Sponsored by Adobe Systems

Security of Applications: It Takes a Village

June 2011

A SANS Whitepaper

Written by: *Dave Shackleford*

Introduction

The attack landscape has shifted over the past 10 years from attacks on servers to attacks on operating systems (particularly Windows, but also UNIX, Linux, and increasingly, Mac O/S). Over the past five years, attackers have targeted popular, third-party applications that interact directly with the web, including all major browsers and browser plug-ins such as Adobe Reader, Flash Player, QuickTime, and other browser applications.

Client-side applications—ranging from browsers to multimedia programs to document readers—are widely used in organizations and by individuals for interfacing with the web, gaming, social networking, and other online media types. In addition, more enterprise users are accessing critical applications and social media through their browsers, and a significant amount of business is done via email.

Attackers are leveraging these client-side vectors, hitting first with social engineering attacks to get victims to download malware, get in the middle of secure browser sessions, or phish their credentials directly through their browsers. When consumers click a link or media file on their social networks or elsewhere, or when they land on a malicious site or search engine link, their browsers and browser plugins are immediately scanned for vulnerabilities to exploit.

Being a client-side application vendor in this landscape isn't easy. Sometimes serious vulnerabilities need to be repaired quickly. Yet, making these repairs means pushing out patches to what could be millions of highly-distributed users running a number of versions of their applications that also interact with other applications. When vendors create patches or make software improvements, they must consider any variety of enterprise environments that include multiple desktop images, elevated desktop user privileges, and numerous applications installed on each individual system. All of these conditions make the process of software patching and upgrading extremely difficult.

Operating system vendors who've gone before client-side application vendors have developed processes for patch management that most consumers and consumer organizations have accepted and grown used to. Today, client-side application vendors are learning to make updates and patch management part of the users' routine. Some client-side application vendors are also improving security for their users by using sandboxing techniques. Sandboxing opens executable files in a protected environment, preventing any potentially malicious code from ever directly interacting with other applications.

Vendors have grown and improved in their security processes, which will be discussed in this paper. However, vendors aren't alone in these efforts. Organizations using client-side software have their own roles to play in protecting their applications. For example, upgrading to the most secure versions of applications and standardizing on those versions as much as possible reduces internal risk and helps organizations streamline their patching and upgrade processes.

This paper discusses the role of vendors and consumers in protecting against client-side application attacks. It also includes a simple roadmap that enterprises can use to improve the state of client-side software throughout their environments.

What's Changed: The Client-Side Threat Landscape

Back in 2005, Core Security Technologies reported that client-side vulnerabilities were starting to outpace server-side and operating system vulnerabilities.[1] Since then, the focus on client-side applications and their users has continued to intensify.

Client-side applications in contact with the Internet are under full-on attack. Attackers will keep probing at them until they find a weak spot, which is often caused by lack of patching. Yet, unpatched client-side applications were the "primary initial infection vector used to compromise computers that have Internet access," according to the SANS "Top Cyber Security Risks" report released in September 2009.[2]

The report identified missing patches and attacks on client-side software as the top priority for security practitioners. However, even patched systems can be vulnerable because client-side zero-day attacks are also becoming more common. A zero-day attack is one that contains exploit code that is used in the wild to affect systems before vendors have advance notice to develop patches. SANS, in its September 2009 report, cited multiple zero-day vulnerabilities affecting Adobe and Microsoft software. In the "2011 Global Security Report" from Trustwave, client-side attacks combined with social engineering attacks were the second most prevalent attack vector for data breaches. Combining social engineering attacks such as phishing with client-side attacks via malicious PDF documents and other attachments were commonly used tactics, according to the report.[3] In addition, the Sophos "Security Threat Report for 2011" also chronicled numerous zero-day vulnerabilities targeting popular Adobe and Sun Java software.[4]

Many different types of software representing a wide variety of vendors are now primary targets for attackers trying to gain a foothold into organizations through their client-side applications. According to the TippingPoint Zero Day Initiative (which chronicles vulnerabilities discovered by researchers but not yet released) the most common software vendors targeted include Apple, Adobe, Microsoft, Mozilla, and several others.[5] The SANS "Top Cyber Security Risks" report identified these same applications as most targeted back in 2009. With the widespread install base of software from these providers, it is no surprise they are the leading targets.

1 www.coresecurity.com/content/client-side-exploits

2 www.sans.org/top-cyber-security-risks/summary.php

3 www.trustwave.com/downloads/Trustwave_WP_Global_Security_Report_2011.pdf

4 www.sophos.com/en-us/security-news-trends/whitepapers/gated-wp/sophos-security-threat-report-2011-wpna.aspx

5 www.zerodayinitiative.com/advisories/upcoming/

Here's how attacks on these types of client-side applications usually unfold:

1. An attacker creates a simple client-side exploit package by generating a Metasploit[6] reverse shell or Meterpreter executable and packaging it in a PDF or Microsoft Excel file.

2. Using a custom phishing email generated manually or with tools like the Social Engineering Toolkit,[7] the attacker sends an email to the target user that includes the Excel or PDF file as an attachment.

3. The victim executes the attachment, causing the Metasploit module to launch in the background, most likely unbeknownst to the user. At this point, the shell is sent outbound, where it connects to a listener set up on the attacker's system. Now, the attacker is free to execute commands on the victim's system.

Attacks just like this one were chronicled in numerous case studies released by incident response consulting firms such as HBGary and Mandiant. These cases illustrate advanced data breaches incorporating these same client-side attacks. Some of the more notorious examples, such as Operation Aurora, where Google and other large software vendors were impacted by the breach, incorporate all the major elements of a sophisticated, well-planned attack.[8] In most of these cases, victims were specifically targeted using phishing emails that contained attached documents or were directed to websites with embedded malicious content that exploited browser flaws.

All of these scenarios include some type of client-side software vulnerability as a major initial vector of attack. The Mandiant M-Trends whitepapers on advanced persistent threat (APT) describe multiple cases where defense contractors, law firms, manufacturing companies, and other types of businesses (including government organizations) are targeted in this same way.[9] Although each case varies somewhat, the attackers consistently (and successfully) bet on the most common local software variants being installed, including document readers, Microsoft Office products, browsers, and multimedia players.

6 www.metasploit.org/

7 www.social-engineer.org/framework/Computer_Based_Social_Engineering_Tools:_Social_Engineer_Toolkit_(SET)

8 www.hbgary.com/hbgary-threat-report-operation-aurora

9 www.mandiant.com/products/services/m-trends

The reason third-party software is so difficult to secure involves both the vendors of the applications and their users.

Vendor Challenges

For vendors of client-side software, improving security of their applications presents many challenges. The first problem stems from the number of different operating system platforms (and versions of those platforms) they must support. Unlike server-specific software that addresses enterprise-level services and provides robust capabilities for multiples users simultaneously, client-side software is typically installed on a vast number of disconnected systems that may represent a mix of different operating system (OS) versions and types, such as Windows XP, Windows Vista, Windows 7, and Windows Server 2003 and 2008. There may also be a variety of Mac OS X and Linux distributions mixed into the environment.

So in the example of a Windows version upgrade, numerous dynamic linked libraries (DLLs) and software components may change, requiring application changes that range from minor to significant. Completely different software architectures are likely needed for Mac OS X and Linux platforms, adding to the overall complexity of development and release cycles.

Organizations may be using multiple versions of software for different platforms, and there may be a legitimate business need to continue using older versions of software in some cases. In-house applications have been written to integrate directly with specific versions of third party software, and upgrading the vendor software may require that the application to be changed significantly. In the case of client-side software, where systems even include small mobile devices, this level of software and system support is a significant challenge for vendors who don't know the device, system and application configurations their clients will be patching or upgrading to.

A good example of this scenario is client-side software that must integrate with different browsers across multiple platforms. Adobe Flash Player, for instance, must integrate with Internet Explorer, Firefox, Safari, and other browsers, providing equivalent functionality across each browser for different versions. This complexity is illustrated in Figure 1.

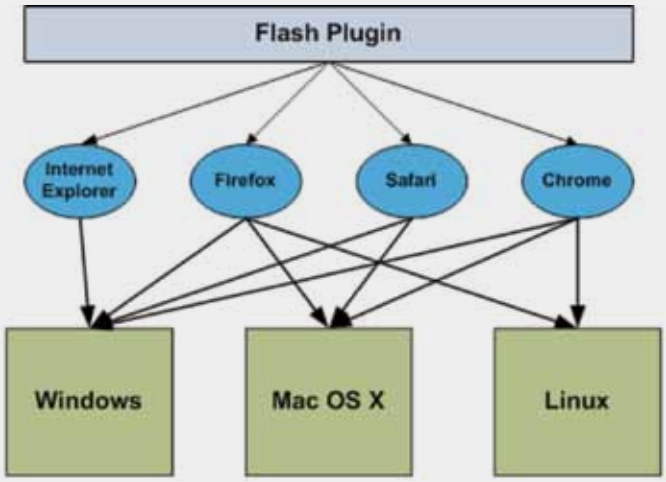

Figure 1: Client-side software complexity

The diagram in Figure 1 is greatly simplified and does not represent the multiple versions of these (and other) applications and plugins along the chain. Yet even in its simplicity, it illustrates the often-overlooked discussion around the customer responsibility to regularly update and maintain all client-side software.

User Challenges

Although code security and quality falls definitively on the shoulders of the vendor (and no one is suggesting otherwise), there are actually quite a few elements to a robust data and system protection program that enterprises must pay careful attention to in order to manage their client-side applications securely. For example, their failure to standardize on versions and even applications where possible hampers the organization's ability to accept changes and upgrades. By failing to standardize and upgrade to the latest versions, organizations are failing to make use of the built-in security hardening included in the product upgrade.

In addition, client-side software is often overlooked in patch cycles. This is due to many reasons, including delays in patch notification from vendors, lack of operational and testing capacity during the existing cycle, or simply forgetting to include client-side software in their patch management cycles.

Client-side software must now be considered during traditional patching cycles. For example, Microsoft releases patches on a monthly basis with their Patch Tuesday program. It is anticipated that large, client-side software vendors will follow suit or at least make patches widely available and known as soon as they have them publicly available. Aside from occasional out-of-band patches, system administrators can plan for testing, deployment, and change management to manage the patch process by knowing on which third party applications they should check for updates and where to find those updates.

Most patch management and deployment packages do not include third-party applications (as is done by default for multiple OS platforms). In the case of client-side applications, third-party configuration and patch management vendors in this space (such as Lumension, Shavlik, BMC, and others) all support most popular client-side applications; however, coverage of Linux and Mac operating systems is lacking in most cases.

Software provisioning tools often support upgrades and patches to third-party applications, but they do not automatically notify administrators and other IT staff when these updates are available. In fact, the nature of client-side software itself usually implies a standalone installer and/or update mechanism. Adobe and some other vendors are changing this by moving to a more traditional patching and update release model (much like Microsoft's). This, however, will take some time to become widespread.

A final point to consider with regard to enterprise patching and updating of third-party client software concerns the issue of control and flexibility in updating endpoint systems. In many cases, enterprises may have numerous separate images with widely varying users (with different privilege levels), a large number of different client applications and versions, and less-than-ideal configuration management and software inventory practices. For organizations with distributed IT teams managing groups of systems—or business unit IT groups that are decentralized—creating consistent policies and configuration practices for installing and updating client applications can also be a challenge.

Fortunately, popular client-side software vendors are taking the threats to their products seriously, as Trustwave noted in their "2011 Global Security Report." The following are some examples.

Incentivizing Security Researchers

Google is offering up to $3,133.70 for reporting serious security flaws in their Chrome browser, upping the amount after Mozilla increased its "bug bounty" from a maximum of $500 to $3000 in mid2010.[10] Many vendors have also created product security response teams that actively work with researchers and coordinate between them and the development organization. Vendors such as Apple, Adobe, Microsoft, Mozilla, and Google's response teams also interface with the security community and larger incident response groups such as CERT/CC[11], FIRST[12], and others. This, in turn, has allowed vendors to improve their overall software development lifecycles to more rapidly patch and push out fixes for bugs as they come in and to safely announce the vulnerabilities via a trusted disclosure channel.

Vendor Collaboration

The sharing of knowledge is critically important to the overall software ecosystem, because many vendors have integrated products together and rely on one another for secure operation of client systems. One example of a comprehensive program is Microsoft's Security Ecosystem Collaboration, which describes the lifecycle of how Microsoft works with partners, public organizations, customers, security industry organizations, and security researchers to create a closed loop of collaborative effort to improve software security.

A key component of this program is the Microsoft Active Protections Program (MAPP), which provides Microsoft bug information to key partners and security vendors in advance of official patch releases and announcements, giving them time to incorporate changes into their own software.[13] Many leading client-side application vendors subscribe to this service because it provides a place where they can share and gather bug information that helps all vendors to coordinate release and patching cycles more effectively, which can benefit end-user organizations.

10 http://code.google.com/p/chromium/issues/list

11 www.cert.org/certcc.html

12 www.first.org

13 www.microsoft.com/security/msrc/collaboration/mapp.aspx

Improvement in Client-side Software Security

A number of new client-side software features are emerging that will help mitigate attacks that utilize endpoint software as a means of compromising systems and infecting them with malware. Here are some examples:

- The first method of protection is known as sandboxing, which severely restricts the rights of the executing program and runs the program in a restricted environment on the operating system. Without this sandboxing technique applied, software and browser plugins may be able to write files to the hard drive, manipulate or create registry entries on Windows systems, launch new processes, or interact with the OS kernel in a number of ways. With a sandbox approach, software can only execute in a very limited context, and it is usually prohibited from writing to the system's hard drive or interacting with other programs. Sandboxing has helped block many attack types today, but it is only one security mechanism to which client-side vendors are turning.

- Another security measure many vendors are taking involves mitigating well-known buffer overflow and memory manipulation attacks with efforts such as Windows Address Space Layout Randomization (ASLR) and Data Execution Prevention (DEP). These techniques make it much harder for attackers to guess where to return their exploit code within the memory of a target system (a key element of creating and executing buffer overflows) by increasing the overall address space used by applications and marking memory as nonexecutable.

- Vendors are also exposing their code security processes. Vendors are improving security both internal to their applications and from the user perspective, as well. Many vendors are exposing details of their internal code review process (using techniques like static code analysis) as well as the overall software development lifecycle (SDLC). They are also creating new features like JavaScript Blacklists in the software and generating user alert mechanisms for enhanced security awareness. These features are also helping to improve the interactions between the user, the software, and the client system itself.

Implementation of these techniques varies markedly from one platform to another, and some researchers have identified ways to bypass these protections in certain cases. Overall, though, they can help create a more effective defense-in-depth security posture for systems, and many application vendors are taking advantage of these capabilities for runtime execution of their software.

As software vendors strive to make ongoing security, process and platform improvements, organizations using commercial software have their responsibilities, too. Primarily, they must have established policies and processes to improve their endpoint security and reduce the vulnerability surface attackers can use against them.

The SANS "20 Critical Controls Project," a subset of NIST 800-53, provides a baseline of fundamental security best practices that range from operational to very focused and targeted controls. Several of the critical controls are reflected in this simple approach for enterprises to improve client software security, including:

- Critical Control 2: Inventory of Authorized and Unauthorized Software[14]
- Critical Control 3: Secure Configurations for Hardware and Software on Laptops, Workstations, and Servers[15]
- Critical Control 7: Application Software Security[16]

Take Inventory

The first key step in improving the security of client-side software is to generate a detailed software inventory for all desktop, laptop, and server images and builds. This can be accomplished in a number of ways, but the most logical process for many larger organizations is to implement an agent-based system configuration and monitoring solution. Examples of tools in this category include Microsoft System Center Configuration Manager (SCCM), Novell ZENworks Asset Management (now part of Attachmate), Symantec Altiris, and BMC Asset Management. Free tools include Microsoft's Assessment and Planning (MAP) Toolkit and the Spiceworks Software Inventory and Audit Tool.

Inventorying software can be a significant project, especially if multiple platform images exist and are distributed throughout a geographically dispersed enterprise. Additional challenges include gathering cross-platform software inventory data, centralizing this information, and ensuring that all system types were represented.

14 www.sans.org/cag/control/2.php

15 www.sans.org/cag/control/3.php

16 www.sans.org/cag/control/7.php

Standardize Where Possible

The next step in taming client-side software security is to standardize versions of software deployed when feasible. In many cases, administrators are surprised at what version creep they find during their assessments! Often, legacy and older platforms have entirely unsupported versions of software installed, and newer platforms have a mix of applications running on them, some of which may not be supported. Enterprises should leverage this assessment information to propose upgrade plans where possible, or introduce additional protective controls for legacy platforms that must remain in place for some time.

In addition, one or more versions may be installed with standard system image builds, but these tend to shift and drift over time as individual users upgrade or make changes to the original version installed. It's entirely possible that enterprises might need to support and maintain several versions of client-side software in order to ensure compatibility with all endpoint operating systems and successfully integrate with other applications. However, the fewer versions installed the better. Importantly, all versions should be supported by the vendor and have a reasonable lifespan left in them. In addition, organizations should standardize the individual applications' security controls for implementation within system builds. These controls can then be managed as part of a larger configuration management process.

Upgrade to New Versions

Unfortunately, organizations leave multiple versions of client-side software on their endpoint systems. Some of those versions may no longer be supported by the application vendor. Other versions, while still supported, contain gaping security holes that may be patched but not repaired (or in the worst case, may not be patched at all).

Upgrading to the latest version of client-side applications should be part of the organization's secure, endpoint configuration process recommended in SANS Critical Control 3. Gold build images for physical and virtual endpoints should be inclusive of client-side software upgrades, which not only improves overall security in these applications, but can also improve performance and integration issues addressed by the vendor in its latest version. By knowing their client-side applications and their vendors' upgrade processes, organizations can incorporate client-side application upgrades into their gold build upgrade processes.

Have a Patch Process

Better and more proactive patch assessment is the third critical element of a better client application maintenance program. Organizations should first determine whether their existing patch management software supports patching for third-party programs. If so, this is likely the simplest approach to integrating better and more consistent patching for client-side software. If the patch management program does not support third-party applications, organizations should explore the various options made available by the vendors, themselves, to see what might work best. In general, the simpler and more easily deployed the patching tools are, the more likely they are to be used quickly and consistently.

Patches for client applications should also be tested regularly and deployed as soon as possible, just like critical operating system patches. This process should include a test environment that mimics current use scenarios to the best possible extent. Integrating these patches into the overall patch management process will increase the time needed for patch testing, but in most cases, not enough to warrant avoiding it altogether.

Work with Vendors

Application vendors like Adobe, Apple, Mozilla and Microsoft need feedback from users and IT groups, particularly in the areas of patching and security. Helping the vendors understand security concerns, interoperability and compatibility issues, and operational challenges with their products can help them to improve the software over time. Vendors should make submission of this information simple and painless. Organizations need to leverage this to help improve product security, especially when the products are required or important for day-to-day business needs.

Conclusion

The leading cybercriminal attack vector has shifted away from attacks on servers to attacks on client-side applications, most commonly through web-interfacing applications.

By following a simple process of discovery, configuration management, standardization, upgrading and patching, organizations can minimize their exposure to attacks on well-known applications. In addition, by instituting new security features from vendors, enterprises can start to combat the growing threats to client-side applications that are constantly under attack.

Application vendors are aware of the issues and are working hard to improve their software, both internally and by working with security researchers and the community in general. By submitting bugs and providing feedback on features and tools that need to more effectively secure their systems, the entire security community can aid in the effort to improve client-side applications so they can stand up to attack.

About the Author

Dave Shackleford, founder and principal consultant with Voodoo Security, is a SANS analyst, instructor, and course author, as well as a GIAC technical director. He has consulted with hundreds of organizations in the areas of security, regulatory compliance, and network architecture and engineering. He has previously worked as CSO for Configuresoft, CTO for the Center for Internet Security, and as a security architect, analyst, and manager for several Fortune 500 companies. Dave is the co-author of "Hands-On Information Security" from Course Technology as well as the "Managing Incident Response" chapter in the Course Technology book "Readings and Cases in the Management of Information Security." Recently, Dave co-authored the first published course on virtualization security for the SANS Institute. He currently serves on the board of directors at the Technology Association of Georgia's Information Security Society and the SANS Technology Institute.

SANS would like to thank its sponsor: