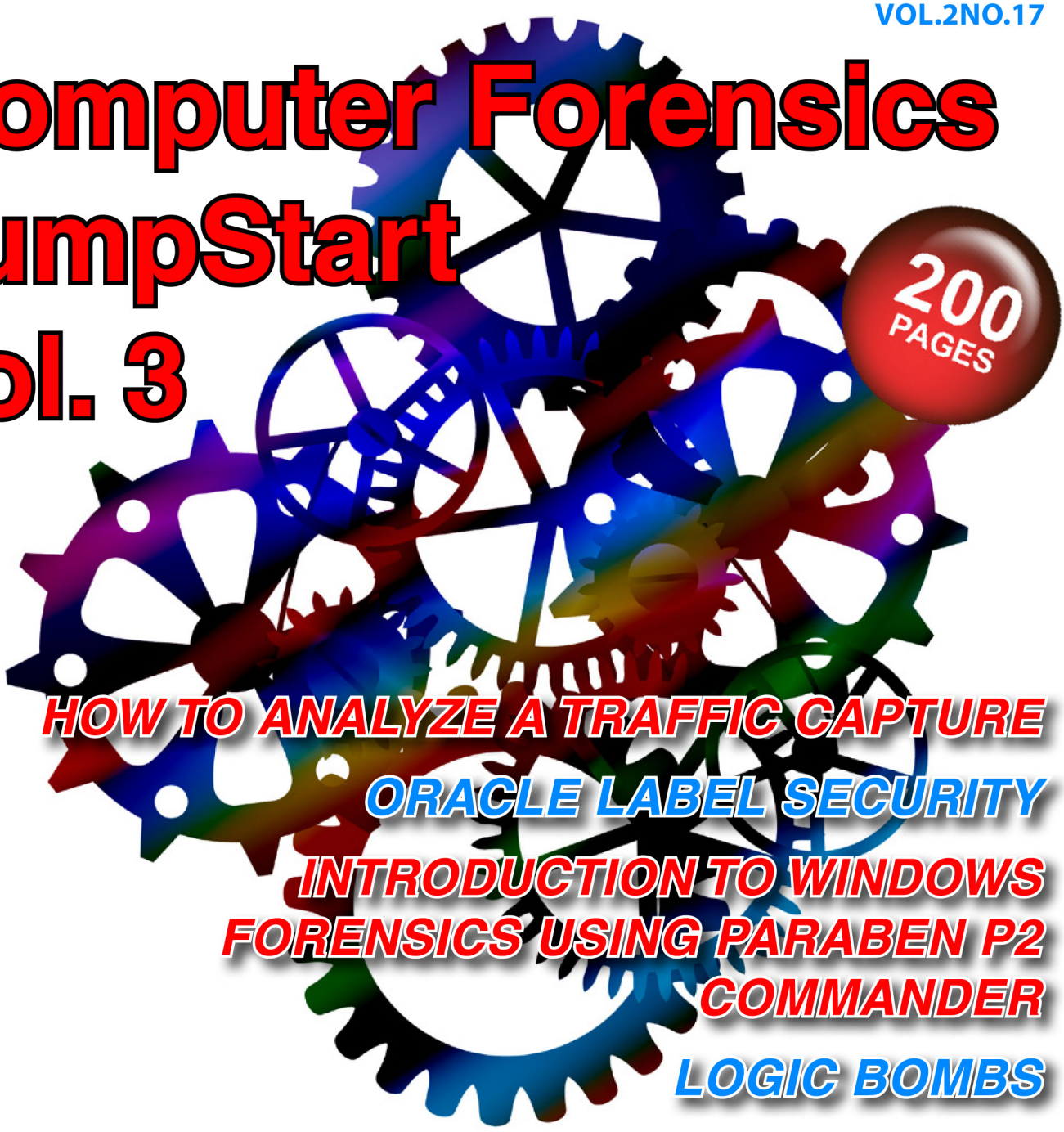
eForensics Magazine

Computer

VOL.2NO.17

Computer Forensics JumpStart Vol. 3



HOW TO ANALYZE A TRAFFIC CAPTURE

ORACLE LABEL SECURITY

INTRODUCTION TO WINDOWS FORENSICS USING PARABEN P2 COMMANDER

LOGIC BOMBS

STEP-BY-STEP TO ASSESS IT SYSTEM CONTROLS

Issue 17/2013 (21) October ISSN 2300-6986

INTRUSION DETECTION USING A VIRTUAL MACHINE ENVIRONMENT

by Niranjan P. Reddy

Malware attacks against single hosts and networks are extremely dangerous and could compromise the security of the entire network. Protection from malware is one of the top worries for system administrators who have to ensure that there are no unnecessary threats to their systems. These threats can cause an adverse effect on a running business or some other mission critical operations. Over the past few years intrusion detection and other security measures have gained critical importance in the fight against malware. Selecting the right IDS is the key to mitigate malware attacks. This article discusses an attempt to create a more robust IDS while mentioning the limitations of traditional detection systems.

What you will learn:

- Limitations of HIDS, NIDS
- Virtual Machine mechanism for Intrusion Detection

What you should know:

- What is an IDS
- Types of IDS : HIDS, NIDS
- Basic understanding of Virtual Machines

Today's architectures for intrusion detection force the IDS designer to make a difficult choice. If the IDS resides on the host, it has an excellent view of what is happening in that host's software, but is highly susceptible to attack. On the other hand, if the IDS resides in the network, it is more resistant to attack, but has a poor view of what is happening inside the host, making it more susceptible to evasion. This article presents an architecture that retains the visibility of a host-based IDS, but pulls the IDS outside of the host for greater attack resistance. By applying intrusion detection techniques to virtual machine based systems, the intrusion detection system is kept out of reach from intruders.

COUNTERING THE INTRUSION DETECTION SYSTEM

Widespread study and deployment of intrusion detection systems has led to the development of increasingly sophisticated approaches to defeating them. Intrusion detection systems are defeated either through attack or evasion. Evading an IDS is achieved by disguising malicious activity so that the IDS

fails to recognize it. Attacking an IDS involves tampering with the IDS or components it trusts to prevent it from detecting or reporting malicious activity.

VISIBILITY V/S RISK

Countering these two approaches to defeating intrusion detection has produced conflicting requirements. On one hand, directly inspecting the state of monitored systems provides better visibility. Visibility makes evasion more difficult by increasing the range of analyzable events, decreasing the risk of having an incorrect view of system state, and reducing the number of unmonitored avenues of attack. On the other hand, increasing the visibility of the target system to the IDS frequently comes at the cost of weaker isolation between the IDS and attacker. This increases the risk of a direct attack on the IDS. Nowhere is this trade-off more evident than when comparing the dominant IDS architectures: network-based intrusion detection systems (NIDS) that offer high attack resistance at the cost of visibility, and host-based intrusion detection systems (HIDS) that offer high visibility but sacrifice attack resistance. Another problem that arises is the difficulty in getting reliable information from a compromised system. Once an intrusion has occurred, the monitoring data coming from such system is no more reliable, as the intruder can disable or modify the system monitoring tools in order to hide his/her presence.

HIDS

A host-based intrusion detection system offers a high degree of visibility as it is integrated into the host it is monitoring, either as an application, or as part of the OS. The excellent visibility afforded by host-based architectures has led to the development of a variety of effective techniques for detecting the influence of an attacker, from complex system call trace to integrity checking and log file analysis, to the esoteric methods employed by commercial anti-virus tools. In HIDS, anti-threat applications such as firewalls, antivirus software and spyware-detection programs are installed on every network computer that has two-way access to the outside environment such as the Internet.

NIDS

Network-based intrusion detection systems offer significantly poorer visibility. They cannot monitor internal host state or events, all the information they have must be gleaned from network traffic to and from the host. Limited visibility gives the attacker more room to maneuver outside the view of the IDS. An attacker can also purposefully craft their network traffic to make it difficult or impossible to infer its impact on a host. The NIDS has in its favor that, it retains visibility even if the host has been compromised. In NIDS, anti-threat software is installed only at specific points such as servers that interface between the outside environment and the network segment to be protected.

VIRTUAL MACHINE APPROACH

Virtual Machines provide a strong isolation between the virtual environment and the underlying real system and hence can also be used to improve the security of a computer system in face of attacks to its network services. This allows us to pull our IDS “outside” of the host it is monitoring, into a completely different hardware protection domain, providing a high-confidence barrier between the IDS and an attacker’s malicious code. The Virtual Machine Monitor (VMM) also provides the ability to directly inspect the hardware state of the virtual machine that a monitored host is running on. Consequently, we can retain the visibility benefits provided by a host-based intrusion detection system. Finally, the VMM provides the ability to interpose at the architecture interface of the monitored host, yielding even better visibility than normal OS-level mechanisms by enabling monitoring of both hardware and software level events. This ability to interpose at the hardware interface also allows us to mediate interactions between the hardware and the host software, allowing to us to perform both intrusion detection and hardware access control. As we will discuss later, this additional control over the hardware lends our system further attack resistance.

An IDS running outside of a virtual machine only has access to hardware-level state (e.g. physical memory pages and registers) and events (e.g. interrupts and memory accesses), generally not the level of abstraction where we want to reason about IDS policies. We address this problem by using our knowledge of the operating system structures inside the virtual machine to interpret these events in OS-level semantics. This allows us to write our IDS policies as high-level statements about entities in the OS, and thus retain the simplicity of a normal HIDS policy model.

VMI IDS

Intrusion detection systems attempt to detect and report whether a host has been compromised by monitoring the host’s observable properties, such as internal state, state transitions (events), and I/O activity.

An architecture that allows more properties to be observed offers better visibility to the IDS. This allows an IDS's policy to consider more aspects of normative host behavior, making it more difficult for a malicious party to mimic normal host behavior and evade the IDS. A VMI IDS directly observes hardware state and events and uses this information to extrapolate the software state of the host. This offers visibility comparable to that offered by an HIDS. Directly observing hardware state offers a more robust view of the system than that obtained by an HIDS, which traditionally relies on the integrity of the operating system. This view from below provided by VMI-based IDS allows it to maintain some visibility even in the face of OS compromise. Similar to NIDS, VMI-based IDS retains visibility even if the host has been compromised.

COMPARING HIDS, NIDS AND VMI IDS

VMI and network-based intrusion detection systems are strongly isolated from the host they are monitoring. This gives them a high degree of attack resistance and allows them to continue observing and reporting with integrity even if the host has been corrupted. This property has tremendous value for forensics and secure logging. In contrast, a host-based IDS will often be compromised along with the host OS because of the lack of isolation between the two. Once the HIDS is compromised, it is easily blinded and may even start to report misleading data, or provide the adversary with access to additional resources to leverage for their attack.

Host-based intrusion detection tools frequently operate at user level. These systems are quite susceptible to attack through a variety of techniques once an attacker has gained privileged access to a system. Some systems have sought to make user-level IDSs more attack resistant through "stealth," i.e. by hiding the IDS using techniques similar to those used by attackers to hide their exploits, such as hiding IDS processes by modifying kernel structures and masking the presence of IDS files through the use of steganography and encryption. Current systems that rely on these techniques can be easily defeated.

In host-based IDS, an IDS crash will generally cause the system to fail open. In user-level IDS it is impossible for all system activity to be suspended if the IDS do crash, since it relies on the operating system to resume its operation. If the IDS is only monitoring a particular application, it may be possible to suspend that application while the IDS is restarted. A critical fault in kernel- based IDS will often similarly fail open. Since the IDS runs in the same fault domain as the rest of the kernel, this will often cause the entire system to crash or allow the attacker to compromise the kernel.

Unfortunately, when NIDSs do fall prey to an attack they often fail open as well. Consider a malfunction in an NIDS that causes the IDS to crash or become overloaded due to a large volume of traffic. This will virtually always cause the system to fail open until such time as the NIDS restarts. Failing closed in an NIDS is often not an option as the network connection being monitored is often shared among many hosts, and thus suspending connectivity while the IDS restarted would amount to a considerable denial-of-service risk.

In VMI-based IDS the host can be trivially suspended while the IDS restarts in case of a fault, providing an easy model for fail-safe fault recovery. In addition, because a VMI IDS offers complete mediation of access to hardware, it can maintain the constraints imposed by the operating system on hardware access even if the OS has been compromised, e.g. by disallowing the network card to be placed into promiscuous mode. So the idea is to have a virtual machine introspection, an approach to intrusion detection which co-locates an IDS on the same machine as the host it is monitoring and leverages a virtual machine monitor to isolate the IDS from the monitored host. The activity of the host is analyzed by directly observing hardware state and inferring software state based on a prior knowledge of its structure. This provides high evasion resistance in the face of host compromise, provides high attack resistance due to strong isolation, and provides the unique capability to mediate access to host hardware, allowing hardware access control policies to be enforced in the face of total host compromise.

VMM

A virtual machine monitor (VMM) is a thin layer of software that runs directly on the hardware of a machine. The VMM exports a virtual machine abstraction (VM) that resembles the underlying hardware. This abstraction models the hardware closely enough that software which would run on the underlying hardware can also be run in a virtual machine. VMMs virtualize all hardware resources, allowing multiple virtual machines to transparently multiplex the resources of the physical machine [16]. The operating system running inside of a VM is traditionally referred to as the guest OS, and applications running on the guest OS are similarly referred to as guest applications.

LEVERAGING THE VMM

VMI IDS leverages three properties of VMMs:

ISOLATION

Software running in a virtual machine cannot access or modify the software running in the VMM or in a separate VM. Isolation ensures that even if an intruder has completely subverted the monitored host, he still cannot tamper with the IDS.

INSPECTION

The VMM has access to all the state of a virtual machine: CPU state (e.g. registers), all memory, and all I/O device state such as the contents of storage devices and register state of I/O controllers. Being able to directly inspect the virtual machine makes it particularly difficult to evade a VMI IDS since there is no state in the monitored system that the IDS cannot see.

INTERPOSITION

Fundamentally, VMMs need to interpose on certain virtual machine operations (e.g. executing privileged instructions). A VMI IDS can leverage this functionality for its own purposes. For example, with only minimal modification to the VMM, a VMI IDS can be notified if the code running in the VM attempts to modify a given register.

VMM offers other properties that are quite useful in a VMI IDS. For example, VMMs completely encapsulate the state of a virtual machine in software. This allows us to easily take a checkpoint of the virtual machine. Using this capability we can compare the state of a VM under observation to a suspended VM in a known good state, easily perform analysis off-line, or capture the entire state of a compromised machine for forensic purposes.

CONCLUSION

Virtual Machine Introspection (VMI) is an approach to intrusion detection which co-locates an IDS on the same machine as the host it is monitoring and leverages a virtual machine monitor to isolate the IDS from the monitored host. The activity of the host is analyzed by directly observing hardware state and inferring software state based on a prior knowledge of its structure. This approach shows certain advantages: maintain high visibility, provides high evasion resistance in the face of host compromise, provides high attack resistance due to strong isolation, and provides the unique capability to mediate access to host hardware, allowing hardware access control policies to be enforced in the face of total host compromise.

ABOUT THE AUTHOR

Niranjan Reddy – He is an Information security Evangelist and Expert with 9+ yrs. of professional experience and known for various activities and accolades. He is the founder & CTO of NetConclave Systems, an Information Security Consultancy, Trainings & Research firm. He has been closely associated with Pune Police-Indian Police and has supported them very closely in setting up a Hi-Tech Cyber Crime Investigations Lab and also assisted and solved numerous cyber-crime cases. He was awarded the prestigious Commendatory Certificate for successfully solving critical cyber-crime cases from the Commissioner of Police in the year 2010.He has been awarded 5 years in a row (2009-2013) the prestigious ECCouncil Circle of Excellence Award as the best Trainer for Certified Ethical Hacker (CEH) in South East Asia by ECCouncil,USA at the Hacker Halted Conference, Miami-Florida-America. He has trained over 500+ till date professionals in Ethical Hacking & Cyber Forensics worldwide. He is also a core member of Data Security Council of India (DSCI) a venture of NASSCOM. He has executed critical Vulnerability, Penetration Testing and Web Application projects in India and abroad. He has been forecasted in major newspapers like Times of India, Pune Mirror, DNA and Mid-Day and is a Security Advisor for expert opinions for Cyberthreats. He is the Security Advisor for TechMahindra and Accenture. He has his articles published in Hakin9 which is an International magazine on IT Security.