SANS ANALYST PROGRAM

*Sponsored by McAfee*

# Application Whitelisting: Enhancing Host Security

**A SANS Whitepaper – October 2009**

*Written by Dave Shackleford*



# Introduction

Today's host-based security tools are facing an unprecedented number of issues, stemming largely from the increasing complexity and volume of enterprise applications, and threats to these applications. For more than a decade, organizations have applied blacklisting—blocking and alerting applications and behaviors based on signatures and heuristics—to protect their networks and, later, their end point systems.

While still a powerful and highly useful technology, blacklisting is up against rapidly reproducing malware variants and malicious behaviors. For this reason, many organizations are reinforcing blacklisting with real-time analysis techniques that incorporate behavior, reputation and threat correlation. However, for malware such as the Storm and Conficker worms, which use adaptive signature-morphing methods to outpace the end point's ability to adapt, lighter and more agile prevention techniques are essential. Agility is particularly important for dedicated systems and devices, such as remote point-of-sale terminals, which can't carry big signature files nor be online to check for signature updates.

For these reasons, whitelisting—accepting only applications and behaviors that are on the approved list and denying everything else—provides a lighter means to protect end points. If implemented correctly with gold builds and flexible updating capabilities, whitelisting can be particularly useful for securing legacy applications and systems, as well as embedded systems and kiosks. It is also a helpful addition for any robust end point security plan, as it can reduce false positives (and negatives), as well as work with preventative measures to block malicious code from installing.

# Blacklists and Signatures

The most prevalent category of host-based security is malware prevention, comprising a broad group of agent-based solutions that look for particular signatures and behavioral signs of malicious code execution. This approach, known as blacklisting, focuses on matching specific aspects of application code and particular actions being attempted by applications (behavior monitoring or heuristics) for detection.

Signature-based/blacklisting detection has been around for more than 15 years. In that same time, viruses, worms, sniffers, Trojans, bots and other forms of malware have infiltrated e-mail, IM, and later, social networking sites for the purpose of criminal financial gain. With improvements in correlation and centralized management, blacklisting still works effectively in most distributed enterprise and SMB environments. However, because these signature-based models depend on advanced knowledge of malicious code and behaviors, some instances can be missed, leading to potential malicious execution.

There are other instances in which trimmed-down dedicated systems can't get updates and host signature files. For each new variant of malware, signature-based systems require some sort of update to protect against that new variant. Variants hit the half million mark in 2007, according to reports by McAfee and F-Secure.[1] As of July this year, Fortinet was clocking 5,000 unique variants a month.[2]

Policy enforcement on installed applications is another area end point security still needs to address with better clarity. As of September 2009, eight of the 20 major categories in the SANS Top 20 Threat Report were related directly to vulnerabilities in or misuse of installed software applications, including instant messaging and peer-to-peer programs, Web browsers, Office applications and e-mail clients, media players, and even anti-virus software![3]

[1] www.securityfocus.com/brief/655
[2] www.fortiguard.com/report/roundup_august_2009.html
[3] www.sans.org/top20/

# Application Whitelisting

Application whitelisting can be defined in a number of ways. In technical terms, it refers to software that resides on a host computer and maintains a local "fingerprint" of applications that are allowed based on policies tied to users, groups, systems, and other potential attributes. It then it denies everything else.

By defining the allowed applications on a system, then restricting all other applications and programs from running, the tried-and-true information security practice of "Deny All" is now effectively brought to the local host. For many years, this has been a concept instituted to great success on network devices using firewall rules and access control lists (ACLs).

With proper integration into end point systems, an application whitelist can provide valuable correlating, reporting and alerting capabilities to existing protective technologies. Standing alone, whitelisting systems can also provide security for the dedicated end point when update and patching cycles may not be ideal.

There are a number of different whitelisting methodologies, but they are generally based on some common means of identifying application components and tying them to whitelisting policies, including:

- **Certificates:** Also known as "signing," certificates are created to certify that the applications come from a trusted source. Applications that have been digitally signed by a software vendor's trusted certificate can be assessed by the operating platform and the whitelisting software. Many whitelisting tools refer back to the central server for digitally signed applications because they typically have a higher degree of trust associated with them.
- **Path values:** The software path, or location on the system, is another attribute commonly used to fingerprint and identify software. In this case, whitelisting software specifies a path. Anything outside of that path is bad. This could include standard file and directory hierarchies, as found on all Windows and UNIX-based platforms, or specific types of entries found only in the Windows registries or the /proc file system on UNIX or Linux.

- **Hash values:** A cryptographic hash can be created for a file or groups of files affiliated with an application using commonly accepted protocols like MD5 or SHA-1. Some of these hashes could be pre-generated by a vendor and available publicly, such as those distributed by Sun Microsystems for Solaris application executables. Others can be generated by whitelisting software at the time of policy generation and enforcement. These hashes are periodically compared to new hashes generated "on the fly" to ensure the software is the same and has not changed.
- **Services:** Many applications have installed or affiliated services associated with them. Such services might be represented in the Services Management Console on Windows and the **`/etc/xinetd`** folder on Linux, for example. States are defined per policy, and then the software assesses, identifies and compares state at a given time.
- **System and user behavior:** Users define specific user and system behavior sequences that the whitelisting program allows. Examples might include TCP ports in use by the software, access logs generated by specific users, and times of day for access. These are most often used in conjunction with some of the other methods listed.

# Considerations

Application whitelisting software should perform several critical functions. First, it should be able to accurately identify and fingerprint applications and services on a given operating platform using methods similar to those just described. Second, it should be capable of allowing administrators to create, configure, and enforce policies based on the software requirements and desired behaviors of systems and applications within the organization.

Below are some key areas to take into consideration when developing whitelisting capablities:

- **Coverage:** Application and platform support are the primary considerations when determining what type of whitelisting system to use. Because most organizations run dozens of application on multiple platforms, the more coverage the better. It's also important to consider types of applications covered. Many whitelisting tools only cover executables (EXEs) and dynamic-link libraries (DLLs), leaving out Java, ActiveX and specialty code such as drivers and kernel components—all of which are actively involved in exploits today.
- **Policy creation:** The ability for IT administrators to create policies, configure host-based agents (and possibly deploy them), and respond to alerts easily is a critical requirement.
- **Centralized management:** Any true enterprise-class solution should facilitate ease of administration in a centralized management console where the authorized administrator can define trusted update sources, including authorized users, application publishers, and file servers, and set up mix-and-match relationships among them. Some specific areas to consider include connection methods, users and groups, and the time of day.
  - **Dynamic whitelist management:** Dynamic management of whitelists is important because it overcomes much of the operational overhead/bottleneck of administrators managing static lists manually.
  - **Connection method:** How and where a user connects to the system may dictate what software can be run safely. For example, the organization may allow Telnet or other unencrypted connections for older systems that can't support SSH or other encryption, but sensitive applications may not be allowed to run when a user attempts to access a Telnet connection.

- **Users and groups:** Applications might need to be installed on a system but run by only certain users and groups. For example, network engineers might need to run Wireshark or other sniffer tools, but no one else should be able to access and run them.
- **Time of day:** In conjunction with other tools like Microsoft's Group Policy, whitelisting tools can also restrict the behavior of software based on time of day. So, for example, if accounting staff is regularly cleared out by 7 p.m. on a weeknight, systems and software should not be active during late night hours. Whitelisting can use this information to enforce access restrictions and detect violation attempts.

• **Maintain gold builds:** Whitelisting can enforce secure builds on end points by enforcing software versions, restricting changes to or downloads of new applications, alerting, and restricting behavior (possibly disabling applications entirely when they are out of date). It can further guard the system and its applications while in use, as well as monitor operations to the memory and block any attempt to write to memory that is not authorized.

• **Integration:** There are also many integration points to consider, particularly when it comes to directories, patch management and provisioning.

- **User directory stores:** Major LDAP and Active Directory directories should be supported.
- **User context:** Whitelisting tools must also be aware of varying degrees of user awareness, in which the particular user context associated with application activity can be applied to the policy functionality.
- **Software deployment and provisioning:** Integration with software deployment tools like Microsoft SMS, IBM Tivoli, and others.
- **Patch management:** The ability to recognize legitimate patching activities generated by tools like Microsoft WSUS, Shavlik, Patchlink, and others is also critical to ensuring that permitted software on the whitelist is kept up to date.
- **Existing infrastructures:** Most enterprises will likely deploy whitelisting technology as an additional measure for defense-in-depth, rather than replacing their host-based IPS/anti-malware software. Integrated solutions allow whitelisting technology to be included and centrally managed with other end point security controls.

With any security tool that blocks application or user activities, there is always a risk of incorrectly identifying a benign program or activity as malicious (false positives), as well as the missing a malicious file or activity (false negative). Security tools should be able to learn and adjust easily. They should also fit into the organizational end point management infrastructure and support multiple use arrangements.

Whitelisting tools should also integrate into existing configuration management practices more easily than other host-based security applications. This is because whitelisting relies on the configuration aspects of the system and applications to identify programs, files, and behaviors that are explicitly allowed or otherwise denied. So, whitelisting can actually be more easily included as a component of standard/gold system builds or images, with clearly defined policies that map directly to various settings of the platform or applications included in the image. Whitelisting tools should even be capable of leveraging a "gold build" or template to define and maintain policies for numerous disparate system types.

In many organizations, there are numerous departments and groups, each with their own specific set of users and privileges, applications, and platform requirements. By creating a number of different images that correspond to these groups' workstations, IT departments can ensure that whitelisting policies stay current with the latest system configurations. By placing these images in a central location accessible by whitelisting software, the central manager will assess the state of the local system and policy as compared to that of the available "gold build." If the "gold build" has been updated, the whitelisting policy can update to accommodate new configuration details, different software versions, patches that have been applied, and so on.

Some whitelisting technologies are starting to employ newer protection capabilities, as well. One of these capabilities is memory monitoring, or behavioral monitoring of system memory to look for signs of buffer overflows and other common memory-based exploits. This capability, in conjunction with application fingerprinting and behavior monitoring, allows whitelisting technology to cover the vast majority of vulnerabilities that could potentially be exploited on a host.

# Whitelisting Use Cases

Although whitelisting technologies can be employed across any number of different platform types, they can be particularly beneficial in environments with very specific technology requirements. Most organizations continue to use traditional host-based security and add whitelisting functionality for another layer of protection. Whitelisting is also suitable for organizations that have legacy systems and applications or single-function devices, such as kiosks, that have been too small to support blacklisting protections.

## Legacy Systems Use Case

For a number of reasons, many organizations are still using older legacy platforms and applications, including Microsoft Windows NT and older varieties of UNIX and Linux, for important business functions. Custom-built and in-house applications that work only on older operating systems are a common cause of this, but simple economics also plays a role in many cases, as businesses look to save money by extending the life of their existing infrastructure as long as possible. Unfortunately, many host-based security tools do not work well on older platforms or might cause conflicts with legacy applications. An even more pressing issue might arise when certain critical patches and updates cannot be applied to legacy systems for fear of breaking them. Sometimes patches simply cannot be applied at all.

Whitelisting technology can be enormously useful in cases where legacy technology is still in use. As long as the operating system is supported, whitelisting agents can fingerprint the allowed applications in use and disallow all other activity, thus circumventing malware from installing through unpatched vulnerabilities. This feature can buy administrators additional time for patch testing and implementation of additional controls if needed.

## Kiosk/Embedded Systems Use Case

One of the most common single-use systems today is the point-of-sale (PoS) terminal. These systems, used for conducting credit card or other payment transactions in retail environments, are becoming more sophisticated and often have special versions of operating systems, such as Windows XP, embedded. Other systems with specialty operating platforms and applications include kiosk applications for banking, conferences, or at major airports for ticketing and flight check-in. In addition, purpose-built systems, such as medical applications and SCADA (Supervisory Control and Data Acquisition) systems, are typically running embedded operating systems that can be vulnerable to common threats affecting Windows and UNIX-based operating systems.

All of these dedicated systems share several unique attributes, the first of which is reduced or nonexistent Internet connectivity, meaning they can't phone home for signature updates very often. Most embedded or kiosk systems have very low bandwidth circuits to the Internet, or no connection at all. Instead, they are more likely to communicate only with a local management or reporting infrastructure or connect to the Internet for short bursts to send small amounts of data. For this reason, they are not good candidates to receive large signature updates from update sites. Because kiosks and embedded systems tend to have fewer changes than day-to-day desktops and servers, maintaining a "Deny All" policy on these systems becomes a simple way to reduce the day-to-day maintenance of security to update cycles these systems can manage.

# Whitelisting and Compliance

For the PCI standard in particular, whitelisting software is gaining momentum as a replacement for traditional anti-virus tools for embedded systems like point-of-sale devices. Although acceptance of whitelisting tools is still currently subject to the individual auditor, more are approving whitelisting as a primary or compensating control in such devices.

"Whitelisting technology is becoming more prevalent in many payment card environments, especially for retail organizations with point-of-sale systems," said Branden Williams, director of Verisign's PCI practice, during a recent interview with SANS about this subject. "Many assessors are now accepting whitelisting as a substitute for anti-malware software to achieve compliance with Requirement five."

The following table is a short list of key areas of major regulations for which whitelisting plays a role. Although brief, it should give organizations a way to look at other regulations with an eye on how whitelisting might fit in.

| Compliance Mandate | Section(s) | How Whitelisting Helps |
|---|---|---|
| Sarbanes-Oxley Act | 404: Management Assessment of Internal Controls | Whitelisting can provide granular control over software running on hosts and what actions can be performed on each host. It should also provide detailed audit trails of actions taken or attempted on hosts and by whom. |
| NERC Critical Infrastructure Protection (CIP) | CIP-003: Change & Configuration Management<br><br>CIP-007: Patching, Malware Prevention, Privilege Control<br><br>CIP-008: Incident Handling and Reporting | Whitelisting manages configuration on a "Deny All" basis: Provides malware prevention, while also controlling actions that can be performed by applications. It can also be a compensating control for patching on legacy systems. It can be used to identify attacks and other incidents in progress, as well as provide audit trail data for investigations. |
| Payment Card Industry Data Security Standard (PCI-DSS) | 2.2 Configuration Standards<br><br>5.1–5.2 Install and Maintain Anti-Virus Programs<br><br>6.1 Patch Management<br><br>7.1–7.2 Restrict User Access and Privileges | Whitelisting can help satisfy a number of the PCI DSS requirements related to configuration management, maintaining anti-virus software, patch management, user behavior controls, and others.<br><br>It is particularly useful for enforcing secure builds on PoS systems in retail environments, where financial data and currency are handled. |

# Looking Forward

Many enterprises have significant investments in host-based anti-malware products that they have fine-tuned over time to meet policy needs in their myriad environments. The ability to control application behavior and who can run those applications gives organizations a powerful new tool to achieve a more effective defense-in-depth posture at the host level.

For those dedicated and legacy systems that cannot run more traditional host-based security tools (such as kiosks, medical devices, and PoS terminals), whitelisting provides protection from malware infections and other compromise even if systems can't be patched and updated regularly. By creating a "Default Deny" security posture, whitelisting works by maintaining a persistent state of approved applications on the end point until the system can be updated.

For kiosks, legacy systems, and remote end points, whitelisting provides security options that were previously unavailable to them. For other end points across the enterprise, whitelisting provides another layer of the defense-in-depth strategy that is critical in today's complex IT environments.

# About the Author

**Dave Shackleford** is the owner of Blue Heron Security, an information security consulting firm, and instructor and course author for the SANS Institute, where he serves as a GIAC technical director. Previously, Dave worked as the chief security officer of Configuresoft and the chief technology officer for both the Center for Internet Security and a security consulting firm in Atlanta. He has managed information security for a major airline and has also worked as a security architect, analyst, and manager for several Fortune 500 companies. In addition, he has consulted with hundreds of organizations in the areas of regulatory compliance, security and network architecture, and engineering. 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. Dave 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 this paper's sponsor*