

ViSe: The Virtual Security Testbed

Michael Richmond
Reliable Software Group
Department of Computer Science
University of California, Santa Barbara
mrichmon@cs.ucsb.edu

28 June 2005

Abstract

Traditional means of testing Intrusion Detection Systems (IDSs) require the creation of isolated physical test networks (testbeds) using machines that must be individually configured for each test. This process becomes cumbersome and resource-intensive when malicious attacks, launched against the pre-configured systems, cause significant harm and require the reinstallation of software before testing may continue. Virtual testbeds can minimize many of these costs and greatly increase the testing efficiency while accurately replicating physical environments. ViSe, a virtual security testbed, is a unique solution to the problem of security testing. ViSe's broad base of installed operating systems (OSs) and vulnerable applications provides an environment where researchers can test real attacks against vulnerable systems in a reliable and efficient manner.

Keywords: Security Testing, Testbed, Exploits, Intrusion Detection, Virtual Machine.

1 Introduction

Intrusion Detection System (IDS) research faces many ongoing challenges. The ever-present drive to reduce false alarm warnings (false positives), coupled with the constantly evolving sophistication of the attacks these systems attempt to detect, create a scenario where researchers must constantly struggle to maintain the relevance of their systems through testing and refinement. In order to verify the effectiveness of an IDS, the system must be tested on real-world (random) traffic. However, the implementors generally have little control over the nature of the generated traffic, which makes the tests irreproducible and unquantifiable. A solution to these challenges must provide efficient means of testing while generating repeatable, quantifiable results. Over the years, IDS developers have experimented with different approaches to achieving this goal.

Traditional methods of testing IDSs require the creation of isolated physical test networks (testbeds) using machines that must be individually configured for each test. This process becomes cumbersome and resource-intensive when malicious attacks, launched against the pre-configured systems, cause significant harm and require the reinstallation of software before testing may continue. Additionally, physical testbeds are limited in terms of operating system (OS) flexibility unless a large hardware infrastructure is present. Despite significant research into extensible yet cost-effective means of IDS testing, no clear solution has emerged. Standardized traffic datasets such as the Lincoln Labs/MIT Dataset or the Defcon "Capture the Flag" traffic logs help mitigate some of these problems, but do not allow for testing flexibility nor reduce the costs associated with reconfiguring the necessary physical systems. Precompiled log tests contain many well known deficiencies such as instant datedness, inherent biases, dataset specific calibration, and lack of flexibility [32, 37, 39]. Thus, traffic datasets are an imperfect solution.

Other barriers to efficient utilization of testbeds include rebuild time, configuration time, and exploit-specific costs. Rebuilding a damaged OS from scratch is a time-consuming endeavor, but remains a necessary task whenever attack testing is performed on physical machines. The time and expertise required to locate, install, and configure vulnerable software on these OSs only adds to the overall costs of a testbed. For example, on a Pentium 4 1.2 GHz machine, it takes 35 minutes to install Windows XP, 5 minutes to configure a basic network connection, and another 10 minutes to install vulnerable server software and activate those services for a total of 50 minutes. Installing Fedora Core 3 and similar services on the same machine takes even longer and further detracts from a researcher's ability to focus on developing novel technologies. A final barrier to testbed creation lies in the procurement of exploit code and its corresponding vulnerable software. Although exploit code is regularly published on security-related websites [38, 10, 13, 23] it must often be modified to run on a particular OS or testbed. The search for the corresponding version of the often out-of-date software remains a time-consuming search, because most websites remove vulnerable software from distribution once flaws are found. Spending such a significant amount of effort to configure a system that could be destroyed within seconds during testing seems like a fruitless task.

Researchers sought to nullify many of the deficiencies of physical testbeds through the creation of virtual testbeds. Some of the most famous include Lariat [37], LLSIM [25], TIDeS [39], DETER [8, 17] and Netbed [44]. All contain their strengths and weaknesses, but none provide an ideal security testing environment. Virtual testbeds' relaxation of the traditional one-to-one requirement of physical machines to running operating systems increases testing flexibility while also reducing install time. Instead of reinstalling an OS from scratch, a clean system may be generated, then booted in a virtual machine. In some cases, network configurations may also be quickly modified through the editing of the virtual testbed's configuration files [27, 28, 44, 8]. Consequently, virtual machine testbeds streamline the testing process by providing an infrastructure for quick, reliable, and flexible means of replicating network scenarios.

We have developed the Virtual Security Testbed (ViSe) as a novel solution to the problem of efficiently performing security testing. ViSe's leveraging of VMware's snapshot system provides a rich environment where developers may rapidly generate traffic using configured attacks against a broad base of installed x86-based operating system images in a safe and reproducible manner. The snapshot system allows researchers to build and configure a variety of vulnerable systems incrementally. By doing this, it is possible to create a rich set of environments with different security postures. Real exploits, based on published proof-of-concept code, form a basis for ViSe testing. For all included exploits, the vulnerable software they attack was installed on one or more virtual machine images and set up to be vulnerable upon boot. These features reduce the time commitment necessary to create vulnerable operating system images for testing and allow the researchers to focus on developing novel technologies rather than rebuilding their test infrastructure.

This paper first discusses related work in Section 2, before introducing the reader to the VMware Workstation 5.0 platform in Section 3. Section 4 illustrates the testbed specification through a description of its networking configuration, OS images, exploit repository, and detection programs. The paper then ties these concepts together through an example use case in Section 5, demonstrating the benefits of ViSe. Section 6 briefly concludes and outlines future work.

2 Related Work

Three categories of related work must be described in order to put ViSe into the appropriate context. A discussion of Vigna, Robertson, and Balzarottis' Testing Network-based Intrusion Detection Signatures Using Mutant Exploits [41] paper provides a foundation for this work by inspiring the need to create a testbed that could handle both attack and IDS testing. Then existing environments are considered in order to show the benefits of VMware [21]. Finally, alternate testbeds are investigated for comparison purposes.

2.1 The Sploit Approach

During 2004, Vigna, Robertson and Balzarotti tested the quality of the leading open and closed-source Intrusion Detection Systems' signatures using a mutant exploit generation engine. The goal of their black-box testing was to determine deficiencies within the rule sets of the analyzed IDSs, without specific knowledge of the systems' rule sets. Their engine combined ten publicized exploits (all of which are reproduced in ViSe) with various modification techniques to generate exploit variations that could be used to attack a target operating system. Evasion techniques were applied at the network, application, or exploit layers (for example, IP fragmentation, HTTP evasion techniques, polymorphic shellcode, and alternate encodings).

Their testbed, while comprised of physical machines, served as a model for ViSe. An attack image running Redhat 9.0, an IDS image running Redhat 7.3, and various victim operating systems were linked on an isolated testbed during the experiments. ViSe uses updated versions of the Redhat systems, Fedora Core 3, and has expanded the range of victim operating systems particularly in terms of Windows-based images. Additionally, Snort v2.3.3 is used instead of Snort v2.1.2.

2.2 Existing Emulation Environments

In this next section, we describe leading operating system emulators and evaluate their relevance to ViSe. To be considered for this section, an emulator must provide an OS virtualization environment. Virtual network testbeds that were not designed explicitly for security testing also received consideration. That is, the approaches discussed here concentrate on the mechanisms needed to create a hardware abstraction layer (HAL) or network of HALs rather than leveraging a HAL to create a full-fledged security testbed.

Before a discussion of emulation environments can take place, some terminology common to the virtual machine community must be clarified. The underlying operating system running on a physical machine shall be referred to as the "Host" operating system. Operating systems running within virtual machines will be referred to as "Guest" images. These conventions are important to ensure accurate communication when discussing physical versus emulated environments.

2.2.1 VMware

VMware [21], a product of the VMware Corporation, leverages virtualization technology to emulate a wide variety of x86-based operating systems on x86-based physical hardware. It is unlike other emulation environments which trade performance to create a completely hardware independent virtual machine environment, or those that make a number of simplifying assumptions in order to achieve good performance figures. For a non-negligible drop in performance, VMware accurately replicates the operation of the system on physical hardware, sacrificing some performance for the assurance of fidelity. VMware out-performs fully virtualized emulators by allowing non-privileged instructions to execute directly on the physical hardware of the host system. By virtualizing only "unsafe" instructions or operations [21], VMware greatly increases the speed at which it may emulate guest operating systems, but restricts itself to x86-based physical hardware. This feature, combined with its support for most major x86-based operating systems and ease of use, made VMware an ideal candidate during the search for ViSe's emulation environment.

2.2.2 User-Mode Linux (UML)

Although limited in scope, User-Mode Linux [20] creates a virtual machine environment that is well suited for executing Linux programs in a sandbox, experimenting with configurations, and performing kernel development. Unlike VMware, UML cannot emulate non-Linux operating systems, which renders it useless for Windows or Unix-derivative OS tests. Unofficial ports of UML to Windows [31] and Sparc [29] have been published, but they are fragile and unmaintained. UML makes up for its lack of scope through efficiency. With a stated maximum performance slowdown

of 20%, UML provides an environment for “concurrent virtual machine execution” [20]. However, in the end, its lack of support for the Windows operating system ruled it out from consideration as a basis for ViSe.

2.2.3 Plex86

The Plex86 Virtual Machine Project [5] provides a Linux-based virtual machine to emulate Linux-based OSs. In order to minimize the mediation between the guest virtual machine and the host system, Plex86 does not virtualize Input/Output behavior. Instead it uses a Hardware Abstraction Layer (HAL) to mediate the information flow between the physical host and the virtual machine, thus limiting the constraints placed on the virtual machine. While Plex86’s lightweight nature implies that it can outperform VMware, its limited scope and lack of support for Windows or Unix-based systems precludes it from consideration in ViSe.

2.2.4 Bochs

Bochs is an open-source alternative closely related to VMware. Unlike UML, Plex86, or VMware, Bochs runs the entire guest operating system in a virtual environment, which allows it to run on most major architectures such as x86, Sparc, MIPS, and PowerPC. Bochs also emulates most major x86-based operating systems such as Windows (95,98,2000, and NT), all Linux distributions, and all BSD distributions [9]. In order to achieve hardware independent flexibility, Bochs sacrifices the speed of guest virtual machine execution. All guest operating system instructions must be handled virtually in order to translate them into the native instruction set of the host system.

2.2.5 VirtualPC

Microsoft’s VirtualPC [15] represents an alternative to VMware. It runs on a subset of Microsoft’s operating systems (Windows XP Professional and Windows 2000 Professional) and closely mimics VMware in terms of memory requirements, guest operating system variety, networking, and virtual disk versioning. Based on VMware Workstation 5.0 and VirtualPC 2004’s respective manuals, beyond VMware’s execution on Windows and Linux host OSs, it is difficult to tell them apart [26]. Conversely, the Apple Macintosh edition of VirtualPC allows Mac users to emulate x86-based Windows virtual machines. This is an important advantage that made VirtualPC a possible finalist.

2.2.6 vBet

In 2003, Jiang and Xu published a paper [27] describing a virtual machine capable of emulating various environments such as the Chord peer-to-peer network, general test networks, and an implementation of OSPF routing. vBet’s network topology specification scripts provide the flexibility and scalability that Jiang and Xu desired from their virtual testbed. vBet leverages UML to support up to 60 concurrent virtual machines running on a single Dell PowerEdge server [27]. Jiang and Xu most likely saw VMware’s ability to support only a few concurrent guest images as a serious drawback for testing virtual networks because it limited the number of nodes they could include in their networks. This decision is logical from their point of view because they would be more concerned with the interconnections between nodes, rather than what exactly happens within a node during a simulation. ViSe, on the other hand, requires exact knowledge of both what happens within a node and what communication takes place between nodes. Supporting only a few concurrent virtual machines, while not an ideal situation, is an acceptable trade-off for the flexibility that VMware provides for installing and configuring guest images.

2.2.7 HoneyNet Project VMware HoneyPots

ViSe is not the first security testbed to leverage the flexibility of VMware for security testing. The HoneyNet Project [18] routinely uses VMware images as honeypots. Honeypots are physical or virtual computers configured to allow and monitor attacks from remote hosts in order to passively study the mechanisms used in practice to attack systems [4].

Often, honeypots are used to study worm binaries and propagation means in order to glean information about the intentions of their creators. VMware images allow the HoneyNet Project to suspend and snapshot exploited guest images to preserve attack states for future forensic analysis.

2.2.8 Denali

The Denali [43] isolation kernel creates a virtual layer between the hardware and the internet services that run upon it to provide a virtual machine that can multiplex a large number of programs on a single machine. It maintains the security of the hardware through mediation of the access of untrusted or unsigned code. The authors acknowledge that Denali's architecture is similar to that of VMware, but it makes numerous changes to the underlying physical hardware to enhance its performance. In testing scenarios, the authors claim that Denali may handle up to 10,000 concurrent virtual machines on commodity hardware running their specially tailored Ilwaco operating system [43]. While these performance numbers may be impressive, Denali would not be practical for ViSe because its virtual layer omits many necessary features of the x86 architecture and cannot run guest OSs exactly as they perform on physical hardware.

2.2.9 Xen

Xen [24] provides a virtual environment to emulate guest operating systems with minimal performance degradation while maintaining all existing functionality. Unlike VMware, which supports full virtualization of the host hardware for a performance hit, Xen "paravirtualizes" the hardware by running guest OSs over specialized guest architecture. This provides superior performance, but may not perfectly mimic the behavior of a physical machine running that particular OS. For example, Xen's "paravirtualized" layer is capable of handling up to 100 virtual machines of varying operating systems with only a few percent slowdown per machine [24]. Unfortunately, one drawback to their system is that OSs must be specially ported to the virtual environment before they may be used in virtual machines. Just like ViSe, Xen attempted to cover the major categories of personal computer operating systems through ports of Windows XP, NetBSD, and Linux, but had not yet finished all the ports. Xen is superior to Denali, because it runs major OS distributions, albeit after porting, rather than a toy guest OS with minimal functionality.

2.2.10 PlanetLab

The PlanetLab project [16] is an ongoing experiment for the creation of a world-wide distributed system. In order to connect to its network services, institutions must configure local nodes to share. Consequently, PlanetLab nodes have spread throughout 25 countries to 160 separate institutions as of June 16th, 2004 [34]. PlanetLab introduces the concept of "distributed virtualization," where users are assigned "slices" that correspond to distributed groups of virtual machines upon which they may run services. The idea of creating a world-wide distributed security testbed is an interesting concept, but is beyond the scope of this paper. The lack of isolation in such a system alone would disqualify it from testing malicious worms or exploits in a traditional manner, because PlanetLab does not provide mechanisms to isolate the virtual machines that make up a slice.

2.2.11 ModelNet

ModelNet [40], represents another solution to the need for a scalable network emulation environment. Its logical layout is comprised of a router core connected to edge nodes. The core precomputes the shortest paths between edge nodes before the simulation begins, and then stores that information internally. All edge nodes are connected to the core and maintain a unique IP address. Although the ModelNet paper states, "Edge nodes can run any OS and IP network stack and may execute unmodified application binaries," [40] no explanation is given regarding the exact emulation environment or how such a remarkable feat is achieved. This precludes comparison to VMware or a possible ViSe setup. Looking beyond this drawback, ModelNet is stated to simulate thousands of edge nodes in Internet-like

conditions (such as peer-to-peer networks, ad-hoc networks, or overlay networks) and appears to be similar to vBet. As mentioned earlier, such simulation efficiency comes at the price of simulation accuracy, meaning that ModelNet relies on a number of abstractions and does not simulate actual network activity in detail, thus violating a requirement of ViSe.

2.3 Existing Testbeds

This section evaluates complete emulation environments that have been created for security testing. To qualify for consideration, a testbed must be primarily geared towards security testing, but need not run on a novel emulation environment. Despite this rather broad criteria, only five testbeds were deemed worthy of serious consideration.

2.3.1 LARIAT

The Lincoln Adaptable Real-time Information Assurance Testbed (LARIAT) [37], represents the first major attempt to create a comprehensive platform for testing Intrusion Detection Systems. Released in 2002, three years after the DARPA tests that it sought to improve upon, its design goals were to create a testbed that could evaluate IDSs in real-time while remaining extensible and user-friendly. LARIAT relies on virtual attack and traffic generators to simulate the network traffic of a typical military installation. The attack generator provides “attack scenarios” involving real multi-step attacks, a feature that ViSe lacks. Its reliance on physical victim machines significantly increases the cost of resetting the testbed in addition to decreasing the ease with which it may be reconfigured. However, it must be noted that, when LARIAT was proposed, virtual machine technology was not as advanced as it is today.

2.3.2 LLSIM

The Lincoln Laboratory Simulator (LLSIM) [25] is a fully virtualized descendant of LARIAT. Written in Java, it provides a highly-customizable environment capable of simulating hundreds of nodes on commodity hardware. In order to achieve scalability and faster-than-real-time network performance, LLSIM omits “complex interaction[s] between testbed components” thus avoiding, “detailed packet-level and flow-level simulation” [25]. This approach relaxes the constraints of a physical environment for performance purposes, but it does so at the cost of an average increase of alert rates with respect to the rates experienced when using physical hardware. In contrast, ViSe strives for exact replication of the network interaction between its simulated hosts. Consequently, it fails to scale to hundreds of hosts on commodity hardware.

2.3.3 TIDeS

TIDeS, the Testbed for evaluation of Intrusion DEtection Systems [39], seeks to quantify the evaluation process when determining the appropriate type of IDS to use given a specific network topology. As with LARIAT, its primary goal is to overcome the numerous published deficiencies in the 1998 and 1999 DARPA IDS tests [39, 37, 32]. The TIDeS architecture is similar to that of ViSe, through its use of Honeyd [35] virtual machines. These Unix daemons generate simulated traffic using up to 40 different attack engines and 6 different legitimate traffic scripts. In order to generate an approximation of a comprehensive series of tests, TIDeS evaluates many criteria such as false alarm rates, consistency, performance under stress, and the breadth and depth of its exploit knowledge.

The authors of the system are vague about the actual tests performed and state very little in terms of actual TIDeS results against a “well-known IDS” [39]. Therefore, comparison to ViSe can only be performed on a conceptual level. In the end, ViSe appears superior due to its emulation of the victim and defensive images because it provides greater flexibility and ease of use.

2.3.4 DETER and Netbed

The Cyber DEfense Technology Experimental Research testbed (DETER) [8, 17] is a partnership project between the National Science Foundation, Department of Homeland Security, USC, UC Berkeley, and McAfee Research. Its goal is to leverage cluster technology, specifically the University of Utah's Emulab [44], to provide a flexible testbed where security researchers may perform a variety of experiments in an isolated environment. Configuration scripts automate the installation of operating systems on DETER's generic nodes, in addition to fragmenting the testbed network through modification of the router and switch settings. Its direct support of x86-based operating systems (FreeBSD 4.7/4.9, Redhat Linux 7.3/9.0 and Windows XP) plus support for OSKit [3], provide the flexibility necessary to handle a variety of experiment classes. As of June 2005, research projects focused on studying worm propagation and means of fighting distributed denial of service attacks. These experiments reflect that fact that the size and scope of DETER is geared toward large-scale tests even though it appears to support small-scale tests as well.

When comparing DETER with ViSe, DETER's support for large numbers of interconnected nodes makes it a clear favorite for large-scale tests. It provides the scalability that ViSe lacks while still maintaining a secure test environment. One drawback of DETER is its limited official support for the OSs running on its nodes. Despite DETER's use of OSKit to port new OSs to the testbed, ViSe's incorporation of those operating systems allows for the testing of a wider range of vulnerable software.

Netbed [44], a predecessor of DETER, also uses the Emulab emulation infrastructure. It successfully integrates geographically distributed machines into a logical simulation network while maintaining the realism of the experiment. As with the other testbeds, Netbed provides similar savings in terms of reconfiguration and automation through its use of `ns` (Network Simulator) scripts. Netbed's isolation from outside networks through the use of FreeBSD's Jail mechanism and network tunneling provide an adequate environment to test malicious network software. The increased prevalence of worms and malware proved the need for a dedicated isolated security testbed, thus inspiring the creation of DETER.

2.3.5 vGrounds

vGrounds [28] represents an extension of Jiang and Xu's vBet to large-scale security testing. Just like vBet, vGrounds is a UML-based virtualization "playground" for network simulation that retains similar performance numbers ("one physical host can support several hundred VMs" [28]), but differs in two important ways. vGrounds was designed to support malicious worm testing, so it features full isolation of the virtual testbed to protect outside networks from the worm behavior. In addition, vGrounds also runs on existing large-scale testbeds such as PlanetLab, making it easy for researchers to share and deploy.

vGrounds and ViSe share similar design goals, yet they represent different philosophies of security research. vGrounds is geared toward the study of worm attack and propagation, while ViSe is geared toward the study of defense against attacks. vGrounds surpasses ViSe in terms of scalability, isolation, and automation, but ViSe's support for Windows, Linux, Solaris, and BSD operating systems, along with its large exploit collection and set of configured vulnerable images remain a unique contribution to the security community.

3 VMware Workstation 5.0

VMware [21], a commercial manufacturer of virtualization products since 1998, sells a variety of products that emulate x86-based operating systems. VMware Workstation, the product designed for desktop use, supports virtual machines for all Windows distributions including experimental support for Longhorn, in addition to official or unofficial support for most Linux and Unix distributions. This makes Workstation 5.0 a very flexible emulation environment suited to fulfill ViSe's requirements. Another important feature is the ability to easily setup and configure emulated networks between virtual images. Thus, users may quickly combine their emulated OS images in various configurations for testing purposes.

Even though VMware's products are closed-source, they represent one of the most flexible and well-supported means of leveraging virtual machine environments for security testing. One obvious drawback of their closed-source approach is that it prevents users from modifying the products to best suit their needs. However, this must be balanced against the fact that VMware accurately emulates all aspects of the operating system including exact memory addressing.

The two most important classes of VMware's image files are configuration files and virtual disk files. Configuration files (ie., *.vmx files) contain the basic information needed to run a guest image such as the location of the virtual disk and setting specifications. Virtual disk files (ie., *.vmdk files) simulate hard disk devices. Every virtual machine must be linked to at least one virtual disk in order to function, but virtual disks do not necessarily have to map to virtual machines on a one-to-one basis. As long as a virtual disk file is marked read-only, any number of virtual machines can use it as a basis for their virtual disks. Other important files are the numbered snapshot and corresponding virtual disk difference files. These file pairs contain the difference information between a particular snapshot and its predecessor in the snapshot tree. More information on the snapshot mechanism and the snapshot manager is given in Section 3.2.

3.1 Benefits over other emulation environments

As mentioned earlier, the use of VMware Workstation 5.0 (Workstation 5.0) provides many benefits to ViSe that other emulation environments cannot provide, such as its support for most x86-based operating systems, detailed emulation of the guest OS, and built-in support for tracking the differences between versions of guest operating systems.

VMware support for multiple operating systems extends beyond its official list of supported guest images to include almost all x86-based operating systems (for a complete and up to date list please go to the company's website) [21]. Workstation 5.0 runs over modern Windows systems and most Linux distributions, making it a highly portable environment. Evaluation versions of the software, available for 30-day trials, facilitate the sharing of configured guest images between researchers. For example, ViSe's Fedora Core 3 Attacker image can quickly be packaged and transmitted to researchers around the world for the replication of the testing experiments. As long as other researchers are not opposed to using closed-source software for testing purposes, these image exchanges will enhance their ability to reproduce tests that others have performed in a controlled manner.

In comparison tests of Windows XP Pro, Fedora Core 3, and FreeBSD 4.5, the operating systems functioned similarly when run on physical hardware and virtual hardware. Stack addresses were used as the primary means of comparison because exact replication of stack addressing is critical for many buffer overflow exploits to work. Provided that acceleration was disabled [12] on the virtual machines, all invariant stack addresses lined up exactly. During testing, variable stack addresses were located in similar locations on the physical and virtual machines, but did not line up exactly. For example, a remote buffer overflow on a FreeBSD system running the Apache Web Server [] targetted a buffer located at 0x080f9930 on the virtual machine, while the attack targetted the same buffer at 0x080f3930 on the physical machine. This variance was most likely due to slightly different environment configurations between the two installations of the OSs. In the end, disabling acceleration results in decreased performance, but it is necessary in order to ensure the exact replication of an operating system. While VMware does not represent a perfect solution to the problem of accurately and efficiently emulating a variety of operating systems, its solution satisfied the requirements of the ViSe environment.

VMware's primary contribution to the usefulness of ViSe environment lies in its ability to gracefully handle numerous versions of a guest image simultaneously. Workstation 5.0 provides a built-in interface to quickly switch between versions of a configured guest image or package specific versions to run as independent images. For example, Workstation 5.0's flexibility allows a researcher to perform security tests on Windows XP Pro Service Pack 2, then perform the same tests on versions of XP with Service Pack 1 during the time span of a few minutes (including guest OS boot time). If the tests destroy that particular version of a guest image, a fresh image may be created and booted within seconds. Similar tests on physical hardware would require the re-installation of the targeted operating system, slowing the testing process down by a few hours for every necessary re-installation. For example, it takes 45 minutes to install

Windows XP, but 40 seconds to create and boot a new image. Workstation 5.0's snapshot feature and corresponding snapshot manager allow for such fluid transitions between guest operating systems.

3.2 The Snapshot and Snapshot Manager

VMware's snapshot mechanism is a powerful tool for building OS images incrementally "A snapshot captures the entire state of the virtual machine at the time you take a snapshot. This includes: Memory State – The contents of the virtual machine's memory, Settings State – The virtual machine settings, [and] Disk State – The state of all the virtual machine's virtual disks" [22]. The saving of memory state, including the exact contents of the virtual processor's registers, is crucial to the usefulness of the snapshot mechanism. Snapshots taken of booted guest images may be quickly restored to their booted state by reverting to the snapshot. It would be possible to create a series of snapshots of booted guest images to save boot time, but doing so could lead to undesired side effects. For example, VMware warns about saving a snapshot while established network connections are transferring data. Upon reverting to the snapshot, the image will assume that it is in the middle of a network connection and resume data transmission, leading to inconsistencies [42]. For this reason, all ViSe snapshots were taken while the guest OSs were in the powered off state.

The snapshot manager increases the usefulness of the snapshot system by displaying its hierarchical format graphically in a tree. Researchers can visually navigate through the snapshot tree to choose the specific image they desire, then instantiate a version of that image by spurring a branch in the tree through the "Go To" button. As snapshots are taken, the manager automatically stores the snapshot data to disk and maintains the links between the data files. Theoretical support for over 100 snapshots per branch exists, but was not tested during ViSe development [42]. See Figure 1 for a generic example of a snapshot tree. In ViSe, roots are typically the baseline (initial) installation of an OS on a virtual machine. Images farther up the tree (snapshots taken after the root), depend loosely on the intermediary images between them and the root. While they may not be moved within the tree, images may be deleted without problem, because Workstation 5.0's snapshot manager will rebuild the tree, provided that no linked-clones were branched after the deleted node in the tree. In order to protect against accidental deletion or the destruction of linked-clones, Template Mode can be activated to prevent the deletion of snapshots.

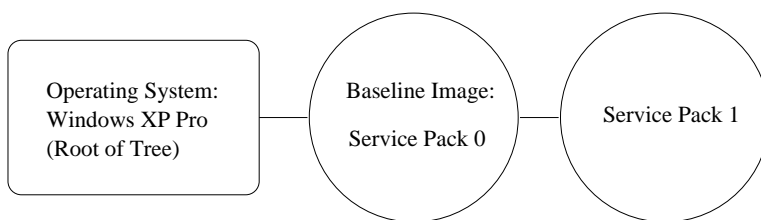


Figure 1: Example Snapshot Tree

Together, snapshots and the snapshot manager provide ViSe users with a powerful tool to manage versions of different operating systems in an efficient manner. In particular, snapshots conserve storage space by only saving the file differences between the current snapshot and its parent snapshot or virtual disk. Consequently, it is possible to create a maximal install of an operating system and snapshot the image (a snapshot that requires significant storage space), then add a vulnerable software package and save another snapshot of the image (a snapshot that requires minimal storage space). The end result is two versions of a maximal install that take up a little more space than that original maximal install. Traditional means of creating two similar versions of an OS install would require twice as much storage space as needed for the maximal install plus additional space for the vulnerable software. For example, a maximal install of Fedora Core 3 requires nearly 7 GB of disk space, yet 18 separate Fedora Core 3 images exist on ViSe taking up only 12.2 GB of space, saving 113.8 GB of storage space. A maximal install of Windows XP

Professional requires 4.1 GB of storage, but ViSe's 8 XP images only require 6.3 GB of storage.

3.3 Clones

In order to increase the portability of Workstation 5.0 virtual machines, VMware implements two separate cloning mechanisms for its virtual machines: linked cloning and full cloning. Linked clones, are very similar to branches of a snapshot tree and depend on the virtual disk of the virtual machine from which they were cloned. They are logical extensions of that virtual disk, but become semi-autonomous after cloning. Linked clones make it possible to mount a baseline virtual machine on a network drive, and then, to allow linked-clones residing on separate workstations to access the same image [42]. In a corporate setting, this use of linked clones gives a network administrator the power to configure a virtual machine as desired before posting it to the network and linking all employee workstations to that baseline image [42]. In terms of ViSe, a similar scenario could arise where the main ViSe images are accessible on a local subnet and researchers link their personal versions of a guest image to the central ViSe repository.

Full clones, on the other hand, are completely independent entities. All virtual disks, configuration files, and settings are replicated in the new virtual machine with a few exceptions: the Universally Unique Identifier (UUID) and MAC address of the virtual machine are modified in order to allow a cloned image to function at the same time as the virtual machine from which it was cloned. Workstation 5.0 provides the ability to generate clones from anywhere on a snapshot tree, so full or linked clones may be obtained from any of the configured versions of a guest OS. Thus, full cloning provides a powerful mechanism to distribute prepared virtual machines amongst researchers. Attacker, detector, or victim images may be full-cloned then transferred in order to share specific OS configurations.

3.4 Networking

Workstation 5.0 provides three network configuration options for virtual machines. At a high level, the options allow for emulated access to an existing network's simulation of a virtual network within the host, or connection to the host operating system. VMware names these options bridged networking, network address translation (NAT), and host-only networking, respectively.

In bridged networking, Workstation 5.0 creates a virtual switch, connects it to a virtual machine, and then bridges a connection to the physical ethernet card of the host. Under bridged networking, the guest image masquerades as a normal network host using a static or DHCP-assigned IP address. It operates as if it was physically connected to the network, but transmits traffic through the network interface card (NIC) of the host machine.

In NAT networking, Workstation 5.0 emulates a NAT router between the physical NIC and the virtual switch. It also emulates a DHCP server to manage internal IP addressing. This mechanism performs similarly to a physical NAT system.

Finally, Workstation 5.0 allows host-only networking through the emulation of a completely virtual network. It assigns virtual NICs to the host and guest images, connects them to a virtual switch, then initializes a virtual DHCP server to manage addressing. Since guests have no direct connection to an outside network (if one exists), this contains the testbed within the local subnet.

It is important to note that there are limitations to the extensibility of Workstation 5.0's virtual network. While Windows hosts may connect an unlimited number of guests to a single virtual switch, Linux hosts are limited to 32 simultaneous guest connections [42]. At the moment, this limitation does not affect ViSe, because it only runs three virtual machines at a time. This could hamper future use unless VMware solves this scalability problem.

4 ViSe

Although ViSe contains 10 versions of popular operating systems, and 40 exploits against them or programs that run on them, it should not be viewed as simply a collection of configured virtual machines and exploits. To do so

would overlook ViSe’s intended purpose as a security testbed. From a security testing standpoint ViSe contains three classes of configured guest images: attacker, a detector, and victim. The current implementation of ViSe includes a single Attacker image, which is a specially configured Fedora Core 3 image that contains the exploit code and detailed instructions for the 40 included exploits. The two basic detector images included in the testbed are a Windows XP Pro image and a other on Fedora Core 3 image. Both images include the default Snort 2.3.3 [36] install without rule set updates. As an added feature, the Fedora Core 3 detector also includes the SyscallAnomaly host-based IDS. The current implementation contains a number of victim images, a few of which are configured for immediate compromise using specified exploits. The current implementation should be viewed as a proof-of-concept for ViSe, because its flexible architecture is able to handle the expansion of the basic classes as well as incorporation of new image classes.

In all cases, guest OS installations are rooted at the basic install level with minimal modifications. One of the first branches of every snapshot tree is called “Maximal,” meaning that it contains close to the entire package set provided by the base OS distribution. See Figure 2 for an example configuration of the testbed.

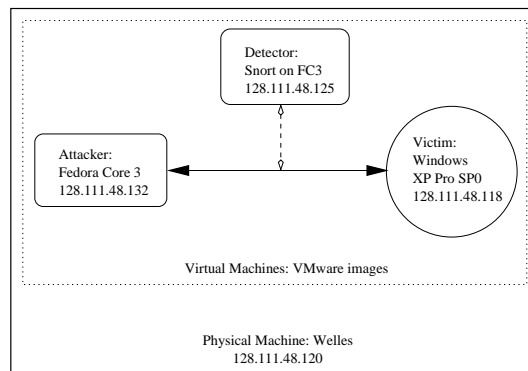


Figure 2: Example Testbed Setup

4.1 VMware Networking Decisions

As mentioned earlier, Workstation 5.0 includes three options for networking guest images. Of the three, bridge and host-only networking received the closest consideration. Host only networking’s greatest asset lies in its isolation of the testbed from outside traffic. Conversely, it also protects the physical network from whatever malicious activity occurs on the virtual network. Overall, host-only networking provides a pristine environment for IDS testing, but limits the testing to only what may be done on the host machine.

Bridged networking, on the other hand, allows direct access to the images from the local subnet. It increases ViSe’s flexibility for security researchers, because they may use whatever technology they desire to connect to the guest images. For example, if an IDS developer already maintains a Mucus [33] or Snot [1] stimulator, they can obtain ViSe guest images and immediately begin testing, rather than configuring their own victim images. Unfortunately, bridged networking also allows attacked guests to connect to outside hosts. During simple attack testing, this did not represent a problem, but it might be a problem during experiments with worms. In the end, the decision was made to favor flexibility over isolation. Therefore, ViSe uses bridged networking. As a compromise, the statically configured guest network interfaces do not contain paths to the local gateway, which limits (but does not guarantee) the possibility of undesired traffic leakage to outside networks.

4.2 Operating System Images

By virtue of Workstation 5.0's ability to support most x86-based operating systems, ViSe includes Windows, Linux, and BSD images. One notable exception is Sun's Solaris operating system. This was not done due to the lack of the time needed to configure a guest image and obtain and port exploit code. Whenever possible, the maximal package set was included early in the snapshot tree to maximize the vulnerable potential of every operating system. In the case of the Windows guests, this practice streamlined the process of setting up attack-specific vulnerable images. For the Linux/BSD images the same benefits were not realized, because the most popular vulnerable packages are distributed separately from the operating system.

4.2.1 Windows

Three Windows OSs were chosen due to their applicability to security testing and their numerous vulnerabilities. Early versions of the OSs (without service packs) contain numerous published security vulnerabilities and exploit code was obtained without significant effort. Later versions proved to be more robust, making it more difficult to obtain exploit code. In fact, at the time of publication, no Windows XP Service Pack 2 or Windows Server 2003 Service Pack 1 exploit code exists on ViSe. This does not mean that such code has not been published on the internet, only that it has not been integrated into ViSe. See Figure 3 for a diagram of the current state of the Windows snapshot trees. Due to licensing constraints, it is illegal to distribute images to the public so they will be used solely for testing in the Reliable Software Lab. Interested parties may create their own images following the design principles outlined in this paper.

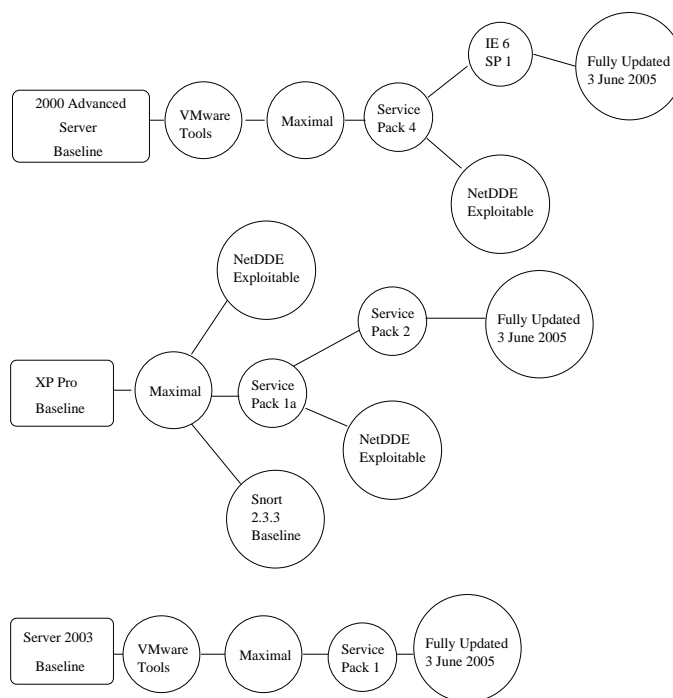


Figure 3: Windows Snapshot Tree

4.2.2 Linux

The ease of obtaining Linux distributions (they are readily available through official homepages or mirror sites) allowed for making ViSe a Linux-rich environment. The only major distribution excluded from the testbed was Mandrake, but that does not take away from the importance of the testbed as a whole, because attacks against Mandrake may be replicated against other distributions. As in the Windows guest image set, maximal package installs were performed whenever possible. Most testing-related actions may be performed from the command-line, or performed more efficiently that way. Therefore, X-Windows was either disabled on startup or not installed on some images to expedite the boot process, except when required for an exploit. Another benefit from the efficiency standpoint is that non-X-windows guests do not require VMware Tools, which is an add-on package to provide enhanced graphics support and time synchronization between the host and a guest image [42]. See Figures 4 and 5 for a diagram of the current state of the Linux snapshot trees.

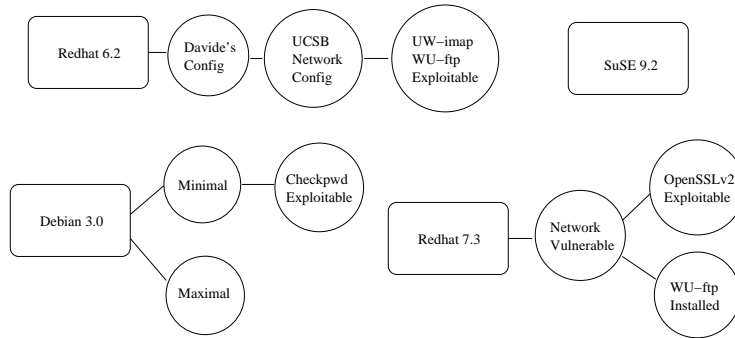


Figure 4: Linux Snapshot Tree

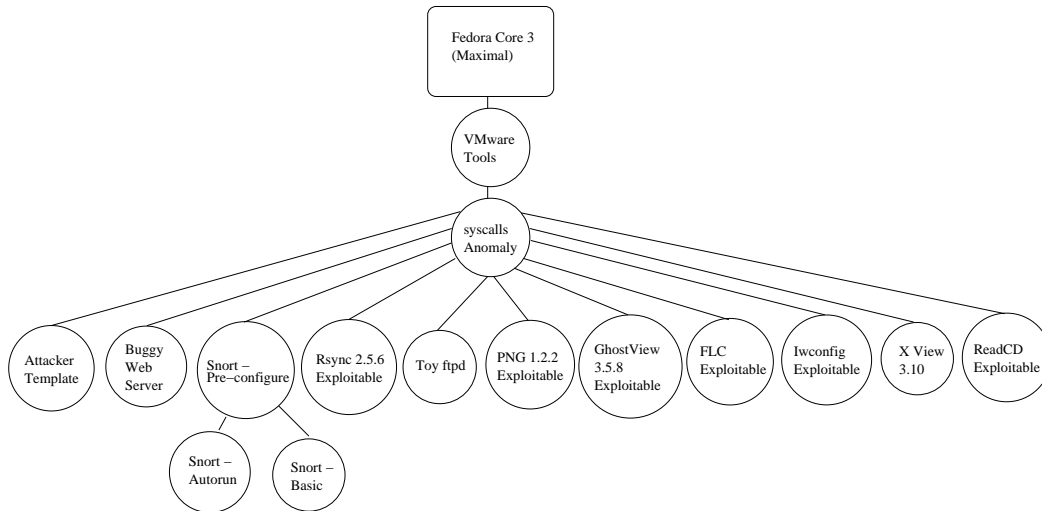


Figure 5: Linux Snapshot Tree

4.2.3 BSD

In order to provide support for Unix distributions while also proving their ability to function as part of ViSe, two versions of FreeBSD have been configured as victim guests. As with some of the Linux images, the X-Windows system is disabled by default so that the images boot directly to the command-line. The VMware Tools package, while provided for FreeBSD as an officially supported OS, was not installed due to a lack of the X-Windows system [42]. See Figure 6 for a diagram of the current state of the BSD snapshot trees.

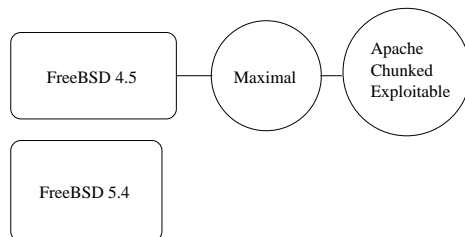


Figure 6: BSD Snapshot Tree

4.3 ViSe Exploits

Early in the evolution of ViSe, a question arose regarding the best means of incorporating exploit code into the system. The easiest method, from a development standpoint, was to bind exploit code to its original environment. For example, if an exploit was written in C using Windows socket programming, the exploit would be placed on the Windows Attacker. If the same exploit was written in C using Linux socket programming, it would be placed in the Linux Attacker. Thus, there would be one attacker image for Windows, one for Linux, and other images as future need arose.

The flaws to this system are readily apparent. One major flaw is that exploits could be duplicated on separate attacker images, if different versions of it exist, leading to wasted effort and possible confusion. Another is that switching between separate attacker images would waste user time when performing different attacks. In order to avoid these flaws, ViSe uses a centralized approach where all exploits reside on the same image. While this makes the ViSe Attacker much easier to use during testing, it requires more effort to modify the exploits so that they compile and run on the Attacker's Linux system.

The official Attacker image runs on a specially configured branch of the Fedora Core 3 guest image tree. It contains ported versions of all exploit source code that works against ViSe images, in addition to detailed execution instructions and links to more information on each exploit. Also, the Attacker image contains many standard security tools, such as nmap, netcat, and ethereal. At publishing time, the exploits were divided into directories for each class of operating system and sub-divided by exploit name. Each exploit directory contains the original source code, ported source code if changes were necessary, a compiled executable named `exploit` (if appropriate), the vulnerable software (if necessary), and a README file. The README file contains, at a minimum, one or more links to detailed information on the exploit, compilation instructions, execution instructions, and a table detailing which victim guest images it will successfully attack.

4.3.1 Ethical Implications

The ethical implications of distributing VMware images with compiled exploits and detailed instructions on how to use them should not be ignored. The most obvious concern regarding the distribution of attack images is that they make it even easier for a "script kiddie" to execute attacks against any of the included guest operating systems. It will no longer be necessary for them to even know how to compile a program because the binaries and execution

instructions are included in the Attacker image. Theoretically, someone with this attack image could obtain a free VMware evaluation license, download and install VMware Workstation 5.0, boot the attack image, and easily launch attacks against arbitrary hosts on the Internet in under an hour.

This is not to say that publishing the attack image should be regarded as unethical or even illegal. Almost all the exploits included in the attack image are freely distributed on the Internet and can be obtained from well known websites that publish exploit code [38, 10, 13, 23]. Those exploits that cannot be found on the Internet were developed to attack teaching software that Vigna developed for his Network Security and Intrusion Detection class [6]. A determined and knowledgeable attacker could replicate the same actions given enough time. The main dilemma involves the elevated ease of use of the attack image exploits. It is an unreasonable expectation to believe that all vulnerable systems affected by these exploits have already been patched, but that should not preclude the RSG from publishing these images. In the end, the benefits of attack images for security researchers outweigh the potential that they could be used for malicious purposes.

4.3.2 Importance of Prepared Exploits

Gathering exploit code and vulnerable software then porting it to a specific operating system or testbed is a time-consuming process that represents a significant challenge, which security researchers must overcome in order to test their products. Additionally, time must be spent configuring exploitable hosts with vulnerable software and then verifying that the attacks perform as expected. For these reasons, developing a well-rounded set of exploits for testing purposes represents a considerable time investment. To increase the usefulness of ViSe and mitigate this cost, a variety of exploits come in binary form on the attack image, in addition to a selection of guest OS images that are configured to be vulnerable to specific attacks. While the majority of ViSe exploits are buffer overflows, the set of available exploits also includes format string attacks, environment manipulations, directory traversals, and Denial of Service (DOS) attacks for completeness. At least ten of the included attacks have been used in previously published IDS testing papers and can be used for comparison testing [41, 39, 7].

4.3.3 Windows Exploits

ViSe includes a variety of Windows attacks covering many exploit classes. Buffer overflows comprise the majority of exploits, with Denial of Service attacks, directory traversals, and remote command execution exploits included for completeness. Please see Appendix A for a complete list of included exploits.

4.3.4 Linux Exploits

ViSe contains two classes of Linux exploits: exploits against the distributions and associated packages, and exploits against the Buggy Web Server (BWS) [2]. The Buggy Web Server (BWS), released under GNU's GPL by Giovanni Vigna in 2002 as a security teaching tool, remains relevant three years later. Its numerous vulnerabilities demonstrate many classic coding mistakes and provide interested parties with a safe and controlled environment where they may identify and investigate exploit and defense techniques. ViSe contains 14 scripted exploits of BWS vulnerabilities that were drawn from one of Vigna's Fall 2004 Network Security and Intrusion Detection homework assignments. Please see Appendix C for a complete list of included exploits.

Of the non-Buggy Web Server exploits, almost all of them are buffer overflows, but they are against a wide variety of software packages. The Buggy Web Server exploits cover a wide range of vulnerabilities and demonstrate a variety of possible attacks against the Linux OSs. Please see Appendix B for a complete list of included exploits.

4.3.5 BSD Exploit

At publication time, ViSe contains a single BSD exploit against the Apache web server. Please see Table D for the included exploit.

4.4 Intrusion Detection Systems

4.4.1 Network-based Intrusion Detection System

Licensing concerns are an important consideration when distributing digital products, so our choices were limited when deciding what IDS to distribute with ViSe. Amongst open-source network-based Intrusion Detection Systems (NIDSs), Snort was a clear choice due to its large install base. Originally written in 1998 by Martin Roesch, Snort has evolved into one of the premier Intrusion Detection/Prevention Systems [36] in the world and remains under the GNU GPL.

ViSe includes Snort v2.3.3 installed on Fedora Core 3 and Windows XP images. Minimal configuration choices were made during the install in order to provide as close to a pristine Snort image as possible. This image should be considered the root of a Snort tree with future branching used to save various Snort configurations for comparison testing.

Workstation 5.0's bridged networking allows the Snort images to function as typical NIDS sensors on the testbed's subnet. Subsequently, the Snort images can monitor traffic outside the testbed, as well as traffic within the testbed. A typical ViSe setup includes the detector image running in NIDS mode and monitoring traffic to and from a victim guest image (all of which have been configured to use the same IP address).

4.4.2 Host-based Intrusion Detection System

SyscallAnomaly, an anomaly-based Host IDS (anomaly-based HIDS), is included in ViSe as another means of monitoring the live attacks generated during testing. Published in 2003, SyscallAnomaly strove to minimize the quantity of false positive warnings produced by anomaly-based HIDS while still maintaining a very high detection rate through an analysis of system call arguments [30].

The specially-configured vulnerable Fedora Core 3 images, the attacker image, and the Fedora Core 3 HIDS image all contain a ported version of SyscallAnomaly, but it is deactivated by default in order to avoid unnecessary bloating of the unshrinkable virtual disks. A startup script, `daemonControl.sh`, located in `/home/testuser/syscalls/`, may be used to start, stop, and restart the module. By default, results are written to `/home/testuser/syscalls/var` and error messages are written to `/var/log/messages`.

5 Example Use Case

ViSe represents a flexible framework built upon the most popular x86-based operating systems. It is easy to use and allows for quick setup or reconfiguration of a security testbed. In order to demonstrate these principles, a use case shall illustrate Snort's ability to detect the Washington University FTP Server remote buffer overflow attack against a Redhat 6.2 machine, when run on ViSe. Three ViSe images are needed for this test:

- The "Attacker" image at 128.111.48.132
- The Fedora Core 3 Snort image at 128.111.48.125
- A branch of the Redhat 6.2 "WU-ftp Exploitable" image at 128.111.48.118

Steps:

1. Boot all three images.
2. Configure the Snort Image to sniff traffic to and from 128.111.48.118:
 - (a) Login as root (password = BR@K\$fst)

- (b) `cd /home/testuser/snort`
 - (c) execute: `snort -d -l ./var -c ./etc/snort.conf`
3. Launch the attack from the Attacker:
- (a) Login as testuser (password = canubrak)
 - (b) `cd $HOME/exploits/Linux/Sploit/wuftp`
 - (c) `./exploit 128.111.48.118`
 - (d) If the attack results in a Segmentation Fault, restart the FTP service on the victim image. Login as root (password = dyhttl) and run: `/etc/rc.d/init.d/inet restart`
 - (e) Else type `id` to verify that you have an Apache shell
4. Verify Snort posted alerts regarding the attack (on the Snort image):
- (a) Press CTRL-C to stop the Snort daemon.
 - (b) `cd ./var`
 - (c) `ls -l`
 - (d) `vim alert`

The specified attack scenario should result in a reverse-connect shell to the victim image at 128.111.48.118. It should also generate 2 Snort alerts based on inclusion of all default rule sets during operation. To launch a different attack against a different victim image given the same setup, a few changes need to take place. In the Attacker image, the user must `cd` to the appropriate attack directory and in the Snort image, Snort must be restarted. The existing victim image must be “powered off,” and a new one booted then logged into. Once these steps have taken place, the testbed will be ready for another test. With practice, these manual steps can be performed in a matter of seconds when excluding boot time. Boot time is excluded because it ranges from 35 seconds for Windows XP Pro, to three minutes and 21 seconds for Fedora Core 3.

6 Conclusions and Future Work

Security testbeds built entirely out of physical hardware remain in use, but are quickly being replaced with more efficient and cost-effective virtual testbeds that replicate similar environments. Throughout the last few years, researchers have created many virtual machine and virtual testbeds including LLSIM, TIDeS, DETER, and vGrounds. All have their strengths and weaknesses when compared with ViSe, yet none of them take ViSe’s unique approach. ViSe provides an environment where existing and future exploits may be tested against the entire range of x86-based operating systems under controlled conditions while monitored by network-based and host-based IDSs. Actual exploit code can be tested against the corresponding vulnerable software and operating systems “in the wild.” ViSe provides researchers with access to configured vulnerable systems, which saves them the trouble of creating those systems themselves. Thus, researchers can focus on their projects, and not on the re-creation of testbeds.

ViSe images are freely distributed and available for download from <http://www.cs.ucsb.edu/~rsg/ViSe>. As mentioned earlier, three classes of images are available for distribution: attacker, detector, and victims. The ViSe architecture allows them to be combined in a variety of ways, to add flexibility to possible test configurations.

During ViSe’s design and construction process, many ideas for improvement were suggested, but could not be incorporated due to time constraints. Future versions of ViSe should expand upon the existing infrastructure by including:

- Support for automated testing mechanisms such as testbed configuration scripts that leverage Workstation 5.0's ability to handle command-line control.
- Integration of an Attack Oracle similar to the one used in [41] to determine the success or failure of ViSe exploits.
- Expansion of the breadth and depth of ViSe victim images to include Solaris, OpenBSD, Windows 98, and more Linux distributions.
- Inclusion of more open-source IDSs to enable comparison testing.
- A formal database of exploits indexed by Bugtraq Identification Number or another unique identifier to enable efficient access and management of ViSe exploits.

Finally, the testbed could be integrated with exploit frameworks such as CANVAS [14], Core Impact [11], and Metasploit [19] to allow for variations of attacks to be easily tested.

7 Acknowledgments

Without the advice of Giovanni Vigna and Richard Kemmerer, this project would not have been possible. Many thanks are due to them and to the student members of the Reliable Software Group at the University of California, Santa Barbara. Particularly, Darren Mutz must be thanked for his patience while porting the syscallAnomaly software to ViSe and for his general assistance.

References

- [1] Snot. <http://www.sec33.com/sniph/>, 2001.
- [2] The Buggy Web Server. <http://www.cs.ucsb.edu/~vigna/buggy>, 2002.
- [3] The OSKit Project. <http://www.cs.utah.edu/flux/oskit/>, 2002.
- [4] Know Your Enemy: Learning with VMware: Building Virtual Honeynets with VMware. <http://www.honeynet.org/papers/vmware/>, 2003.
- [5] The Plex86 Virtual Machine Project. <http://plex86.sourceforge.net/>, 2003.
- [6] CS279: Network Security and Intrusion Detection, Homework 5. Internal Reliable Software Group Document, 2004.
- [7] Neophasis OSEC Project. <http://osec.neophasis.com>, 2004.
- [8] The DETER Testbed: Overview. <http://www.isi.edu/deter/docs/testbed.overview.htm>, 2004.
- [9] Bochs: The Open Source IA-32 Emulation Project. <http://bochs.sourceforge.net/>, 2005.
- [10] Bugtraq Vulnerability Database. <http://www.securityfocus.com/vulnerabilities>, 2005.
- [11] Core Security Technologies CORE IMPACT. <http://www.coresecurity.com/products/coreimpact/index.php>, 2005.

- [12] Differences between OS operation on physical hardware versus VMware. <http://www.vmware.com/community/thread.jspa?forumID=19&threadID=15557&messageID=171460#171460>, 2005.
- [13] FrSIRT Exploits and 0-day Exploits. <http://www.frstirt.com/exploits/>, 2005.
- [14] Immunity Canvas Professional. <http://www.immunitysec.com/products-canvas.shtml>, 2005.
- [15] Microsoft Virtual PC 2004. <http://www.microsoft.com/windows/virtualpc/default.mspx>, 2005.
- [16] PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services. <http://www.planet-lab.org/>, 2005.
- [17] The DETER Testbed: Technical Information. <http://www.isi.deterlab.net/index.php3>, 2005.
- [18] The HoneyNet Project. <http://www.honeynet.org>, 2005.
- [19] The Metasploit Project. <http://www.metasploit.com/>, 2005.
- [20] User-Mode Linux. <http://user-mode-linux.sourceforge.net/index.html>, 2005.
- [21] VMware. <http://www.vmware.com>, 2005.
- [22] Workstation 5.0: What is Captured in a Snapshot. http://www.vmware.com/support/ws5/doc/ws_preserve_sshot_whats_captured.htm, 2005.
- [23] A³. <http://www.addict3d.org/index.php?page=main>, 2005.
- [24] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the Art of Virtualization. In *Proceedings of the ACM Symposium on Operating Systems Principles*, New York, New York, Oct 2003.
- [25] Joshua Haines, Stephen Goulet, Robert Durst, and Terrance Champion. LLSIM: Network Simulation for Correlation and Response Testing. In *Proceedings of the 2003 IEEE Workshop on Information Assurance*, West Point, New York, June 2003.
- [26] Jerry Honeycutt. *Microsoft Virtual PC Technical Overview*. Microsoft Corporation, <http://www.microsoft.com/windows/virtualpc/evaluation/techoverview.mspx>, Nov 2003.
- [27] Xuxian Jiang and Dongyan Xu. vBet: a VM-Based Emulation Testbed. In *ACM Special Interest Group on Communications*, Karlsruhe, Germany, Aug 2003.
- [28] Xuxian Jiang, Dongyan Xu, Helen Wang, and Eugene Spafford. Virtual Playgrounds For Worm Behavior Investigation. In *Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection*, Seattle, Wa, Sept 2005.
- [29] Shant Katta. Porting User-Mode Linux to Ultrasparc Architecture. <http://www.csee.wvu.edu/katta/uml/>.
- [30] C. Kruegel, D. Mutz, F. Valeur, and G. Vigna. On the Detection of Anomalous System Call Arguments. In *Proceedings of the 2003 European Symposium on Research in Computer Security*, Gjøvik, Norway, October 2003.
- [31] Chandan Kudige. User-Mode Linux for Win32. <http://umlwin32.sourceforge.net/index.html>, 2002.

- [32] John McHugh. Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. *ACM Transactions on Information and Systems Security*, 3(4), November 2003.
- [33] Darren Mutz, Giovanni Vigna, and Richard Kemmerer. An Experience Developing an IDS Stimulator for the Black-Box Testing of Network Intrusion Detection Systems. In *Proceedings of the 2003 Annual Computer Security Applications Conference*, Las Vegas, NV, Dec 2003.
- [34] Larry Peterson and Timothy Roscoe. *The Design Principles of PlanetLab*. Princeton University and Intel Research Berkeley, Jun 2004.
- [35] Neils Provos. Honeyd: A Virtual Honeynet Daemon (extended abstract). In *10th DFN-CERT Workshop*, Hamburg, Germany, Feb 2003.
- [36] Martin Roesh. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the USENIX LISA 1999 Conference*, Nov 1999.
- [37] Lee Rossey, Robert Cunningham, David Fried, Jesse Rabek, Richard Lippman, Joshua Haines, and Marc Zissman. LARIAT: Lincoln Adaptable Real-time Information Assurance Testbed. In *2002 IEEE Aerospace Conference Proceedings*, Mar 2002.
- [38] Security Focus Homepage. <http://www.securityfocus.com/>, 2002.
- [39] Gautam Singaraju, Lawrence Teo, and Yuliang Zheng. A Tested for Quantitative Assessment of Intrusion Detection Systems Using Fuzzy Logic. In *Proceedings of the Second IEEE International Information Assurance Workshop*, 2004.
- [40] Amin Vahdat, Ken Yocum, Kevin Walsh, Priya Mahadevan, Dejan Kostic, Jeff Chase, and David Becker. Scalability and Accuracy in a Large-Scale Network Emulator. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, Boston, MA, Dec 2002.
- [41] Giovanni Vigna, William Robertson, and Davide Balzarotti. Testing Network-based Intrusion Detection Signatures Using Mutant Exploits. In *Proceedings of the ACM Conference on Computer and Communication Security*, pages 21–30, Washington, DC, Oct 2004.
- [42] VMware Inc. *VMware Workstation 5.0 User's Manual*, 2005.
- [43] Andrew Whitaker, Marianne Shaw, and Steven Gribble. Scale and Performance in the Denali Isolation Kernel. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, Boston, MA, Dec 2002.
- [44] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*, pages 255–270, Boston, MA, December 2002. USENIX Association.

A Windows Exploits

Please see Table 1.

B Linux Exploits

Please see Table 2.

C Buggy Web Server Exploits

Please see Table 3.

D BSD Exploits

Please see Table 4.

Exploit	BID	Description	Vuln. Images	Expected Result
IIS-Traversal	1806	"Microsoft IIS 4.0 and 5.0 are both vulnerable to double dot ".." directory traversal exploitation if extended UNICODE character representations are used in substitution for "/" and "" [10]	2000 AS SP0	Remote Command Execution
IIS-Printer	2674	"Due to an unchecked buffer in msw3prt.dll, a maliciously crafted HTTP .printer request containing approx 420 bytes in the 'Host:' field will allow the execution of arbitrary code." [10]	2000 AS SP0	Reverse-connect Shell
IIS-Double Decoding	2708	Internet Information Services mishandles security checks on CGI_BIN requests leading to arbitrary code execution. [10]	2000 AS SP0	Remote Command Execution
Microsoft FTP	4482	DOS in the FTP service caused by STAT requests with a large number of characters. [10]	2000 AS SP0 XP Pro SP0	Denial of Service
NSIISlog.dll	8035	A Windows Media Services buffer overflow "due to a problem with how the logging IS-API extension handles incoming client requests. This could cause arbitrary code execution in IIS, which is exploitable through Media Services" [10]	2000 AS SP0 2000 AS SP4	Bind Shell
DCOM RPC	8205	Buffer overflow in Microsoft's remote procedure call interface due to improper bounds checking of requests to TCP port 134 [10].	2000 AS SP0 2000 AS SP4 XP Pro SP0 XP Pro SP1a Server 2003 SP0	Remote Shell
LSASRV.dll	10108	LSASS (Local Security Authority Subsystem Service) buffer overflow in the lsass.exe or netrap.dll files.	2000 AS SP0 2000 AS SP4 XP Pro SP0 XP Pro SP1a	Bind Shell
NetDDE	11372	The NetDDE service fails "to properly verify the lengths of strings contained within unspecified network messages prior to copying them into finite buffers" [10].	2000 AS SP4 XP Pro SP0 XP Pro SP1a	Bind Shell
EMF Metafile	11375	In Windows WMF/EMF image rendering library, "a failure of the affected library to properly verify the lengths of strings contained within an affected image file prior to copying them into finite buffers" [10].	XP Pro SP0 XP Pro SP1a Server 2003 SP0	Bind Shell
IE ani Files	12233	"The ANI file header handling routines contained in the 'user32.dll' library...may be leveraged to force the execution of attacker-supplied instructions" [10]	XP Pro SP0 XP Pro SP1a Server 2003 SP0	Bind Shell

Exploit	BID	Description	Vuln. Images	Expected Result
IE 6 CSS	12765	Malformed Cascading Style Sheets embedded in “honeypot” webpages can cause a buffer overflow in Internet Explorer leading to a browser crash or possible Bind Shell.	2000 AS SP4 XP Pro SP0 XP Pro SP1a Server 2003 SP0	Browser Crash
Modern LAND	N/A	A malformed packet causes a temporary DOS condition.	Server 2003 SP0	Denial of Service
HTTP .hta Files	N/A	The HTML Application Host (MSHTA) contains an error where opening non-executable files can lead to the execution of arbitrary commands.	2000 AS SP0 2000 AS SP4 XP Pro SP0 XP Pro SP1a Server 2003 SP0	Remote Command Execution

Table 1: The ViSe Windows Exploits.

Exploit	BID	Description	Vuln. OS	Image	Expected Result
UW-Imapd	1110	A buffer overflow exists in the list command. "By supplying a long, well-crafted string as the second argument to the list command, it becomes possible to execute code on the machine" [10].	Redhat 6.2	Sploit Expl.	Remote-connect Shell
WU-ftpd	1387	"Wu-ftpd is vulnerable to a very serious remote attack in the SITE EXEC implementation. Because of user input going directly into a format string for a *printf function, it is possible to overwrite important data, such as a return address, on the stack" [10].	Redhat 6.2	Sploit Expl.	Remote-connect Shell
OpenSSL	5363	"A buffer overflow has been reported in the handling of the client key value during the negotiation of the SSLv2 protocol. A malicious client may be able to exploit this vulnerability to execute arbitrary code as the vulnerable server process, or possibly to create a denial of service condition" [10].	Redhat 7.3	OpenSSL Expl.	Reverse-connect Shell
Ghostview 3.5.8	5808	"It has been reported that an insecure sscanf() function exists in gv. Due to this function, an attacker may be able to put malicious code in the %%PageOrder: portion of a file. When this malicious file is opened with gv, the code would be executed in the security context of the user opening the file" [10].	Fedora Core 3	GV 3.5.8 Expl.	Bind Shell
Iwconfig	8901	strcpy() functions are used to copy command-line input to fixed size buffers, leading to a classic buffer overflow.	Fedora Core 3	Iwconfig Expl.	Shell
Rsync 2.5.6	9153	"rsync has been reported prone to an undisclosed heap overflow vulnerability when running in daemon mode. The issue has been reported to be remotely exploitable and will provide for an execution of arbitrary code" [10]	Fedora Core 3	Rsync Expl.	Shell
PNG Slapper	10857	"A stack-based buffer overrun vulnerability exists" where an attacker can send "a malicious image to an unsuspecting user. When this image is viewed, the vulnerability may be triggered resulting in code execution occurring in the context of the user that viewed the malicious image" [10]	Fedora Core 3	PNG 1.2.2 Expl.	Bind Shell

Exploit	BID	Description	Vuln. OS	Image	Expected Result
XView 3.10	10985	"A stack based buffer overflow exists in the 'xvbm.c' source file. It is reported that a user-supplied value is employed to iterate a loop that copies data into a finite stack based buffer" [10]	Fedora Core 3	XView Expl.	Bind Shell
ReadCD 2.01	11075	readcd does not check the validity of the RSH environment variable before use, making it possible to set it to a shell generating file.	Fedora Core 3	CDrecord Expl.	Shell
FLC 1.0.4	N/A	File Listing Creator is vulnerable to buffer overflow based on the command-line arguments passed to the program.	Fedora Core 3	FLC Expl.	Shell
Checkpwd	N/A	A toy program used as part of a Fall 2004 cs279 homework assignment to let students practice local buffer overflows.	Debian 3.0	Checkpwd Expl.	Shell
Ftpd	N/A	A toy ftp daemon used as part of a Fall 2004 cs279 homework assignment to let students practice remote buffer overflows.	Fedora Core 3	ToyFtpd Expl.	Bind Shell

Table 2: The ViSe Linux Exploits.

Exploit	Line #	Description	Expected Result
Password Disclosure	70	The attacker may reconstruct <code>/etc/passwd</code> and get the encrypted password of any user by sending a command request. [6]	Reconstruction of password file
Buffer Overflow	169	The function <code>exec_command(..)</code> performs a vulnerable string copy by placing a command of arbitrary length in <code>buffer[256]</code> . The web server attempts to cram a 512 byte array into a 256 byte array with no bounds checking. [6]	Bind Shell
Long Jump Manipulation	169	In the <code>exec_command(..)</code> function, the <code>jmpbuf</code> structure is placed in memory just above the buffer <code>char buffer[256]</code> which can be overflowed by providing a sufficiently long string as input. [6]	Bind Shell
Buffer Overflow	195	In <code>manage_request(..)</code> , a while loop fills <code>buffer[512]</code> until the server receives a newline character. [6]	Bind Shell
Ignored Error	235	Even if GET does not prepend to a request, BWS executes as if it were despite checking for it. [6]	Command Execution
Unchecked Input Exec.	254	If a request is for <code>/cgi-bin</code> then <code>system()</code> is called to execute unchecked user input. [6]	Command Execution
Symlink Creation	263	When BWS checks whether <code>CGI_PATH</code> exists, it does not check if it is a link. Since <code>/tmp</code> is a directory open to public, any user can create a symbolic link inside this directory. [6]	Link Creation
Cross Site Scripting	263	Requests with embedded javascript can be made from spoofed IPs. [6]	Gather Information
Symlink Creation	275	When a request of the form <code>'GET /'</code> is made, the server retrieves <code>/tmp/html/index.html</code> , but it does not check if it is a link. Thus, a symbolic link called <code>/tmp/html/index.html</code> can be created to point to any program and can be executed. [6]	Link Creation
Unchecked Input	302	No checks are made on <code>DOC_PATH</code> requests, so the command can be used to view any file that BWS has permissions to view. [6]	File Browsing
Format String	311	The function <code>manage_request(...)</code> contains an unvalidated <code>printf(buffer)</code> command. Shellcode is passed to the function as a parameter then jumped to upon return. [6]	Bind Shell

Exploit	Line #	Description	Expected Result
Permissions Unset	356	BWS creates files and directories without modifying their permissions. The default umask may allow access to files or directories from the world, meaning anyone can modify or delete them. [6]	cgi-bin file replacement
Command Placement	356	None of the system utilities used by BWS to create its directories or files include the absolute path to each command. For example, <code>mkdir /tmp/cgi-bin</code> . If the PATH of the user who executes BWS is set to '.' then the attacker can create his/her own version of <code>mkdir</code> in the directory where the server is launched. [6]	Executable Pre-placement
Clear-text Password	455	During login, the username and password are transmitted in an unencrypted state that allows for sniffers to obtain valid username and password pairs. [6]	Sniffed Password

Table 3: The ViSe Buggy Web Server Exploits.

Exploit	BID	Description	Vuln. OS	Image	Expected Result
Apache Chunked Encoding	5033	"When processing requests coded with the 'Chunked Encoding' mechanism, Apache fails to properly calculate required buffer sizes. This is believed to be due to improper (signed) interpretation of an unsigned integer value. Consequently, several conditions may occur that have security implications. It has been reported that a buffer overrun and signal race condition occur. Exploitation of these conditions may result in the execution of arbitrary code" [10]	FreeBSD 4.5	Apache Chunked Encoding Expl.	Reverse-connect Shell

Table 4: The ViSe BSD Exploit.