Editors: Deborah A. Frincke, Deborah.frincke@pnl.gov
Richard Ford, rford@se.fit.edu

Sandboxing and Virtualization

Modern Tools for Combating Malware

Chris Greamo and Anup Ghosh
Invincea

Malware is probably the most significant computer security threat to enterprises and businesses alike. The numbers show this dramatically; in the third quarter of 2010, McAfee reported an average 6 million new botnet infections every month, 60,000 new pieces of malware per day, and 60 percent of top Google search terms returning links to malicious sites within the first 100 results (www.mcafee.com/us/about/news/2010/q4/20101117-01.aspx). In fact, Cisco has reported that 10 percent of all Web malware is encountered through search engine traffic, with Google being the largest search engine provider of malicious links (www.cisco.com/en/US/prod/collateral/vpndevc/3q10_cisco_threat.pdf).

Although targeted malware such as Stuxnet and Aurora grabbed the headlines, most malware in McAfee's Global Top 10 list supports cybercrime, which means the threat is largely profit driven. Another important trend we're seeing is that users play a critical role in infecting their machines. The "Here You Have" email worm in September 2010 lured users to click on a link that installed malware and sent the message out to the users' contacts. Also in September, an email purporting to be a LinkedIn update contained links to sites that infected users when they clicked through.[1]

While it's clear that malware continues to be a major scourge, a study by Cyveillance in April 2010 showed that current antivirus products are largely ineffective in addressing the threat.[2] The Cyveillance study, which included 13 of the most popular antivirus engines, found that, on average, only 19 percent of malware was detected on the first day after the malware became known. Even more shocking, the average detection rate for all 13 products only increased to 61.7 percent on average by day 30. Antivirus products are still an important component of desktop security, but it's clear that they aren't effective enough against a threat that prodigiously produces new malware variants by the tens of thousands every day.

Enter Sandboxing and Application Virtualization

To address the exposure gap left by antivirus products, an emerging category of desktop security products that use application-level sandboxing attempts to address malware threats by containing their malicious behavior. High-profile applications that now employ sandboxing include the Google Chrome browser, Internet Explorer Protect Mode, and Adobe Reader X.

Although these three applications employ Microsoft's Practical Sandbox approach,[3] there are in fact several techniques to encapsulate and contain bad behavior from exploited Internet-based applications and malware. These techniques range from restricting account privileges to defining a separate file system for an application to running an application in its own full virtual machine. Separating untrusted code from the rest of the system using some form of a sandbox can significantly mitigate the malware threat by preventing malicious actions from compromising the desktop. While this approach holds much promise, the details of any particular implementation are critical when it comes to fully understanding what protection a product offers.

Figure 1 depicts the range of sandboxing and virtualization technologies available today to address the malware threat and compares them in terms of level of protection and ease of deployment. Today, many users employ some form of in-browser security as shown in the far left quadrant. Many antivirus suites include plug-ins for the browser that stop users from visiting known "bad" websites or examine Web content for known malicious code. However, these approaches have proven inadequate for addressing current malware, which is why the more robust approaches (to the right) have been developed. The dividing line in the middle of Figure 1 separates "native" approaches from fully virtualized ones—this

 1540-7993/11/$26.00 © 2011 IEEE COPUBLISHED BY THE IEEE COMPUTER AND RELIABILITY SOCIETIES

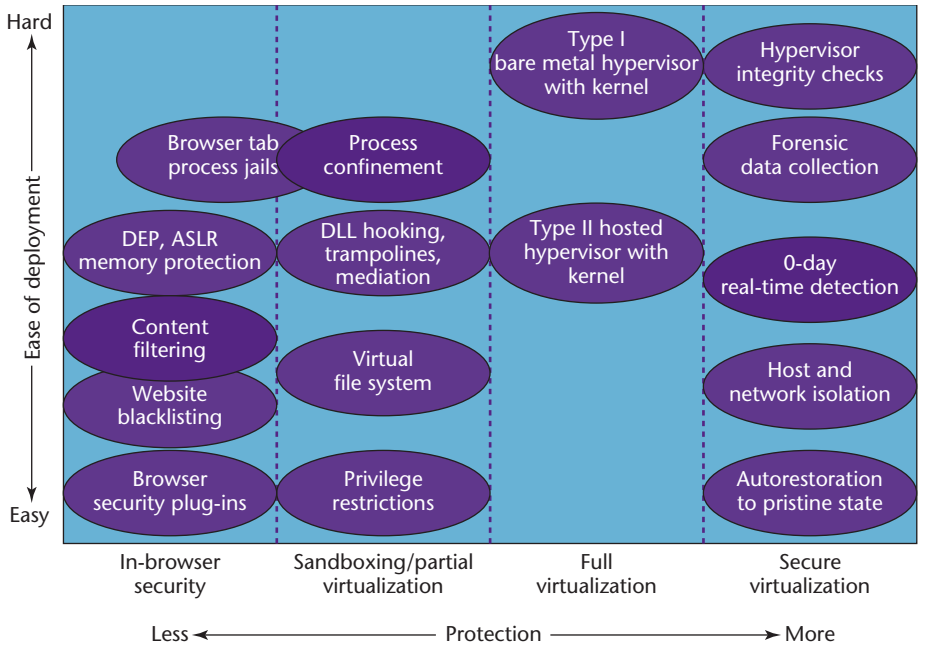

Figure 1. Sandboxing and virtualization malware threat protection landscape. Comparison of current sandboxing/virtualization security technologies in terms of protection level and ease of deployment.

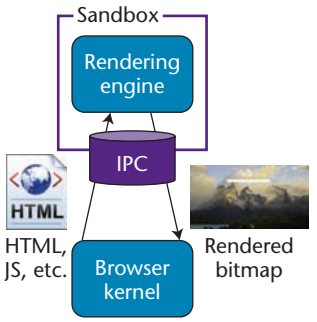

Figure 2. Google Chrome's security architecture. Google has used sandboxing technology in Chrome to limit the host modifications possible from attacks that exploit the browser's rendering engine.

turns out to be the most significant differentiator in addressing the malicious code problem. Technologies to the left of the dividing line in Figure 1 run the application natively on the host operating system (OS) and will use some form of sandboxing technique to contain malicious behavior. Techniques to the right run the target application in its own virtual OS that includes the full set of system libraries and services needed for that application.

Application Sandboxing

Many application sandboxes, including those that employ Microsoft's version, require the application to use new APIs to take advantage of the sandbox environment. For existing applications, this means recoding the application. Microsoft Practical Sandbox is the technology underlying the protected modes found in Internet Explorer 7, 8, and 9, Microsoft Office, Google Chrome for Windows, and Acrobat Reader X. The changes to the application are significant and also require architectural modifications such as dividing a traditionally monolithic (single-process) application into multiple processes, where one of the processes runs in the sandbox, while the other process manages the sandboxed processes' requests. This is, for example, the case with both Chrome and Reader X.

Sandboxes are typically applied to the components of the application with the highest vulnerability. The sandboxed process runs with restricted privileges and will employ DLL hooking and memory trampolines as needed to certain APIs. This means that the sandbox can examine certain system calls for malicious behavior, then rewrite or block them as appropriate. Chrome is a great example. Figure 2 depicts its software architecture (http://dev.chromium.org/developers/design-documents/sandbox). Because Chrome's authors were primarily concerned with preventing exploits that target the rendering engine (the most common vulnerabilities for browsers) and install persistent malware on the host (the most common attack scenario), they chose to place only that component in the sandbox, allowing other browser components to run unmanaged in the "browser kernel" process.

The restrictions placed on the rendering engine limit write access to sensitive areas of the host. The restrictions don't prevent a compromised rendering engine from reading and leaking data directly from the system. In these instances, sandboxing is more of a surgical approach to security, allowing the author to target risky components or functionality in applications. While the security mechanisms in this category can help reduce the application's exploitable surface area, they don't prevent attacks that target functionality outside the sandboxed process or system calls that aren't mediated from gaining privileged access to the host. For example, published exploitable vulnerabilities in Chrome, including vulnerabilities CVE-2010-2898 and CVE-2010-2897, illustrate this point.[4,5]

Although we use Chrome as an example, the same applies to Reader X and Internet Explorer. An attack vector Verizon pointed out (www.verizonbusiness.com/resources/whitepapers/wp_escapingmicrosoftprotectedmodeinternetexplorer_en_xg.pdf) shows how Internet Explorer's Protect Mode can be bypassed by

exploiting the fact that it's disabled for local or intranet zone sites. The attack surface that will likely be exploited in application sandboxing approaches will be system calls into the kernel that aren't mediated—for example, the kernel exploit Microsoft reported and patched on 10 August 2010 contained by application sandboxes (www.microsoft.com/technet/security/bulletin/MS10-048.mspx). This type of kernel exploitation has become a favorite target of hackers, partially due to the emergence of application sandboxes.

Besides kernel vulnerabilities, sandboxing still leaves the desktop open to attacks that involve asking the user's permission. Attacks that lure users into installing software, such as a codec multimedia plug-in, will effectively bypass the sandbox. These so-called trust-exploitation attacks convince users to install software unwittingly by sending links to malicious sites from users in the victim's social network or as tweets from people the user follows on Twitter. Rogue antivirus sites often bait users to install malicious software by scaring them into clicking on dialog boxes that supposedly clean up infections. If a user can be lured or scared into clicking through dialog boxes that install software, then application sandboxes aren't going to provide much protection, even if they ask users if they want to install the software twice.

Partial Virtualization

Partial virtualization techniques usually involve a combination of privilege restrictions by user accounts and a virtual file system. Other terms for this approach include lightweight virtualization, OS virtualization, process virtualization, and process confinement. The "virtualization" typically refers to providing the target application the appearance of its own virtual file system, often including a system registry on Windows, so that file writes are to this virtualized resource rather than the file system of the underlying host. When users exit the environment, they're often faced with making a decision about which file writes to keep persistently on the host—a decision rife with security consequences. What distinguishes partial virtualization techniques from full virtualization is that partial virtualization approaches share the same OS kernel as the host. Consequently, an exploit of the kernel will defeat a partial virtualization approach.

Full Virtualization

Full virtualization techniques, shown on the right half of Figure 1, are also called hardware virtualization because the hardware layer and resources, including device drivers, are virtualized for a "guest" OS by a hypervisor layer. Utilizing hardware virtualization-based techniques, a malware protection solution runs the target application in its own OS in a virtual machine. The full virtualization hypervisor can be Type I, in which the virtual OS runs on a "bare metal" hypervisor that runs directly on the CPU, or Type II, in which the virtual OS runs as an application within a host OS. In either case, the application runs with its own full kernel and standard OS resources. The implications for malware threat protection are significant—with hardware virtualization in place, a malware infection of the application or the kernel, caused by a drive-by download exploit or by user interaction, will be contained within the virtual environment.

Employing full virtualization for malware threat protection has recently become feasible on standard commodity PC hardware because of significant improvement in CPU chipsets (including virtualization instruction support in x86 instruction set architecture) and the availability of substantial memory at low cost. Hardware virtualization involves running the target application in a virtual machine on the host, along with a kernel and OS services. Most people think of hardware virtualization in a server context (such as running a Type I bare-metal hypervisor), but when applied to desktop security, Type II hypervisors can run a separate OS as an independent process on the host OS. The most significant difference between full virtualization and sandboxing/application virtualization is that the target applications run in a dedicated OS, known as the guest, on the host. Consequently, kernel exploitation in the guest is isolated from the host OS kernel. Even a privilege escalation attack that installs a Ring 0 keystroke logger or rootkit will be contained to the virtual environment and subsequently restored to a pristine image using a full virtualization-based approach. In full virtualization-based solutions, the guest OS type can be the same as the host type (such as Windows on Windows) or different (Linux on Windows) to further change the nature of the vulnerabilities and threat space.

Full virtualization provides a high degree of containment, in terms of the vulnerable application, without requiring changes to the application. If any piece of the application is exploited, the attacker only succeeds in gaining access to the guest environment's data, applications, resources, and OS, not the underlying host's.

The isolation afforded by full virtualization can present significant usability challenges. The virtualized application must be integrated with the host desktop in a way that provides a seamless user experience. Full virtualization presents several usability challenges, such as users'

personalizations of virtualized applications, access to local devices, seamless access to persistent storage, and seamless launch for application and document requests. Users have low tolerance for security mechanisms that introduce friction into the experience.

Secure Hardware Virtualization

Simply employing hardware virtualization doesn't provide a secure solution. Rather, hardware virtualization can be a core component of a secure confinement solution. Although not an exhaustive list, the following attributes are necessary for a secure confinement solution:

- host and network isolation,
- real-time detection that covers previously unseen attacks,
- fast and complete recovery to a known clean state,
- forensic data collection on infection, and
- hypervisor integrity checks.

While hardware virtualization approaches provide kernel confinement from the host OS, it does not, by default, provide network confinement, which prevents an infected process from causing damage on the network. Real-time detection of unknown threats coupled with automated recovery to a known pristine image is necessary to prevent an infected virtual machine from compromising user sessions and credentials or attacking other machines on the network. Finally, the virtualization layer itself might be attacked by malicious code through vulnerabilities in its interface with software running in the virtual machine. Advanced inspection techniques can be employed to vouch for the virtualization layer's integrity to ensure it isn't compromised by malicious software.

It's clear the market is moving in the direction of sandboxing now that vendors such as Google, Microsoft, and Adobe are including sandboxes in their flagship products to contain malicious code threats. Although application sandboxing and partial virtualization approaches can help contain some malicious code threats, they still leave a broad attack surface area exposed, while at the same time frequently requiring users to make good security decisions, a flawed strategy the security industry must change.

While lightweight sandboxing technologies are now in mainstream products, it's clear that the market will be pulled to more robust sandboxing approaches to address the current and future malware threats. Full virtualization techniques offer significant promise in combating malware. Researchers have known this for some time and have employed full virtualization techniques to study malware behavior in a safe environment. Riding Moore's law on improvements in hardware at lower cost, commodity PC hardware now enables standard and corporate users to benefit from the same robust approach to malware protection. Although virtualizing applications for malware protection is a necessary step, it isn't sufficient by itself. Instead, secure virtualization with high usability is essential to properly confine, monitor, remediate, and report malware infections from exploited applications. □

References

1. MX Lab, "Email Messages with Subject 'LinkedIn Alert' Lead to Malware," blog, 27 Sept. 2010; http://blog.mxlab.eu/2010/09/27/email-messages-with-subject-linkedin-alert-lead-to-malware.
2. "Cyveillance Testing Finds AV Vendors Detect on Average Less Than 19% of Malware Attacks," Cyveillance Press Release, 4 Aug. 2010; www.cyveillance.com/web/news/press_rel/2010/2010-08-04.asp.
3. D. LeBlanc, "Practical Windows Sandboxing Part 1," blog, 27 July 2007; http://blogs.msdn.com/b/david_leblanc/archive/2007/07/27/practical-windows-sandboxing-part-1.aspx.
4. "Google Chrome before 5.0.375.125 Does Not Properly Mitigate an Unspecified Flaw in the GMU C Library, which Has Unknown Impact and Attack Vectors," CVE-2010-2898, 2010; http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-2898.
5. "Google Chrome before 5.0.375.125 Does Not Properly Mitigate an Unspecified Flaw in the Windows Kernel, which Has Unknown Impact and Attack Vectors," CVE-2010-2897, 2010; http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-2897.

Chris Greamo *is vice president of research at Invincea, a venture-backed security software start-up developing next generation Internet security products to protect desktops and computer networks, and manages Invincea Labs, the research and development arm of Invincea. He has a BS in electrical engineering from Carnegie Mellon University and an MS in electrical engineering from MIT. Contact him at chris.greamo@invincea.com.*

Anup Ghosh *is founder and chief scientist of Invincea. Ghosh is also a research professor in George Mason University's Volgenau School of Engineering. Ghosh received his PhD in electrical engineering from the University of Virginia. Contact him at anup.ghosh@invincea.com.*

cn *Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.*

This article was featured in

computing now
ACCESS | DISCOVER | ENGAGE

For access to more content from the IEEE Computer Society,
see computingnow.computer.org.

IEEE IEEE computer society