Sandboxing & Virtualization: Modern Tools for Combating Malware

Chris Greamo & Anup Ghosh
Invincea, Inc

December 2010

The Malware Scourge

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 reports [1]:

- 6,000,000 new botnet infections every month on average
- 60,000 new pieces of malware per day
- 60% of top Google search terms returned links to malicious sites within the first 100 results

While targeted malware such as Stuxnet and Aurora grabbed the headlines, most malware in McAfee's Global Top 10 list supports cyber crime, which means the threat is largely profit driven. As the last bullet above shows, malware is largely pulled by users onto their machines.

Another important trend highlighted by the last bullet 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 email to the user's contacts. Also in September, an email purporting to be a LinkedIn email update contained links to sites that infected the user when they clicked through [2]. Cisco reported that 10% of all Web malware was encountered through search engine traffic, with Google being the largest search engine provider of malicious links [3].

While it is clear that malware continues to be a major scourge, a study [4] by Cyveillance in April 2010 showed that current anti-virus products are largely ineffective in addressing the threat. The Cyveillance study, which included thirteen of the most popular anti-virus engines, found that on average only 19% of malware was detected on day 1 after the malware was known. Even more shocking, the average detection rate for all 13 products only increased to 61.7% on average by day 30. While anti-virus products are still an important component of desktop security, 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 anti-virus products, an emerging category of desktop security products that use application level sandboxing is attempting to address

malware threats by containing their malicious behavior. High-profile applications that now employ sandboxing include Google Chrome browser, Internet Explorer Protect Mode, and Adobe Reader X.

While these three applications employ Microsoft's Practical Sandboxing approach [5], there are in fact several techniques to encapsulate and contain bad behavior from exploited Internet-based applications and malware. The techniques range from restricted 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 without having prior knowledge of the threat. While the approach holds much promise, the details of any particular implementation are critical when it comes to fully understanding the protection afforded by a product.

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 anti-virus suites include plug-ins for the browser that will stop users from visiting known bad websites or examine web content for known malicious code. 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 approaches – and this 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 OS and will use some form of sandboxing technique to contain malicious behavior. Techniques to the right of the dividing line run the target application in its own virtual operating system that includes the full set of system libraries and services needed for that application.

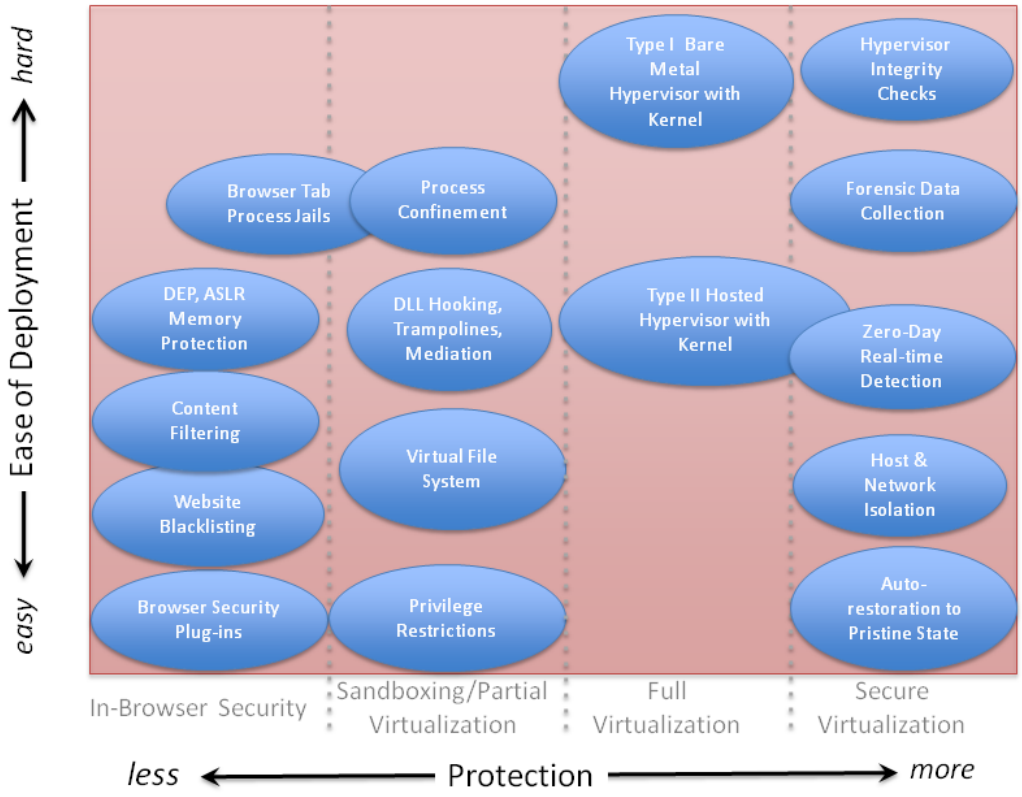

Figure 1: Sandboxing and Virtualization Malware Threat Protection Landscape

Application Sandboxing

Many application sandboxes, including those that employ Microsoft Practical Sandbox, require the application to utilize new APIs to take advantage of the sandbox environment. For existing applications, this means re-coding the application. Microsoft Practical Sandboxing is the technology underlying the protected modes found in Internet Explorer 7, 8, and 9, Microsoft Office, Google Chrome for Windows, and Reader X. The changes to the application are significant and also require architectural modifications such as dividing a traditionally monolithic (i.e., 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 the Chrome Browser 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 re-write them or block them as appropriate. The Chrome browser is a great example. Figure 2 depicts Chrome's software architecture [6]. Since Chrome's authors were primarily concerned with preventing exploits that target the rendering engine (the most common vulnerabilities for Web 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 components of the browser to run unmanaged in the "browser kernel" process.

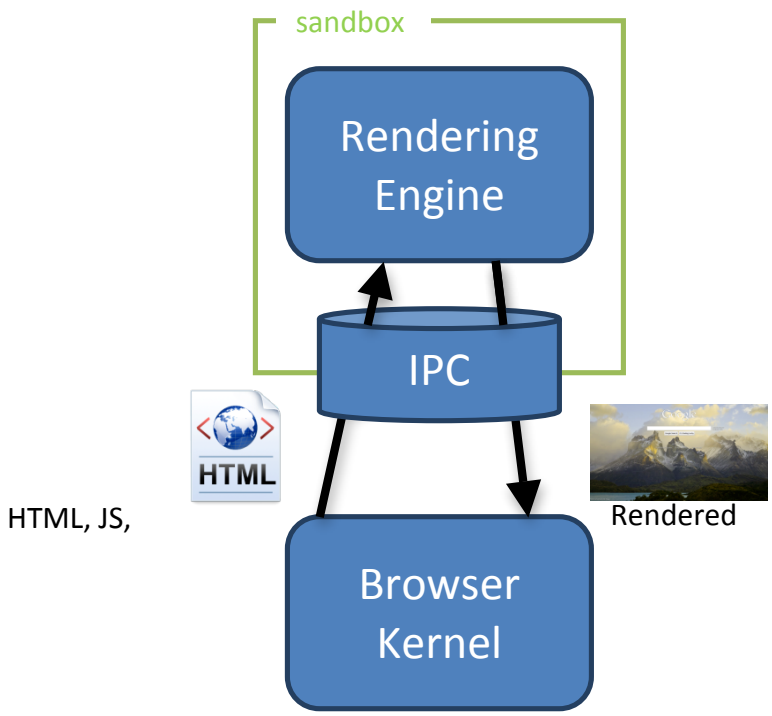

Figure 2: Google Chrome's Security Architecture

The restrictions placed on the rendering engine limit write access to sensitive areas of the host. The restrictions do not 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 exploitable surface area of the application, they do not prevent attacks that target functionality outside the sandboxed process or system calls that are not 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 [7] [8].

While we use Chrome as an example, the same applies to Reader X and Internet Explorer. An attack vector pointed out by Verizon [9] shows how Internet Explorer's Protect Mode can be bypassed by exploiting the fact that it is 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 are not mediated. For example, the kernel exploit reported and patched by Microsoft on August 10, 2010 [10] was not contained by application sandboxes. 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 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, for example. Rogue anti-virus sites will 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 scared or lured into clicking through dialog boxes that end up installing software, then application sandboxes are not 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 the user exits the environment, he/she is often faced with making a decision on 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 operating system kernel as the host. As a result, 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 VM on the host, along with a kernel and OS services. While most people think of hardware virtualization in a server context (e.g., running a Type I bare-metal hypervisor), when applied to desktop security, Type II hypervisors are used to 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. *As a result, the exploitation of the kernel 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 (e.g., Windows on Windows) or different (e.g., 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 data, applications, resources and OS of the *guest* environment, not the underlying host.

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 a number of usability challenges however, including supporting users' personalizations of virtualized applications, access to local devices such as printers, 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 does not provide a secure solution. Rather, hardware virtualization can be a core component of a secure confinement solution. While not an exhaustive list, the following attributes are necessary for a secure confinement solution:

- Host & network isolation
- Real-time detection that covers previously unseen attacks
- Fast and complete recovery to a known clean state
- Forensic data collection on infection

- Hypervisor integrity checks

While hardware virtualization approaches provide kernel confinement from the host OS, it does not by default provide network confinement. *Network confinement* 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 may 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 integrity of the virtualization layer to ensure it isn't compromised by malicious software.

CONCLUSION

Malware is able to defeat traditional security measures including firewalls, Web gateways, network-based monitoring, and end-point anti-virus solutions because existing approaches depend on prior knowledge of malware to detect infections. To address this gap in malware threat protection, emerging malware protection tools include sandboxing and virtualization techniques.

In this article, we examined the technologies that underpin sandboxing and virtualization techniques. Application sandboxing has gone mainstream now that vendors including Google, Microsoft, and Adobe are including sandboxes in their flagship products to contain malicious code threats. While application sandboxing and partial virtualization approaches can help contain some malicious code threats, they leave a broad attack surface area exposed, e.g., unmediated system calls, while also frequently depending on users to make good security decisions.

Full virtualization techniques offer significant promise in combatting malware. Malware researchers have realized this for some time as they have employed full virtualization techniques to study the behavior of malware 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. While 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] **McAfee, Inc. Q3 Threat Report Reveals Average Daily Malware Growth at an All Time High; Spam at Lowest Point since 2008**, McAfee Press Release, http://www.mcafee.com/us/about/news/2010/q4/20101117-01.aspx

[2] **Email messages with subject "LinkedIn Alert" lead to malware…**, MX LAB Blog, 27 September 2010, http://blog.mxlab.eu/2010/09/27/email-messages-with-subject-linkedin-alert-lead-to-malware/

[3] **Cisco 3Q10 Global Threat Report**, http://www.cisco.com/en/US/prod/collateral/vpndevc/3q10_cisco_threat.pdf

[4] **Cyveillance testing finds AV vendors detect on average less than 19% of malware**, Cyveillance Press Release, 4 August 2010, http://www.cyveillance.com/web/news/press_rel/2010/2010-08-04.asp

[5] **Practical Windows Sandboxing – Part 1**, David Leblanc, 27 July 2007, http://blogs.msdn.com/b/david_leblanc/archive/2007/07/27/practical-windows-sandboxing-part-1.aspx

[6] **The Chromium Projects – Sandbox**, http://dev.chromium.org/developers/design-documents/sandbox

[7] 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, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-2898

[8] 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, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-2897

[9] **Escaping from Microsoft's Protected Mode Internet Explorer**, Verizon, http://www.verizonbusiness.com/resources/whitepapers/wp_escapingmicrosoftprotectedmodeinternetexplorer_en_xg.pdf

[10] **Vulnerabilities in Windows Kernel-mode Drivers Could Allow Elevation of Privilege (2160329)**, Microsoft Security Bulletin MS 10-048, 10 August 2010, http://www.microsoft.com/technet/security/bulletin/MS10-048.mspx