Interested in learning
more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

A Virtually Secure Browser

The goal of this paper will be to provide a very brief exploration of different attack vectors affecting web browsers, and then to posit a better method for securing, and maintaining the security of browsers through virtualization.

Copyright SANS Institute
Author Retains Full Rights

AD

© SANS Institute 2009, Author retains full rights.

A Virtually Secure Browser

GSEC Gold Certification

Author: Seth Misenar seth@contextsecurity.com

Adviser: Jim Purcell

Accepted: June 13, 2009

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

Abstract

This paper will discuss an increasingly important aspect of information security, the browser. I will first survey the current threat and vulnerability landscape associated with the browser. After basic risk assessment, I will proffer two virtualization oriented approaches, sandboxing and application virtualization, which could help to mitigate the increased risk associated with using a browser. Finally, I will explore two applications, Sandboxie and ThinApp, which serve as representative examples of how one can leverage sandboxing and application virtualization to possibly achieve greater browser security.

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

Table of Contents

1. Introduction.....5

2. Browser (In)security.....7

2.1 Threat and Vulnerability Landscape 7

2.1.1 Threat Trends 7

2.1.2 Vulnerability Trends 10

2.2 The Browsers 11

2.2.1 Internet Explorer 12

2.2.2 Firefox 13

3. Sandboxing.....16

3.1 Sandboxie 19

4. Application Virtualization.....23

4.1 ThinApp 24

5. Conclusions.....29

6. References.....31

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

1. Introduction

As organizations have shown themselves more capable of applying basic operating system patches and denying trivial access to unnecessary services on internet facing systems, attackers have been engaging more regularly in client side attacks. Exploits leveraging vulnerabilities in basic client productivity applications have become increasingly common (Turner, 2008a). Enterprise organizations have been forced to bolster their patch management capabilities to account for client side applications rather than merely maintaining basic Operating System patches. Even those organizations that have shown a great facility for patching 3rd party desktop applications generally are still encumbered with the significant exposure due to allowing end users to access the internet for web browsing purposes.

Although few would deny that the best solution to the pernicious problem of browser related security exposure is to simply deny end users the right to access the internet through a browser, I find this solution to be largely untenable for the modern enterprise[1]. This paper will work from the assumption that the browser is indeed a necessary component for at least some end users, but one that is due some security oriented

1 Although this paper will proceed under the assumption that users accessing the internet via the browser is, in fact, necessary, this fact needs to be borne out in your environment. Each organization might find segments of their population that, in all honesty, have no pressing business need for internet access. Naturally, the most secure option in this case is to remove the superfluous access.

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

attention. The goal of this paper will be to provide a very brief exploration of different attack vectors affecting web browsers, and then to posit a better method for securing, and maintaining the security of browsers through virtualization.

To this end, two approaches to browser virtualization will be considered, sandboxing and application virtualization. Sandboxie will be the product used to explore the sandboxing approach to browser security. Sandboxing, as it relates to browser security, implies only exposing to the browser a virtualized environment with which to interact[2]. After use, the sandbox to which the browser has potentially written changes can be completely wiped. This approach to virtualization provides for the end user controlling what will and will not be allowed through the sandbox.

For some organizations the degree to which Sandboxie might require end user involvement could present a significant obstacle to deployment. VMWare's ThinApp (formerly Thinstall) will be offered as an alternative means for browser security through application virtualization. ThinApp presents a different virtualization paradigm than does Sandboxie. Rather than virtualizing the operating system components with which the browser will interact, ThinApp can be used to virtualize the browser itself, which can be streamed from a central server using a particular standardized configuration. The possibility for centralized management and control of the browser might

2 As a point of fact, Sandboxie is never described on its website as a virtualization product, but rather simply as a tool that leverages isolation.

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

remove one of the major disincentives to enterprise adoption of Firefox. While a default install of Firefox is not demonstrably more secure than the latest offering of Internet Explorer, the extensibility of the browser does seem to offer some additional functionality with respect to security.

Browser insecurity presents a significant exposure to large organizations. This paper seeks to offer two differing approaches to virtualization as means by which we can mitigate some of this risk associated with browsers. However, before proffering the solutions, we need to have at least a basic understanding of some of the problems which are in need of remedying.

2. Browser (In)security

Though our task is not specifically to review the vast landscape of browsers and their relevant security, or lack thereof, a brief exploration of the current threat and vulnerability environment, as it pertains to browsers, will aid us in our review of the efficacy of virtualization as means to achieve a more secure browser. After a brief overview of the threats and vulnerabilities related to browsers, I would be remiss to not also at least touch on the current major browser offerings as they relate to security.

2.1 Threat and Vulnerability Landscape

2.1.1 Threat Trends

Symantec's Internet Security Threat Reports always contain some timely insight on threat trends. The latest offering,

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

Internet Security Threat Report Volume XIII: April, 2008, is certainly no departure from their usual quality. The most important trend related to our purpose is summarized by their highlight, “malicious activity has become web-based” (Turner, 2008a, p. 2). Obviously not all malicious activity is web based, but “attackers adopted stealthier, more focused techniques that target individual computers through the World Wide Web” (Turner, 2008a, p. 2). Although throughout the industry there have been various reasons suggested for this trend, we need not bother ourselves with questions of causality, but rather suffice with acknowledgement of this trend. Though there are some client oriented applications that are rather pervasive, Microsoft Office immediately springs to mind, few, if any, are as ubiquitous as the browser. Opportunistic attackers could do little better for themselves than by weaponizing an exploit targeting the browser as a threat vector.

To this end, we find another major attack trend to be attackers creating or co-opting websites as a delivery mechanism to automatically infect browsers of said site. This nefarious technique is commonly known as drive-by malware or drive-by downloads. This technique has become an increasingly common vector, which is highly successful given the rising numbers of browser and third party extension vulnerabilities. A recent study tested more than 60 million unique URLs, and found that over 3 million of those tested were hosting malware in this fashion (Provos, Mavrommatis, Abu Rajab, & Monrose, 2008, p. 5). While that percentage might seem astounding, the efficacy of these malware propagation sites would be be appreciably

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

diminished if they merely existed and were not widely referenced or linked. However, the same study found that Google queries return at least one of these nefarious sites almost 1.3% of the time (p. 5). Further, of the "top one million URLs appearing in the search engine results, about 6, 000 belong to sites that have been verified as malicious" (p. 5). The prevalence of drive-by download sites coupled with the highly vulnerable surface area presented by the common browser+plugin armed user makes for a rather malware infested petri dish. Although the aforementioned study makes it clear that infections could still occur by simply letting users stumble across malware pushing sites on their own, a more common method is for the attackers to entice the browser to visit the malicious site via a simple email, IM, or other form of social engineering.

Our next threat trend, phishing, further leverages social engineering to attack users. With phishing, an attacker typically attempts to trick users into providing otherwise confidential information. The way in which phishing attempts to trick the user is by making the user believe that they are disclosing the information to a legitimate site. The most basic form of phishing would be for the attacker to present a site that appeared, at first glance, much like the legitimate site to which the user might conceivable disclose this information during the normal course of business.

Though there are certainly other threat trends important to discussion of browser security, we will take this as our point of departure from the threat landscape and instead turn to vulnerability trends.

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

2.1.2 Vulnerability Trends

If threat is one side of the risk coin, then vulnerability is surely the other. Even if attackers were more commonly banging against client side applications, if there was no underlying vulnerability that could be touched then the risk of compromise would be moot. Sadly, as you would expect, there seems to be no shortage of vulnerabilities in the browser and related helper applications. The most recent Symantec Global Internet Threat Report suggested that during 2007 that for all but one vendor included in the study the majority of patched vulnerabilities pertained to browser or client side applications (Turner, 2008b, p. 6).

In addition to the growing prevalence of browser vulnerabilities, there is also no shortage of websites with Cross Site Scripting (XSS) vulnerabilities that can be exploited. Although “XSS is an attack technique that forces a Web site to display malicious code,...the server is merely the host, while the attack executes within the Web browser” (Grossman, Hansen, Petkov, Rager, p. 68, 2007). Though XSS has been known since 1999, most people do not understand how often it is used to exploit browsers. Exploitation of an XSS vulnerability can be extremely damaging to both the system running the victim browser as well as the network on which that client is situated. For example, a compromised client can be used as a pivot point such that, effectively, an external attacker can leverage an the compromised internal system's vantage point for launching further attacks (Cross et al., p. 172, 2007).

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

Another vulnerability in which the browser is actor during exploitation is known as Cross Site Request Forgery (CSRF). Attacks against CSRF vulnerabilities are ones that exploit the browser itself, but rather leverages the trust that a vulnerable application places in an authenticated browser. The actual vulnerability lies in the fact that an attacker can somehow get an authenticated browser to, with the users intent, submit a transaction, which the application implicitly trusts because it was submitted from an authenticated user (Stuttard, Pinto, p. 442, 2007). Imagine an attacker tricking a user into clicking a link which caused the "transfer funds" functionality of a bank to be invoked. Again, this isn't an attack against the browser itself, but the browser contributes to the exploitation of CSRF vulnerabilities, and the user on the other end of the browser is commonly the one being ultimately abused.

2.2 The Browsers

Although there are hundreds of web browsers currently available in the marketplace, we, and our friend the opportunistic attacker will focus on the more major players. The most common browsers, as reported by Net Applications in January of 2009, are, in order by percent market share: Internet Explorer (68.15%); Firefox (21.34%); Safari (7.93%); Chrome (1.04%); Opera (0.71%) (Browser Market Share, 2009). Although both Safari and Chrome are trending upwards with respect to market share, Internet Explorer and Firefox currently seem to be the browsers of choice for the vast majority of users (Top Browser Share Trend, 2009). The goal of this section is to

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

provide a quick security related overview of the two most popular browsers found today.

2.2.1 Internet Explorer

Although Internet Explorer 8 is, at the time of writing, currently available for download, it is still in Beta and has not yet seen wide adoption.[3] The most widely used version of Internet Explorer is currently Internet Explorer 7 (IE7), which was released in October of 2006. Currently, Secunia lists IE7 as having had 70 vulnerabilities over its lifetime, 33 of which have had Secunia advisories created for them with 45% being deemed "highly critical" or above (Vulnerability Report: Microsoft Internet Explorer 7.x, 2009). Also noteworthy is that Secunia suggests that there are currently 9 unpatched vulnerabilities in Internet Explorer 7 with the most severe being rated "moderately critical" (Vulnerability Report: Microsoft Internet Explorer 7.x, 2009). The number of patched and unpatched vulnerabilities notwithstanding, Internet Explorer 7 does present a vastly more secure application than its previous iterations of this popular browser.

Internet Explorer Extensions

Though not nearly as robust or numerous as the possible extensions to Firefox, there are some free addons that bolster the security of Internet Explorer. WOT, Web of Trust, is available for both Firefox and Internet Explorer and provides a

---

3 It also seems likely that adoption after IE8 goes gold will take considerable time given the continued use of IE6 (around 20% market share) more than two years after IE7's release (Browser Market Share, 2009).

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

visual indicator of how risky other users have found a site to be. Additional information on WOT can be found here: http://www.mywot.com. One of the first addons to significantly increase the security of Internet Explorer was DropMyRights by Michael Howard. The purpose of this addon is to ensure that, even if the user running the browser had administrative privileges, Internet Explorer will run with limited privileges, which could mitigate the impact of some successful exploits. Additional information can be found here: http://nonadmin.editme.com/DropMyRights

Enterprise Management

Perhaps the most significant security advantage that Internet Explorer offers is its ability to be managed on an enterprise scale via Group Policy. This feature alone is typically enough to make pursuing alternate browsers a foregone failure. Every other browser typically presents a rather significant administrative burden to centrally manage in the way that Internet Explorer can be.

2.2.2 Firefox

Firefox 3, which was released June 2008, is the most popular alternative to Microsoft's Internet Explorer 7.[4] In the 6 months since its release, Secunia notes 39 vulnerabilities and has released 8 advisories of which 75% were rated "highly critical" or above (Vulnerability Report: Mozilla Firefox 3.x,

4 Strictly speaking, current market share reports still at times show Internet Explorer 6 to be the second most popular browser to Internet Explorer 7, with Firefox 3.0 presenting the 3rd most popular browser (Browser Market Share, 2009).

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

2009). Although Firefox offers security features built into the browser, perhaps the most significant security feature of the browser lies in its aforementioned ability to be extended using freely available code. It should also be noted that this extensibility can, and has, been leveraged for nefarious purposes as well by means of malicious extensions, such as FirestarterFox and FFsniFF (Costoya, 2006).

Firefox Extensions

As alluded to above, the most significant reason that Firefox presents a compelling alternative to Internet Explorer is due to its numerous freely available extensions. Most extensions, which can typically be found at https://addons.mozilla.org, have nothing to do with security, but there are some extremely compelling browser security oriented extensions. We highlight a few of these offerings below:

NoScript – Perhaps the most important Firefox security extension is NoScript developed by Giorgio Maone. At its most basic, NoScript blocks Java, JavaScript, and Flash from running on untrusted pages. By default, almost all sites are configured as untrusted, allowing the user to select which, if any, domains are allowed to execute these types of client side code. There are many more security features beyond the simple, yet effective, default deny stance to client side code (Noscript).

Firekeeper – Firekeeper is an experimental Firefox security extension that seeks to serve as an intrusion detection system (IDS) for the browser. (FireKeeper).

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

RequestPolicy – RequestPolicy is an experimental Firefox Extension that configures the browser with a default deny posture with respect to cross site requests. This extension can provide significant protection against the exploitation of CSRF attacks. Some mistakenly believe that NoScript provides ample protection against CSRF attacks (RequestPolicy).

WOT – The WOT (Web of Trust) extension seeks to warn the end user about potential risky websites before allowing access. In addition the extension can integrate with common search engines and webmail, providing a visual indicator of the level of trust that other users have placed in the target site. WOT can prove especially helpful in protecting the user from phishing sites and drive-by malware sites (WOT: Web of Trust).

Enterprise Management

While extensions provide the most significant security advantage that Firefox wields over Internet Explorer, Firefox's lack of enterprise management represents the most significant disadvantage. Firefox does not natively integrate with Microsoft's Group Policy, and does not offer a compelling alternative means for the manageability of browsers once installed on end-users systems. Effectively, most enterprise deployments of Firefox rely on the users themselves for maintaining the patches, configuration, and extensions. This lack of central management presents an extreme disincentive for enterprise deployment, and significantly decreases the value of additional security that might be had through Firefox's extensions.

© SANS Institute 2009, Author retains full rights.

Browsers Reloaded

Since the browser is commonly targeted by attackers, let's consider methods to harden the browser and resist browser based attacks. Sandboxing and application virtualization are currently external to the browsers themselves, though these approaches may well be adopted as part of future browsers.[5] Some particular challenges associated with the security of browsers include patching, configuration, and maintenance of the browser as well as 3rd party ancillary applications. Further, the threat of drive-by malware installation, Cross Site Scripting attacks, and the impact of Cross Site Request Forgery attacks should also be a consideration when thinking on browser security.

3. Sandboxing

The technical concept known as a sandbox dates back quite some time. Although the term sandbox is used quite widely within computing to mean various things, the most well known example of of a sandbox is offered by Java Applets.[6] The concept of the sandbox security model is very clearly defined within early versions of Java, so we will take a moment and reflect on this notion since much of this paper will be spent explicating this idea with respect to browsers.

---

5 Google Chrome, though currently in beta, is suggested to be using isolation techniques from their acquisition of GreenBorder to achieve greater resilience in the browser (Methvin, 2008). While this technique is intended primarily for performance and better end user experience when using thicker and thicker web based applications, it will likely also be leveraged for security sandboxing type functionality.

6 Although, strictly speaking, the sandbox security model is employed more widely than just applets with versions starting with Java 2 (Oaks, 2009).

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

> This security model centers around the idea of a sandbox. The idea is when you allow a program to be hosted on your computer, you want to provide an environment where the program can play (i.e. run), but you want to confine the program's play area in certain bounds. You may decide to give the program certain toys to play with (i.e., you may decide to let it have access to certain system resources), but in general, make sure that the program is confined to its sandbox. (Oaks, p. 10, 2001)

From the above, we can see that the goal of the sandbox is twofold: first, we would like to allow the ostensibly untrusted code the ability to run; second, we would like to be certain that its running will not impact the rest of our systems. The reference above also hints at the idea of being able to offer the untrusted code extra “toys” with which to play. The possibility of allowing other, possibly trusted, code to play in the sandbox with the untrusted code is important to note, as it can allow for a richer experience when we turn our attention to the browser.

Though we have spent time detailing what is traditionally meant by the term sandbox with respect to information security, we are now going to discuss the efficacy of the sandbox notion as it applies to browser security. Let us set the stage. Imagine a piece of code that has become at once vital to the performance of the majority of job functions, and yet is also plagued with a rather constant barrage of attacks. Further imagine that the selfsame functionality required by the code to carry out its legitimate and well intentioned tasks is that

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

which is also most commonly abused by the attackers. That code is the browser.

Applications are being pushed onto a web server that we never even thought possible to be run in that environment. Part of the reason that novel applications are web enabled is due to the increasingly thick web technologies: AJAX; ActiveX; Flash; Silverlight; AIR; etc. More of our infrastructure is leveraging web front-ends for management and review. The browser is absolutely ubiquitous. With that ubiquity comes highly motivated attackers looking to exploit weaknesses in the same technologies that are being used by legitimate applications. This “double-edged sword” aspect of the browser's features being, at once, what both attackers and applications leverage, engenders the need for a different type of security than is typical of most other applications and systems. Certainly, a hardened browser configuration would be advantageous from a security perspective, but the hardening countermeasures most likely to disrupt attackers also frequently disrupt legitimate users.

What then does a sandbox offer to the browser from the vantage point of security? First and foremost, a sandbox can allow the browser limited access to critical system files, libraries, and binaries. Only the components and access necessary for the function of the browser can be offered for interaction. Thus, should a browser vulnerability be exploited, perhaps the impact of the threat could be more limited. Further, an additional aspect of a sandbox is that it can provide a facade of storage to the browser. As far as the browser is

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

concerned it is capable of writing permanent changes to hard disk. However, in reality, the sandbox can provide simply an abstracted virtual storage than can ultimately be discarded. This last point is especially important for browser security. Most browser based exploits attempt to make persistent changes to the underlying system so as to provide ongoing interaction with the attacker or solidify a piece of malware's foothold in the system. Although some attacks could still be devastating even if they do not have the ability to persist, many attacks and malware would be greatly limited if unable to make lasting changes to the underlying system.

3.1 Sandboxie

To illustrate both some of the security gains as well as the limitations of sandboxing the browser, we will turn our attention to Sandboxie. Though there are indeed other applications that effect some of the same functionality as is offered by this product, Sandboxie's low unit cost (less than $11/seat with as few as 100 seats) and robust feature set make it especially compelling (Sandboxie Online Store). Sandboxie's author is Ronen Tzu, who first developed the program in 2004 in response to his computer being infected with spyware (Gibson & Laporte, 2008). Though the program was initially designed specifically to run Internet Explorer in an isolated fashion, the functionality now has been abstracted to provide isolation for myriad programs and their interaction (Gibson & Laporte).

Before delving specifically into how Sandboxie serves as an salve for our browser security woes, let us attend to how Sandboxie functions more generally. The Frequently Asked

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

Questions page of the Sandboxie website provides a nice little analogy, which explains the basic functionality offered by this tool:

> Think of your PC as a piece of paper. Every program you run writes on the paper. When you run your browser, it writes on the paper about every site you visited. And any malware you come across will usually try to write itself into the paper.
>
> Traditional privacy and anti-malware software try to locate and erase any writings they think you wouldn't want on the paper. Most of the times they get it right. But first the makers of these solutions must teach the solution what to look for on the paper, and also how to erase it safely.
>
> On the other hand, the Sandboxie sandbox works like a transparency layer placed over the paper. Programs write on the transparency layer and to them it looks like the real paper. When you delete the sandbox, it's like removing the transparency layer, the unchanged, real paper is
>
> revealed. (Sandboxie - FAQ)

The paper analogy not only conveys Sandboxie's functionality, but also serves to differentiate its approach from more traditional antimalware solutions. Sandboxie serves as a prophylactic layer that blocks all permanent interaction with “the paper”, whereas most traditional antimalware solutions merely try to block specific “words” from being written to “the paper” (or, commonly, scan the words written for evidence of those bad “words”). Though the changes applied to this transparent prophylactic layer are typically discarded, they

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

need not be. With respect to the web browser, for simple browsing and web application usage, the read-only default should suffice, however, should a downloaded file or component be needed, recovery, is fairly straightforward, albeit manual.

Even with this simplistic understanding of the basic functionality provided by Sandboxie, we can already begin to see how this might serve to protect our users some from of the more nefarious threats to the browser. At its most basic, Sandboxie can provide protection against the threat of drive-by-downloads having a persistent impact. As soon as the infected sandboxed browser is terminated, the ongoing threat agent can be purged from the disks. Perhaps the most important security setting to be used with Sandboxie is the AutoDelete=yes in the sandbox.ini file. This will cause the purging of the sandbox to occur automatically upon exiting the sandboxed program.

One of the biggest challenges of Sandboxie in an enterprise is that in order for us to realize some of the benefits of its use requires us to delete the sandbox to ensure that any malicious changes will not persist. Since all downloaded files, writes to disk or registry, actually occur in the sandbox, files that a user intentionally downloaded will also be deleted when the sandbox is purged. Although this will necessarily require user awareness training, Sandboxie, through its Quick Recovery command, does attempt to make it easier for the end user to quickly recover changes intentionally made to, by default, the Desktop, Favorites, and My Documents folders (SandBoxie – Quick Recovery). Further, the AutoRecover=yes setting can be

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

configured in the sandbox.ini file to automatically move files from the sandbox to the physical system.

Sandboxie Settings:

This section will provide a quick list of important Sandboxie settings that can be configured in the sandboxie.ini file.

AutoDelete – This setting forces the contents of the sandbox to be automatically deleted upon exiting the sandboxed program. (Sandboxie – AutoDelete).

AutoRecover – Although it might seem counterintuitive, this setting can be used along with the AutoDelete setting. AutoRecover ensures that all files that would be found in the Quick Recovery are automatically moved from the sandbox to the physical disk. Granular control of which folder paths, file names, extensions are included or excluded from quick recovery.(Sandboxie – AutoRecover).

DropAdminRights – This setting causes any program running from the sandbox with administrative rights to have those privileges stripped from it. (Sandboxie – Drop Admin Rights).

ClosedFilePath – This setting allows for all access, including read, to be disallowed for specific file program configurations. Note: By default sandboxed applications can read anything the application would normally be able to read, but simply cannot make actual changes to the system. (Sandboxie – Closed File Path).

© SANS Institute 2009, Author retains full rights.

OpenFilePath – While typically all writes are denied for sandboxed applications, the OpenFilePath setting can allow certain paths to be exposed for actual permanent writes. (Sandboxie – Open File Path).

4. Application Virtualization

An alternative virtualization approach that could prove useful for browser security exists in application virtualization. As discussed above, one of the most formidable challenges presented by browser security are related to the ongoing maintenance of the browser configuration. Not only are we tasked with patching the browser itself, we must also be vigilant in the patching and secure configuration of all of the helper applications and extensions: Acrobat Reader; Flash; Quicktime; Windows Media Player; Webex; Skype; etc.;. Each of these applications that serve to extend the functionality of the browser also increase the attack surface area of the browser. While some enterprises seem capable of attempting to manage the configuration and patch level of Internet Explorer, most are rather incapable at even keeping all clients' Internet Explorer patched, let alone all of the plugins, and refuse to even try with any browser other than Internet Explorer. Enter application virtualization.

Although precisely defining a term that is commonly (ab)used by vendors can be difficult, a decent working definition is: **Application virtualization** separates a program from the underlying operating system and seeking to increase the portability, manageability and compatibility of applications (Dittner, Rule, 2007). Though virtualized applications are not

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

actually installed on the endpoint system, the application is still executed as if it were. For our purposes, the most salient aspect of the definition above is the “manageability” that application virtualization can afford us. The ostensibly unattainable goal of achieving a homogeneous and consistent browser deployment becomes possibly achievable. With application virtualization, one centrally located instance of the application in question, here, the browser, can be, not only, patched and hardened prior to deployment, but also maintained almost seamlessly throughout the applications life cycle. The technology that provides the basis for this functionality has existed for some time, but their utility for browser security still seems somewhat novel.

There are many and varied applications that provide this type of functionality: Citrix XenApp; Microsoft Application Virtualization (formerly SoftGrid); VMWare ThinApp (formerly Thinstall); etc.. Our attention will be focused on VMWare ThinApp. However, this selection is simply to focus attention on one product rather than suggesting its superiority when compared to other solutions.

4.1 ThinApp

Prior to being owned by VMWare, ThinApp was known as Thinstall, and was developed by a company of the same name. Soon after the January 2008 acquisition of Thinstall by VMWare, the product was rebranded as ThinApp. Though the name has changed, the solution remains largely unmodified from its initial incarnation. The ThinApp approach is a patently more enterprise oriented endeavor than Sandboxie. This overtly

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

enterprise approach comes with attendant costs and benefits. Perhaps the most important benefit for our purposes is ThinApp's capabilities for centralized control coupled with its native integration into Active Directory for access control. ThinApps can be configured such that they can only be run by members of specific Active Directory groups. The primary cost difference for some of these additional features is the required capital expenditure and ongoing support costs. At the time of this writing, the cost for the ThinApp suite with 50 client licenses and 1 year of 12x5 support is $6,050 USD . Each additional client with the same level of support is $47.19 USD. So an SME with 150 clients for 1 year of 12x5 support would run $10,769 USD.

ThinApp provides the “ability to deploy software without modifying the host computer or making any changes to the local operating system, file system or registry” (Introduction to VMWare ThinApp, 2009, p. 5). Although the security implications of the above description might seem obvious to those in the information security field, the boon to security does not seem to be one of the more widely touted features of these types of applications, ThinApp included. Portability and reduced testing costs for updated platforms seem to be the primary business justification for these types of applications. However, we will consider the resultant browser security advantages that this type of solution offers.

The choice of browser is a foregone conclusion for almost all enterprises. Regardless of the statistics suggesting that Firefox is becoming ever more popular, Internet Explorer

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

dominates the enterprise space. Even if an organization reached the conclusion that they wished to deploy Firefox, most would avoid doing so because of the lack of manageability. Further, even if an organization intended to offer a more secure browsing alternative by preinstalling Firefox, I would submit that, typically, they had done their organizations a disservice with respect to security due to having pushed a commonly exploited and unmanaged application.

Even if an organization decided that they could offer a more secure browsing platform with an alternative browser, the obstacles to managing that platform usually outweigh the security gains to be had. However, by leveraging a tool like ThinApp an organization could have simultaneously both more centralized and more granular control over the ongoing (re)configuration, maintenance, and patching of the browser and its ancillary applications.

ThinApp helps an organization achieve these lofty goals through application virtualization. We will continue to use the example of deploying an alternative browser platform, though another interesting use case is to install Internet Explorer 8 (Weigel, 2009). With ThinApp we can create a customized application package of Firefox that will be streamed from a central location by the clients. We can configure this package to include a hardened configuration, security oriented extensions, and fully patched versions of Flash, Quicktime, and other ancillary applications. Without application virtualization, if an organization decided to install Firefox they would likely preinstall this Firefox configuration to a

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

system prior to deployment, and then allow the user to keep the browser, extensions, and ancillary applications up to date, or else, try to push out updates as they get tested (hoping that they install successfully). For most organizations relying on end users to update their systems or "push and pray" patching is an untenable solution. With ThinApp style virtualization, the staff tasked with maintaining the Firefox configuration would simply update the centrally located Firefox ThinApp, and then end users would be running the updated version the next time they ran the browser or when the application synched, depending upon the deployment scenario (VMWare ThinApp's User Manual, 2009).

Fundamentally, ThinApp provides with a much easier method to manage the total browser: browser and 3rd party patches, configuration, and plugin/extension management. Although this is intended to be a general overview of using application virtualization for browser security, two technical configuration components bear further consideration. Though these settings are not fully exposed through the standard capture GUI that allows us to simply and quickly create a ThinApp package, they are still easy to configure. The two settings of interest are *DirectoryIsolationMode* and *RemoveSandboxOnExit,* and they work together. The ThinApp user manual explains that a *DirectoryIsolationMode* of Full works by, "blocking visibility to system elements outside the virtual application package," and restricting "any changes to files or registry keys to the sandbox" ensuring "that no interaction exists with the

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

environment outside the virtual application package" (VMWare ThinApp's User Manual, P. 26, 2009). Though we have not previously discussed the sandbox construct as it applies to ThinApp, it can be understood as,

> the directory where all changes that the captured application makes are stored. The next time you start the application, those changes are incorporated from the sandbox. When you delete the sandbox directory, the application reverts to its captured state. (VMWare ThinApp's User Manual, P. 127, 2009)

With this understanding of how the sandbox is considered within ThinApp, the *RemoveSandboxOnExit* setting becomes immediately obvious; it causes all contents of the sandbox to be purged when the ThinApp is exited. The setting chosen for the *DirectoryIsolationMode* dictates what information will be stored in the sandbox. If, as was suggested earlier, the DirectoryIsolationMode is set to Full, then all file and registry changes would be written to the sandbox rather than the physical. Additionally, if RemoveSandboxOnExit is turned on, then the application would effectively delete all of the file and registry changes that were written to the sandbox. These settings combined serve to ensure that the user cannot have any data persist from one running of the application to another, and allows for the intended configuration to remain intact regardless of the end users privileges on the system itself. One potential issue with configuring the browser with settings this stringent is the fact that the user will be unable to have persistent access to files downloaded from the internet.

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

The ease with which a ThinApp alternate browser package can be created and maintained significantly lessens a major obstacle precluding most organizations' deployment of an alternate browser. Alternative browser notwithstanding, the approach of having the browser be a centralized package that is abstracted from the target system can greatly increase the ability for an organization to maintain the security of the entire browser with all of its associated extensions, plugins, and ancillary applications.

5. Conclusions

The threat and vulnerability landscape associated with the browser has been trending towards significantly increased risk. Concomitant with the increased risk of running the browser is a deluge of web enabled applications many of which require software that extends browser functionality. The juxtaposition of both this increased risk and increased use of the browser for “thicker” applications creates a security quagmire. Simultaneously securing the browser, and still allowing the necessary functionality, presents one of the most difficult tasks within information security.

A better browser could rectify many of the security challenges which currently plague us. However, perpetually waiting for the browser to “get fixed” is typically an untenable solution for most enterprises. Virtualization techniques offer a relatively novel approach to achieving greater security of the browser. This paper reviewed two distinct virtualization approaches and an associated product. Sandboxing was reviewed through the product Sandboxie, while we delved into application

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

virtualization through ThinApp. Sandboxing/Sandboxie offers us the ability to prevent browser-borne attacks/malware from affecting permanent changes to the underlying registry or file system. Preventing permanent damage or compromise due to drive-by-malware is very significant, as this threat has been rapidly increasing. In VMWare's ThinApp we found a means for more nimbly managing the entire browser landscape including the patching and configuration of both the browser and the associated plugins/extensions.

While merely purchasing and deploying these technologies is not sufficient for browser security, they each provide tools that, when properly leveraged, can help to increase an organization's browser security posture. At the very least most organizations need to take seriously the risk posed to their organizations by typical browser configurations and review what tools and techniques might be leveraged to better protect the browser landscape.

© SANS Institute 2009, Author retains full rights.

6. References

Browser Market Share. (n.d.) Retrieved January 3, 2009, from Market Share by Net Applications Web site: http://marketshare.hitslink.com/browser-market-share.aspx?qprid=0#

Buy VMWare ThinApp 4 Client License (n.d.). Retrieved May 24, 2009, from http://store.vmware.com/store/vmware/en_US/DisplayProductDetailsPage/productID.105855800

Buy VMWare ThinApp 4 Suite (n.d.). Retrieved May 24, 2009, from http://store.vmware.com/store/vmware/en_US/DisplayProductDetailsPage/productID.105855000

Costoya, J. (2006, March 2). Malicious Firefox Extension: Firestarter. Weblog retrieved from http://blog.trendmicro.com/malicious-firefox-extensions/

Cross, M., Kapinos, S., Meer, H., Muttik, I., Palmer, S., & Petko, P. (2007). *Web Application Vulnerabilities: Detect, Exploit, Prevent.* Burlington: Syngress.

Dittner, R., & Rule, D. (2007). *The Best Damn Server Virtualization Book Period: Including Vmware, Xen, and Microsoft Virtual Server*. Burlington: Syngress.

Firekeeper. (n.d.). Firekeeper – detect and block malicious sites. Website retrieved from http://firekeeper.mozdev.org/

Gibson, S., & Laporte, L. (2008, November 27). Security Now! Episode 172: Sandboxie. Podcast retrieved from http://media.GRC.com/sn/SN-172.mp3

© SANS Institute 2009, Author retains full rights.

Grossman, J., & Hansen, R., & Petkov, P.D., & Rager, A. (2007). *XSS Attacks*. Burlington: Syngress.

WOT: Web of Trust. (n.d.) Retrieved May 23, 2009, from Web of Trust website: http://www.mywot.com/

Introduction to VMWare ThinApp, version 4.0. (2009, March 24). Retrieved May 23, 2009, from VMWare ThinApp Documentation web site: http://www.vmware.com/support/pubs/thinapp_pubs.html

Methvin, D (2008, September 1). Google Chrome Answers The GreenBorder Mystery. Retrieved 7:22, March 14, 2009, from InformationWeek's Microsoft Weblog: http://www.informationweek.com/blog/main/archives/2008/09/google_chrome_a.html

NoScript. (n.d.) NoScript – JavaScript/Java/Flash Blocker for a safer Firefox experience. Website retrieved May 23, 2009, from http://noscript.net/

Oaks, S (2001). *Java Security, 2nd Edition*. Sebastapol: O'Reilly.

Provos, N., & Mavrommatis, P., & Abu Rajab, M., & Monrose, F. (2008, July) All Your iFRAMES Point to Us. *Proceedings of the 17th USENIX Security Symposium,* from http://www.usenix.org/events/sec08/tech/full_papers/provos/provos.pdf

RequestPolicy. (n.d.) RequestPolicy – Firefox addon for privacy and security. Website retrieved May 23, 2009, from http://www.requestpolicy.com/

© SANS Institute 2009, Author retains full rights.

Sandboxie – AutoDelet. (n.d.) Retrieved 06:58, May 25, 2009, from http://www.sandboxie.com/index.php?AutoDelete

Sandboxie – AutoRecover. (n.d.) Retrieved 06:58, May 25, 2009, from http://www.sandboxie.com/index.php?AutoRecover

Sandboxie – Closed File Path. (n.d.) Retrieved 06:58, May 25, 2009, from http://www.sandboxie.com/index.php?ClosedFilePath

Sandboxie – Drop Admin Rights. (n.d.) Retrieved 06:58, May 25, 2009, from http://www.sandboxie.com/index.php?DropAdminRights

Sandboxie – FAQ. (n.d.) Retrieved 06:58, May 25, 2009, from http://www.sandboxie.com/index.php?FrequentlyAskedQuestions

Sandboxie Online Store. (n.d.) Retrieved 09:27, March 14, 2009, from https://www.cleverbridge.com/296/?scope=checkout&cart=29388,29389,29390&cb_ident=655eaa38

Sandboxie – Open File Path. (n.d.) Retrieved 06:58, May 25, 2009, from http://www.sandboxie.com/index.php?OpenFilePath

Sandboxie – Quick Recovery. (n.d.) Retrieved 06:58, May 25, 2009, from http://www.sandboxie.com/index.php?QuickRecovery

Stuttard, D., & Pinto, M., (2007). *Web Application Hackers Handbook*. Indianapolis: Wiley.

Top Browser Share Trend. (n.d.) Retrieved January 3, 2009, from Market Share by Net Applications Web site: http://marketshare.hitslink.com/browser-market-share.aspx?qprid=1

Turner, D (Ed.) (2008, April). Executive Summary. *Symantec Internet Security Threat Report*, *XIII*, from http://eval.symantec.com/mktginfo/enterprise/white_papers/b-

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

whitepaper_exec_summary_internet_security_threat_report_xiii_04-2008.en-us.pdf

Turner, D (Ed.) (2008, April). Symantec Global Internet Security Threat Report. *Symantec Internet Security Threat Report*, *XIII*, from http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_internet_security_threat_report_xiii_04-2008.en-us.pdf

VMWare ThinApp's User Guide (2009, March). Retrieved May 24, 2009, from http://www.vmware.com/pdf/thinapp402_manual.pdf

Vulnerability Report: Microsoft Internet Explorer 7.x. (n.d) Retrieved January 10, 2009, from Secunia Advisories website: http://secunia.com/advisories/product/12366/?task=statistics

Vulnerability Report: Mozilla Firefox 3.x. (n.d) Retrieved January 10, 2009, from Secunia Advisories website: http://secunia.com/advisories/product/19089/?task=statistics

Weigel, Paul (2009). How to ThinApp Internet Explorer 8.0. Retrieved May 23, 2009, from Virtualization Consultant blog: http://virtualizationconsultant.blogspot.com/2009/04/how-to-thinapp-internet-explorer-80.html

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

Last Updated: August 29th, 2015

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

FOR578 Cyber Threat Intelligence	**Tysons Corner, VAUS**	**Aug 31, 2015 - Sep 04, 2015**	**Live Event**
SANS Milan 2015	**Milan, IT**	**Sep 07, 2015 - Sep 12, 2015**	**Live Event**
SANS Crystal City 2015	**Crystal City, VAUS**	**Sep 08, 2015 - Sep 13, 2015**	**Live Event**
SANS Network Security 2015	**Las Vegas, NVUS**	**Sep 12, 2015 - Sep 21, 2015**	**Live Event**
SANS Seoul 2015	**Seoul, KR**	**Sep 14, 2015 - Sep 19, 2015**	**Live Event**
SANS Tallinn 2015	**Tallinn, EE**	**Sep 21, 2015 - Sep 26, 2015**	**Live Event**
SANS Perth 2015	**Perth, AU**	**Sep 21, 2015 - Sep 26, 2015**	**Live Event**
SANS Baltimore 2015	**Baltimore, MDUS**	**Sep 21, 2015 - Sep 26, 2015**	**Live Event**
Data Breach Investigation Summit & Training	**Dallas, TXUS**	**Sep 21, 2015 - Sep 26, 2015**	**Live Event**
SANS ICS Amsterdam 2015	**Amsterdam, NL**	**Sep 22, 2015 - Sep 28, 2015**	**Live Event**
SANS Bangalore 2015	**Bangalore, IN**	**Sep 28, 2015 - Oct 17, 2015**	**Live Event**
SANS DFIR Prague 2015	**Prague, CZ**	**Oct 05, 2015 - Oct 17, 2015**	**Live Event**
SANS Seattle 2015	**Seattle, WAUS**	**Oct 05, 2015 - Oct 10, 2015**	**Live Event**
SOS: SANS October Singapore 2015	**Singapore, SG**	**Oct 12, 2015 - Oct 24, 2015**	**Live Event**
SANS Tysons Corner 2015	**Tysons Corner, VAUS**	**Oct 12, 2015 - Oct 17, 2015**	**Live Event**
SANS Gulf Region 2015	**Dubai, AE**	**Oct 17, 2015 - Oct 29, 2015**	**Live Event**
SANS Cyber Defense San Diego 2015	**San Diego, CAUS**	**Oct 19, 2015 - Oct 24, 2015**	**Live Event**
SANS Tokyo Autumn 2015	**Tokyo, JP**	**Oct 19, 2015 - Oct 31, 2015**	**Live Event**
SANS South Florida 2015	**Fort Lauderdale, FLUS**	**Nov 09, 2015 - Nov 14, 2015**	**Live Event**
SANS Sydney 2015	**Sydney, AU**	**Nov 09, 2015 - Nov 21, 2015**	**Live Event**
SANS London 2015	**London, GB**	**Nov 14, 2015 - Nov 23, 2015**	**Live Event**
Pen Test Hackfest Summit & Training	**Alexandria, VAUS**	**Nov 16, 2015 - Nov 23, 2015**	**Live Event**
SANS Hyderabad 2015	**Hyderabad, IN**	**Nov 24, 2015 - Dec 04, 2015**	**Live Event**
SANS Chicago 2015	**OnlineILUS**	**Aug 30, 2015 - Sep 04, 2015**	**Live Event**
SANS OnDemand	**Books & MP3s OnlyUS**	**Anytime**	**Self Paced**