DLL. Any identified unknown modules can be treated as malicious and action taken. This detection technique would require a DLL examination tool to be executed prior to loading the VM.
- ***Use of Anti-Virus/Anti-Spyware Tools:*** Prior to loading the VM, anti-virus and anti-spyware tools can be executed to identify and remove any known malicious software. The tools would need to be appropriate to identifying the type of malicious software that can subvert VMs, which may mean specific bespoke tools are required.

The advantage of the above two detection techniques is that an untrusted host PC OS can be identified before the VM is loaded. However, the disadvantages include:

- the time required to perform OS scanning and DLL examination
- the tools will only be as good as the database of known problems
- the inconvenience of having to determine the course of action to take if an untrusted environment is identified, i.e. is it feasible to remediate the host PC OS or should an alternative host PC be used?

An alternative detection technique is:

- ***In Parallel Monitoring:*** This technique involves a detection process(es) running in parallel with the VM. Such a process could execute on the host PC OS or within the VM, and would monitor:
    - any unexpected changes to the VM
    - if another process attempts to access the VM
    - the commencement of any suspicious processes on the host PC OS.

The advantage of this technique is that it does not require time consuming pre-processing detection software to be executed before the VM can be loaded. The disadvantage of parallel monitoring is that it is conceivable that the VM could be exploited before the detection process has identified any malicious software.

AN IMPROVED VM BASED SECURE PEE THAT LIMITS THE OPPORTUNITY FOR LIVE ACQUISITION OF DATA FROM A VM BASED SECURE PEE

As the specific brand of VM has not been explicitly stated the reader may have assumed a commercial off the shelf (COTS) product would be used, which would also be the author's desired approach. Access would be required to the VM source code to implement many of the countermeasures proposed in this paper. To preserve its intellectual property most commercial organisations do not publish a products source code. Therefore to achieve the desired security for a COTS VM the product vendor would be required to change the VM. Alternative approaches to using a COTS VM could include:

- developing a bespoke VM that implements all of the required countermeasures; or
- using a freeware VM and adding the countermeasure, however most freeware requires any added functionality to be published and made freely available – which would obviously then be available to an attacker.

In proposing an improved VM based secure PEE the commercial and logistical viability of implementing the required countermeasures are not consider. Observations may be made on the feasibility of implementing a countermeasure, otherwise it is assumed that all required functionality is implementable.

Summary of Attack Scenario

Important aspects of the attack scenario described above can be summarised as:

- The issuer of the VM based secure PEE has to assume the host PC is not secure even though the user may consider it to be secure.
- If the user considers the host PC to be safe, then it will most likely be located in an environment where physical access will be difficult. However, it must be assumed the attacker is able to gain physical access and have sufficient time to examine the HDD using dead forensic tools, install malicious software and possibly use live forensic tools if the PC has been left powered on.
- The host PC will be connected to the Internet and therefore malicious software can be pushed to the host PC.

An Improved VM Based Secure PEE

Table 1 below summarises the set of countermeasures that can limit the opportunity to perform live acquisition of data from a VM based secure PEE. For each countermeasure:

- confirmation is given on whether the countermeasure will be used in an improved VM based secure PEE; and
- comments provided on the effectiveness, useability and implementation.

The decision to include a countermeasure is based upon:

1. does the countermeasure address a vulnerability that could be exploited within the given attack scenario.
2. the level of inconvenience the implemented countermeasure will cause the user, i.e. will the VM based secure PEE now be slower and more difficult to use.

3. in practice could the countermeasure be implemented, ignoring the commercial and logistical viability.

Required Countermeasures for Improved VM Based Secure PEE		
Countermeasure	**To Be Used**	**Comments with respect to effectiveness, useability and implementation in an improved VM Based Secure PEE**
Preventing the detection of a VM – OS features avoidance	**Yes**	Relatively simple to implement and effective.
Preventing the detection of a VM – hiding from monitoring tools	No	Counters an unlikely attack, also difficult to implement.
Preventing the detection of a VM – OS capability avoidance	**Yes**	Even if a bespoke VM is developed it maybe infeasible to avoid the use of DLLs & registry entries by a VM.
Memory acquisition – VM & memory address obfuscation	**Yes**	Only possibly if a bespoke VM developed.
Memory acquisition – complex data structures	No	Complex to implement and unlikely to provide an effective measure.
Memory acquisition – memory encryption	No	Likely to be difficult to implement and resultant implementation slow to execute.
Memory acquisition – preventing overflow attacks	**Yes**	For COTS product it may not be possible to implement.
Keyboard logging – on screen keyboard	**Yes**	Can be built into secure PEE application.
Keyboard logging – embedded authentication credentials	**Yes**	Implemented through user procedure.
Keyboard logging – preventing the use of malicious API hooks	**Yes**	For COTS product it may not be possible to implement.
Keyboard logging – simple encryption	No	Likely to be complex for user and easy to break by attacker.
Screenshot capture – disabling the print screen capability	**Yes**	Can be implemented with guest OS
Screenshot capture – use of graphic card overlays	No	As graphic card APIs will differ any overlay countermeasure would not be portable across a arrange of different host PCs.
Screenshot capture – scrambling screen output	**Yes**	Can be implemented by the secure PEE application.
Reverse engineering - obfuscation	**Yes**	Only possible if a bespoke VM developed.
Reverse engineering - encryption	No	Whilst in theory this countermeasure may be possible, in practice it is likely only parts of the binary can be encrypted.
Detection of malicious software – detection of malicious DLLs	**Yes**	Implement within a detection application.
Detection of malicious software – use of anti-virus tools	**Yes**	Likely to require bespoke tools as standard anti-virus & anti-spyware are unlikely to find all the malicious software designed to attack a VM based secure PEE.
Detection of malicious software – in parallel monitoring	**Yes**	Likely to require bespoke tools as standard anti-virus & anti-spyware are unlikely to find all the malicious software designed to attack a VM based secure PEE.

Table 1

CONCLUSION

A VM based secure PEE device provides a secure platform for portable computing, however it is susceptible to attack and the live acquisition of data due to the VM's reliance upon the underlying host PC OS. In this paper a range of countermeasures have been proposed that can address the VM's vulnerabilities. An improved VM based secure PEE that incorporates countermeasures that are effective, usable and implementable (in theory) has been proposed. The field of virtualisation is developing rapidly. It is possible that VM vendors may consider implementing some of the countermeasures identified in this paper for security applications.

Future work could include a detail review of available VMs to determine the VM most likely to be suitable for use in a secure PEE either because it has some of the countermeasures required or could readily be changed to

implement the required countermeasures. Other work could include a proof of concept of the countermeasures in a VM based secure PEE

REFERENCES

Corsica Productions (2008). "Tsearch - a memory scanner/debugger utility." Retrieved October, 2008, from http://duckduckgo.com/TSearch.

CSIRO (2008). "Virtual Machines: An Initial Analysis of Threats and Remedial Actions."

Ferrie, P. (2006) Attacks on Virtual Machine Emulators. Volume, DOI:

FU Project (2008). "FU Rootkit." Retrieved October, 2008, from http://www.rootkit.com/project.php?id=12.

Hannay, P. W., A (2008). Cold Boot Memory Acquisition: An Investigation into Memory Freezing and Data Retention Claims. The 2008 International Conference on Security & Management, Las Vegas, Navada.

Herley, C. F., D (2006). How To Login From an Internet Cafe Without Worrying About Keyloggers. Symposium on Usable Privacy and Security CMU.

Ivanov, I. (2002). "API Hook Revealed." Retrieved October, 2008, from http://www.codeproject.com/KB/system/hooksys.aspx.

J. Alex Haldermany, S. D. S., Nadia Heningery, William Clarksony, William Paulx, and A. J. F. Joseph A. Calandrinoy, Jacob Appelbaum, and Edward W. Felten (2008). Lest We Remember: Cold Boot Attacks on Encryption Keys. Proc. 2008 USENIX Security Symposium.

James, P. (2008). Secure Portable Execution Environments: A Review of Available Technologies. 6th Australian Information Security Conference, Perth.

OGISO, Y. S., M SOSHI, A MIYAJI (2003). "Software Obfuscation on a Theoretical Basis and Its Implementation." IIEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences Vol.E86-A (1): 176-186.

Ormandy, T. (2007) An Empirical Study into the Security Exposure to Hosts of Hostile Virtualized Environments. CanSecWest Volume, DOI:

Smith, J. (2005). "The Architecture of Virtual Machines." Computer 38(5): 32-38.

Spiro, L. (2008). "Memory Hacking Software (MNS)." Retrieved October, 2008, from http://www.memoryhacking.com/.

SysInternals. (2006). "Process Explorer." Retrieved October, 2008, from http://live.sysinternals.com/.

Woodward, A. H., P (2008). Forensic implications of using the FireWire memory exploit with Microsoft Windows XP. The 2008 International Conference on Security & Management, Las Vegas, Navada.

COPYRIGHT