

Increase Security Posture With Application Whitelisting

Dwight Anderson
Schweitzer Engineering Laboratories, Inc.

Presented at the
13th Annual Western Power Delivery Automation Conference
Spokane, Washington
March 29–31, 2011

Increase Security Posture With Application Whitelisting

Dwight Anderson, *Schweitzer Engineering Laboratories, Inc.*

Abstract—Application whitelisting is a cybersecurity design that increases the security posture for the substation environment, while reducing computational overhead and congestion. This paper discusses application whitelisting as a tool to secure computers in a substation environment, providing a new countermeasure to deal with malware. Application whitelisting only allows applications on the whitelist to run. This security approach eliminates the need for antivirus definition updates.

This paper describes what the benefits of whitelisting are and how to use application whitelisting. Traditional antivirus software works against threat tactics by blocking malware based on a set of definitions. As most users have experienced, this technique tends to become congested and often slows down system operations with the ever-increasing size of virus definitions. Application whitelisting is a technique where the operating system allows a limited number of programs to run, while blocking all other programs from running by default. The end result is better cybersecurity with less overhead obfuscating operations.

I. CONTEXT REGARDING APPLICATION WHITELISTING

This paper focuses on application whitelisting software restriction policies to secure computers that help automate systems found in a power utility environment and other similar mission-critical process control systems. The benefits that computers and networking provide include their ability to centrally locate technical resources and save time, resolving or preventing issues that might negatively impact safe and reliable power system operation. The security of these systems is critical for continued safe and reliable power infrastructure operation.

In the past, resolving a power system problem and restoring power often took hours. Today, restoring power often occurs within minutes, seconds, or even milliseconds, thanks to automation and communication. In the not too distant past, a person with no computer had to drive to the area of concern and attempt to determine the root cause of the failure to resolve it. Identifying, locating, and remediating the failure often took time and depleted valuable resources just traveling to and from the failure. Today, microprocessor-based relays, computers, and communications channels work together to automatically take action and send technical information to the subject matter experts. Centrally locating experts in order to identify, locate, and, in some instances, mediate a problem (without the need to travel) saves a great deal of time and resources. A central location provides access to a larger collection of resources, such as other accessible subject matter experts, to offer advice to resolve issues. Remote access to systems provides access to historical data

that may help resolve problems as well. Unfortunately, these tools, such as computers and networks, operate on a premise of trust. This leads to the need to make these tools more secure.

One area of concern with computer security is the need to use antivirus software. This leads to the problem that antivirus software must update its definitions on a regular basis to be effective. As new viruses arise, new antiviruses must reach the computer to maintain security. This usually requires a connection to the public Internet to update antivirus definitions. If a connection to the antivirus vendor is not available, as is often the case in a substation environment, the administrator must find other ways to distribute updates. Updates are important even on computers not connected to a public network because viruses and malware are often introduced by removable storage. Application whitelisting software restriction policies eliminate the need for a public Internet connection.

Application whitelisting software restriction policies, like all security tools, must be used in context. Just focusing on a single security technology to address a specific security issue may provide a false sense of security and lead to larger security failures in overall system operation. Good security must fall in line with other priorities, such as safety and reliability. Also, good security must follow time-tested security frameworks, such as those found in storage area networks (SANs) and the International Information Systems Security Certification Consortium, Inc., that recommend creating well-formed security policies and procedures. These procedures must align with the way a company does business, as well as support regulations, such as those found in the North American Electric Reliability Corporation (NERC) Critical Infrastructure Protection (CIP) requirements. Once good security policies and procedures are in place, they create a foundation for a tool like application whitelisting.

Computer security is an important part of an overall security program. It is important to approach security by applying multiple layers of security measures, creating what is known as a defense-in-depth security posture. A good defense-in-depth posture is like an onion that has multiple layers. An attacker may compromise one layer, but if the attacker finds new and different layers of security, it becomes too troublesome to continue. Increasing the number of security layers on a computer platform deters an attacker, causing the attacker to retreat altogether.

This paper encourages the use of multiple layers, and the reader is reminded to not focus on one layer. Application whitelisting software restriction policies provide a new tool

that administrators have at their disposal. If the administrator includes strong passwords and firewalls and sets up user accounts with privileges that align with the user's need to know, this tool creates a formidable part of an overall security program.

II. WHAT ARE APPLICATION WHITELISTING SOFTWARE RESTRICTION POLICIES?

Traditional antivirus software uses samples of malware code and compares these samples to file contents. The antivirus software uses a list of known bad signatures to quarantine or remove the offending malware (i.e., blacklisting). In some cases, the virus mutates, and the database of signatures or samples of viral code does not work, or the delay between the virus and its antivirus signature causes what is known as a zero-day vulnerability.

To address zero-day vulnerabilities, antivirus software may use heuristics—a fancy name for the way antivirus software seeks and prevents bad behavior. For example, a virus may behave badly by changing the content of an important program file. The virus might change the computer host file, for example. The computer host file maps Internet Protocol (IP) numbers to fully qualified domain names. If a computer used for supervisory control and data acquisition (SCADA) systems contained malware, a user looking at a human-machine interface (HMI) may unknowingly be looking at a redirected or forged SCADA HMI interface. In this scenario, the heuristic or bad behavior that the antivirus software examines is to watch if a program attempts to change the host file on a computer. Typically, the antivirus software creates a quarantine area that the program starts in, and if it does not exhibit the bad behavior of a virus (such as changing an IP host file), then it is allowed to start and run outside of the quarantine. These actions all take time and require regular updates to maintain good security.

The way antivirus protects a computer is known as blacklisting, a well-known approach to computer security. An alternative to blacklisting is whitelisting. In the case of whitelisting, the security process defines a set of rules that allow or permit files to operate or execute in a computer. In application whitelisting, a system administrator provides a permitted list of software programs and files that, if they meet the appropriate rules, can run on the computer. If the software or file is not on the permitted list and there are no rules to allow it to operate, the software is restricted or not allowed to operate. There are provisions for such a policy and rules in the Microsoft® Windows® XP and Linux® operating systems. There are also third-party software packages that provide this type of approach to dealing with malware and a valuable alternative to malware protection with no need for updates or a connection to the Internet.

The focus of this paper is based on the Microsoft approach to application whitelisting. The Microsoft term for this solution is software restriction policy (SRP). The same principles also apply to Linux, as well as to third-party software approaches, such as McAfee® Embedded Security™.

The following is an overview for application whitelisting software restriction policies. Suppose a user tries to run a new substation training video on a computer. Also, assume that someone modified the video to contain and execute a hidden malicious program. If the user inserts or downloads the video onto the computer, such as from a Universal Serial Bus (USB) thumb drive, and opens the web browser to that file or runs the video, the malicious program loads and becomes operational.

In this example, if application whitelisting software restriction policies are operating on the computer, the malware would not have the correct rules or permissions to run and therefore would not execute. This paper demonstrates similar actions and explains how to generate a warning banner and log event.

There are some assumptions and issues regarding application whitelisting software restriction policies. Namely, when a computer is built, there is an assumption that no malware is installed onto the system. There are ways to address this issue and get the computer to a known good state.

The other issue in using this form of application whitelisting is that the computer must have an administrator who manages software installations and authorizes the application whitelisting software restriction policies. The administrator and application whitelisting enforce and log the software approved to operate. The administrator should periodically review the event log files for optimum security results.

A computer in a power utility control system infrastructure lends itself well to the use of application whitelisting because it operates as a closed system. There is no need to install new software onto the control system computer. Even so, there are ways to permit installation of new application software if it is needed.

III. BASIC STEPS FOR APPLICATION WHITELISTING SOFTWARE RESTRICTION POLICIES

The following is the basic outline of actions that create application whitelisting software restriction policies in a Windows operating system environment [1]:

- Create and enable local software restriction policies.
- Set up and apply enforcement of the policies.
- Designate the file types considered as executable.
- Generate the rules relating to hashes, certificates, paths, and Internet zones.
- Assign the software restriction policies, designated file types, and rules to users or groups.

The following example describes these steps. For a Windows software restriction policy, there are two security levels: disallowed and unrestricted. Disallowed means a program is not permitted to run unless additional rules enable it to run. Unrestricted means the software access rights are determined by the access rights of the user.

In this example case for the substation HMI computer, select disallowed because the system is a closed system and the list of programs needed for operating in the substation environment is a known set.

In Windows, the default software restriction policy is assigned to unrestricted: in other words, it is wide open. When setting up the software restriction policies, this is changed to disallowed. That is, regardless of the user access rights, the operating system will not run the software. In order to allow updates and software installations, the default enforcement setting changes to **All users except local administrators**. In this case, local administrators are able to run all programs. The administrator assigns the policy to particular **Designated File Types Properties**. This configures the application whitelisting software restriction policies to act on particular file types.

IV. APPLICATION WHITELISTING SOFTWARE RESTRICTION POLICY RULES

This section lists the application whitelisting rules that quantify the permissions for the software applications to operate on the computer. Windows supports the following four ways to identify and secure software applications:

- Hash rules provide cryptographic calculations, or hashes, to program files to enable them to operate. The hash on the file provides a mathematical digest or digital fingerprint. The fingerprint can either allow or prevent a program from running. Permission or denial of program operation occurs regardless of its location or name.
- Certificate rules use a signed software publisher certificate for permission to run. Like the hash rule, this rule applies regardless of the program location or name.
- Path rules apply to programs that run from a specific (local or network) path or from subfolders located in the path.
- Internet zone rules apply policy rules based on the Windows Internet Explorer® security zones, but Internet connections are not recommended for use in the substation environment.

The above rules provide the means to restrict or permit programs from running, even if they are modified or changed. The software restriction policy can also apply to file types.

V. RELATING WHITELISTING SOFTWARE RESTRICTION POLICY TASKS TO RULES

The Microsoft TechNet website has a useful table to identify when and what rules to use for various protection scenarios [2]. This table identifies appropriate rules for software restriction policy scenarios.

It is very important to understand that there is precedence in the software restriction policy rules. The rules follow this order (from highest to lowest): hash rule, certificate rule, path rule, and Internet zone rule.

The rules also follow precedence, with the highest assigned to rules that are more specific. If a path rule is defined for **C:\Example** with a disallowed security level and another path rule is defined for **C:\Example\Subfolder** with an unrestricted security level, the rule with **C:\Example\Subfolder** is more specific. Therefore, the

unrestricted path rule takes precedence. The more conservative rule takes precedence if there are two identical rules with differing security levels.

In certain unique situations, the hash rule can even help prevent running code caused by a virus or a Trojan horse. An administrator could create a hash rule by calculating the hash value of the virus program and then restricting the hash value from running. This rule operates independently of the name or location of the malware. Also, if the malware takes a particular known path, a path rule could be set up to prevent execution of the malware.

Performing a successful attack against a system using application whitelisting software restriction policies requires that an attacker exploit the permitted programs list. However, enabling the policy means preventing this action because the policy rules watch for changes in a program by means of hash values.

Another positive aspect of application whitelisting is its ability to generate event logs that alarm and notify of attempts to bypass the security restriction policy. For example, when the application whitelisting software restriction policies prevent a program from running, a log event is generated in the Microsoft Windows event log file, as shown in Fig. 1.

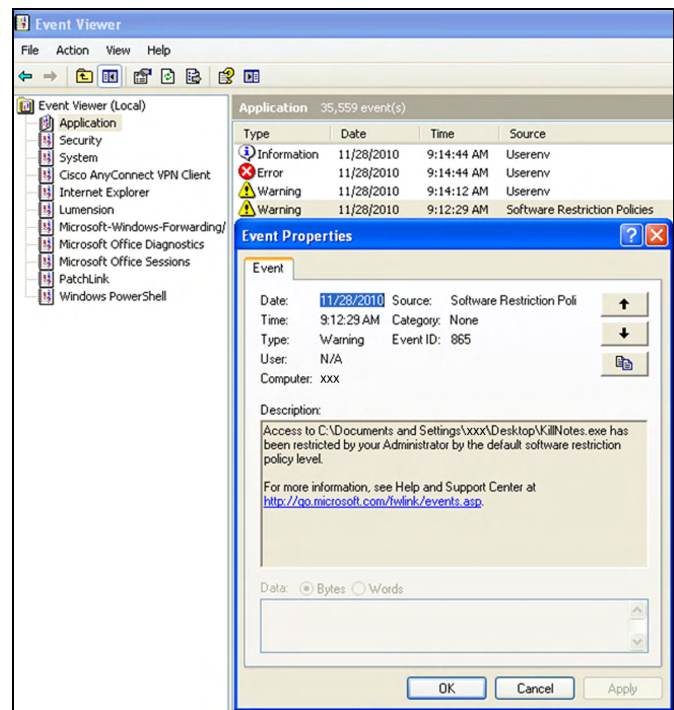


Fig. 1. Example event log

VI. HOW TO SET UP APPLICATION WHITELISTING SOFTWARE RESTRICTION POLICIES ON A WINDOWS COMPUTER

This section identifies the specific steps to take to implement the application whitelisting software restriction policies available for Windows XP Professional [3]. This example assumes a new computer and software image as delivered from the manufacturer. All necessary third-party application software modules were installed with appropriate licenses. Also, in this example, unnecessary software was

removed from the computer. An antivirus and vulnerability scan were conducted prior to setting up the application whitelisting software restriction policies. This process ensures that there are no incipient malware and/or vulnerabilities installed on the computer as received from the manufacturer.

In this example, application whitelisting only protects those accounts logged on as users. If users are permitted administrative-level privileges, they do not receive protection from this policy. When evaluated for application whitelisting, a third-party software program was able to provide protection for both administrative and user access privileges.

The following steps assume the person configuring the computer is logged on with administrative-level privileges and is configuring these policies for user-level accounts. The example demonstrates a basic application of a whitelisting policy using Windows software restriction policies. It is intended to get readers to a starting point and help them become familiar with application whitelisting. The author encourages readers to experiment with additional rules and access rights.

- Step 1. Click **Start > Run**, and enter **gpedit.msc**, as shown in Fig. 2. Alternatively, click **Start > Control Panel > Administrative Tools > Local Security Policy**. Look for the **Software Restriction Policies**.

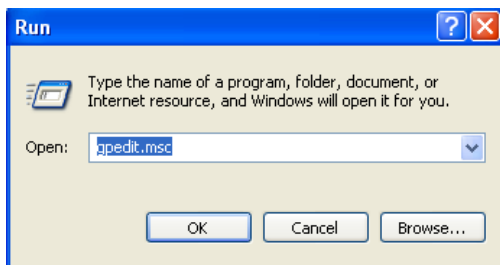


Fig. 2. Quick way to open the policy menu

- Step 2. When running **gpedit.msc**, select **Computer Configuration > Windows Settings > Security Settings > Software Restriction Policies**, as shown in Fig. 3.

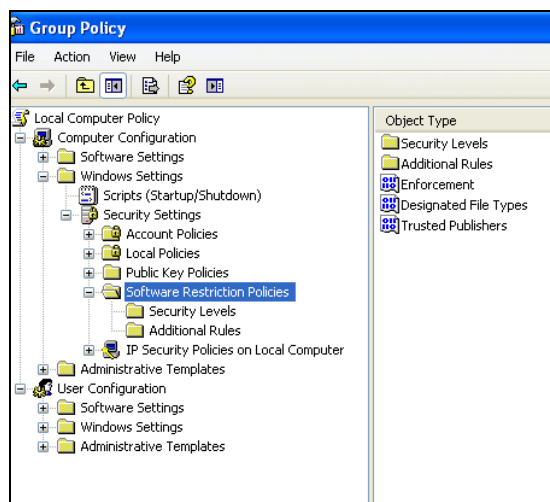


Fig. 3. Start new group policy with **gpedit.msc**

- Step 3. If **Software Restriction Policies** is not visible, go to **Action > Create New Policies** to enable this function. It populates the drop-down list, as shown in Fig. 4.

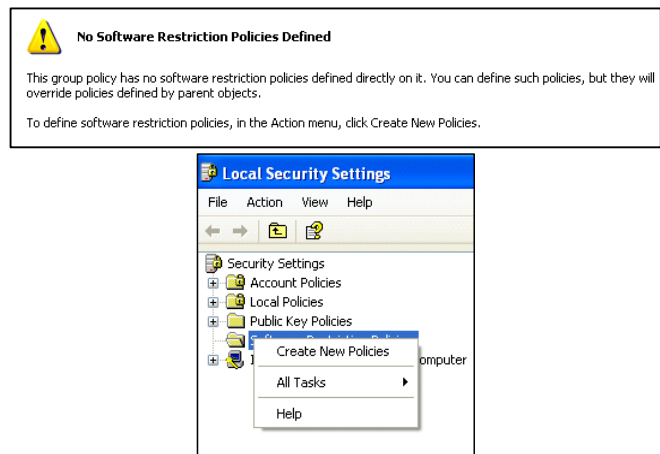


Fig. 4. Create new policy if none is shown

- Step 4. Select **Software Restriction Policies**, and then double-click **Enforcement** in the right pane, as shown in Fig. 5.

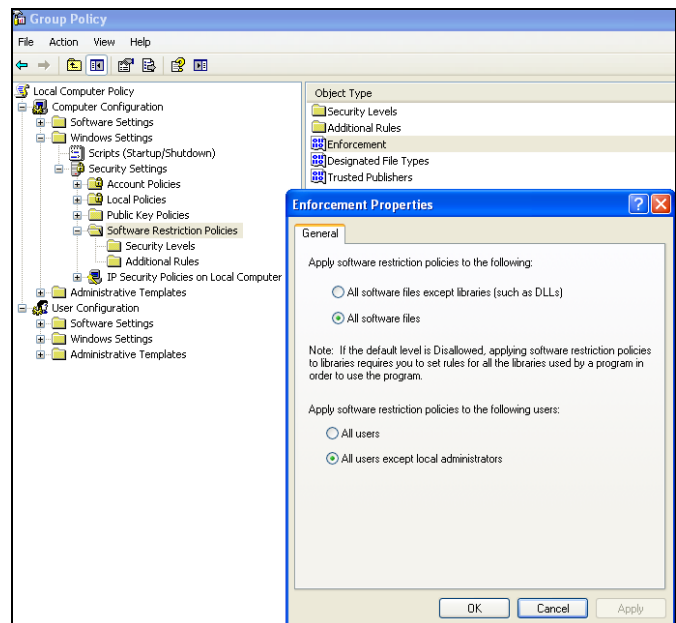


Fig. 5. Enforcement policy settings

- Step 5. It is very important to follow Step 5 to enable the policy to act on all files. Choose **All software files**, select **All users except local administrators**, and click **OK**. As stated previously, the assumption is that the person administering these steps has administrative privileges on the computer and has the authorization to make these changes. The reason for this exception is to allow the administrator the right to make changes to the policy and install and/or update software.

- Step 6. Select **Designated File Types Properties**. The list shows the files that are restricted from running. For example purposes, select the **LNK** file type, and delete this extension from the list. This action allows the Windows shortcuts to work normally. Later, by reinserting this option into the list, the administrator can test user-level access and learn if the software restriction policy is working. Select **OK**, and close the **Designated File Types Properties** with the **LNK** file type not on the list, as shown in Fig. 6.
- Step 7. In the left-hand pane, select **Additional Rules**. Notice the default rules for hashes, certificates, paths, and Internet zones. At this point, do not add any additional rules. For demonstration purposes, a path rule could be added that prevents access to a particular folder location.
- Step 8. The last step is to select **Security Levels**, and click on **Disallowed**. Set this as the default. Select **Yes** to confirm the changes, as shown in Fig. 7.

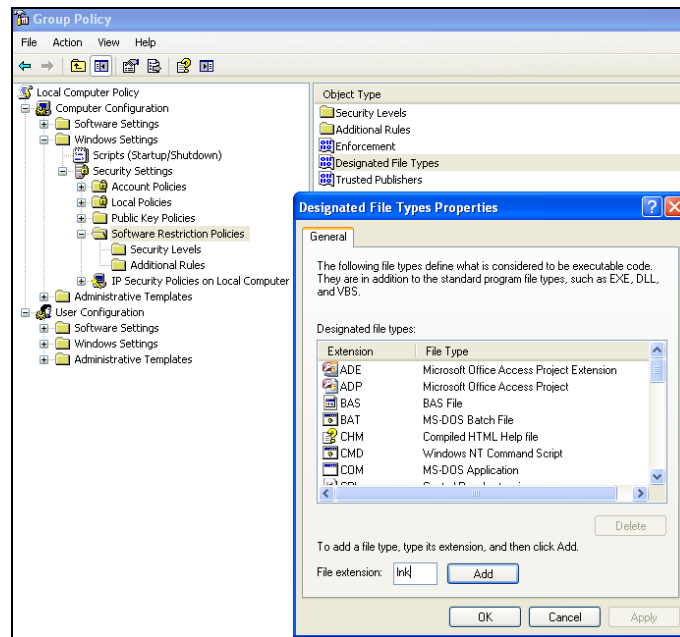


Fig. 6. File type associations

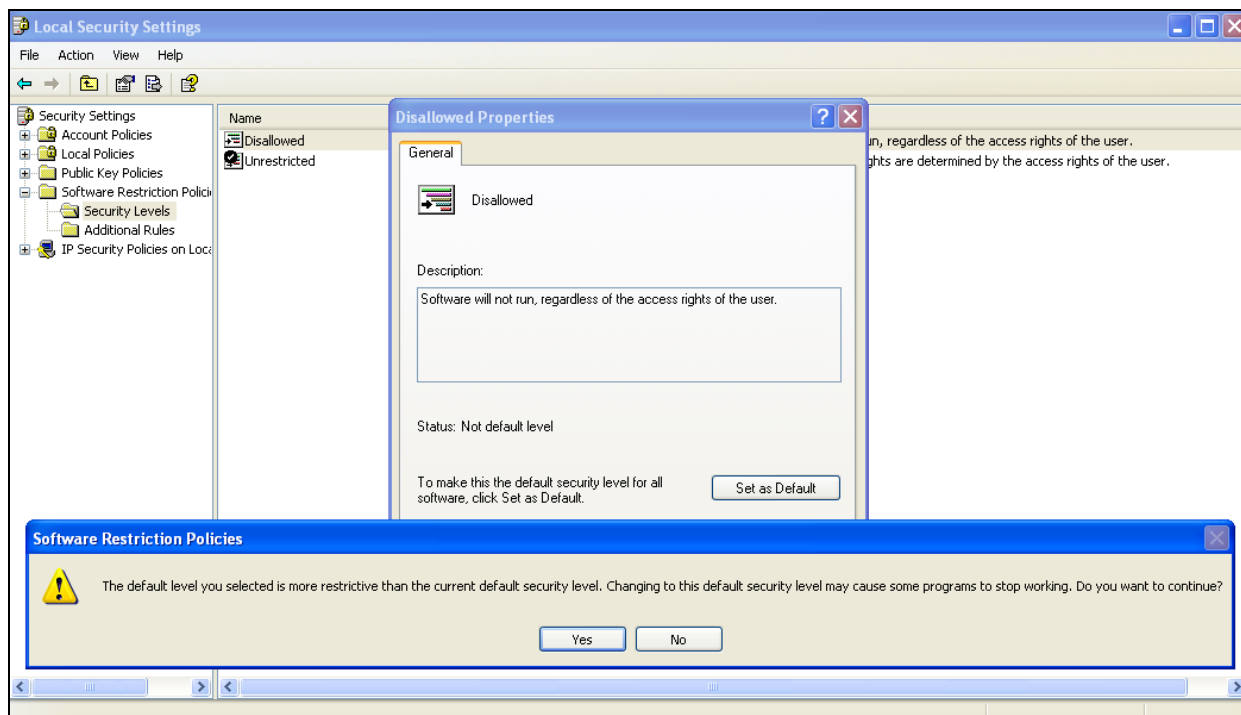


Fig. 7. Changing the default settings

At this point, for the system administrator, there is no change in the computer operation. However, if the system administrator logs off and logs back in as a user, only those files permitted by the software restriction policy can run. Attempting to run or install a new program generates a warning and log event.

VII. TESTING APPLICATION WHITELISTING SOFTWARE RESTRICTION POLICIES

It is important to test the application whitelisting software restriction policies on a computer in a test environment prior to implementing them in an operational environment.

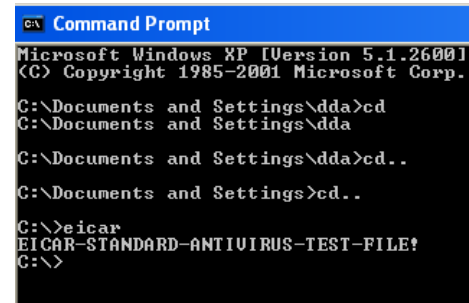
Tests were conducted with both Windows XP Professional application whitelisting software restriction policies and a third-party software package (McAfee Embedded Security). The test computer ran as a real-time distribution automation control system. The computer was set up and interoperating with a real-time power system simulator. The system operated normally with the application whitelisting software restriction policies and third-party software package. During operation, tests were conducted to see how well the application whitelisting protected the substation distribution automation system.

One part of a test strategy is to find a means to simulate malware and attempt to install and run the file. This checks if the administrator deployed the application whitelisting software restriction policies correctly. It also checks to see if the application whitelisting prevents operation of the simulated malware.

For the purpose of this paper, it is preferable not to deliberately generate or play with a real virus in order to test. Because it is an unacceptable risk and dangerous to test with real viruses, the European Institute for Computer Antivirus Research (EICAR) provides a file that can be safely passed around [4]. It is nonviral but tests antivirus measures as if it were a real virus. This simulated test file is the **EICAR Standard Anti-Virus Test File** [4]. It is safe to use because it is not a virus and does not include any fragments of viral code. As part of the testing, consider including the Information Systems (IS) security department to help administer and oversee this test. Because anti-malware products react to EICAR as if it were a virus, the IS department may receive notices of viruses coming from the test computer. Typical anti-malware responses log and report the file as **EICAR-AV-Test**.

Informing the IS department of issues prevents invoking unnecessary alarms and responses to a legitimate testing. The EICAR file is a DOS program and produces the display

message **EICAR-STANDARD-ANTIVIRUS-TEST-FILE!** (see Fig. 8) if the anti-malware does not prevent it from running.



```

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\dda>cd
C:\Documents and Settings\dda
C:\Documents and Settings\dda>cd..
C:\Documents and Settings>cd..
C:\>eicar
EICAR-STANDARD-ANTIVIRUS-TEST-FILE!
C:\>

```

Fig. 8. EICAR test from the DOS command prompt

Another part of the test process was an attempt to run and move the file into different locations. The test file did generate appropriate log entries in the Windows log files. Each attempt to run the malware produced a log entry in the Windows log file, indicating that the application whitelisting software restriction policies prevented the malware from taking action. In some cases, a warning banner directed the user to contact the system administrator for further information (see Fig. 9).

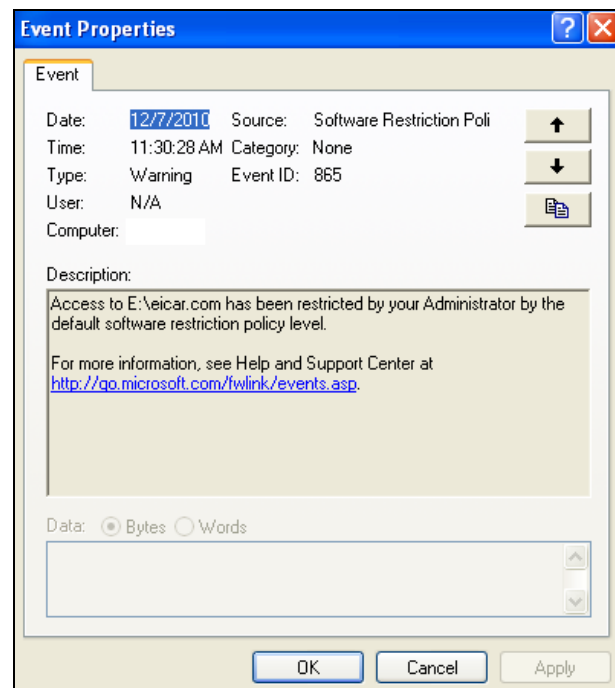


Fig. 9. Event log file warning

The tests showed how important it is that the system administrator create strong access controls to operate in conjunction with the application whitelisting software restriction policies. Adding access controls along with application whitelisting policies prevented attempts to copy the simulated virus from a USB memory stick to the computer system. Access controls, such as file permissions, can disable the write permission for a user, as shown in Fig. 10.

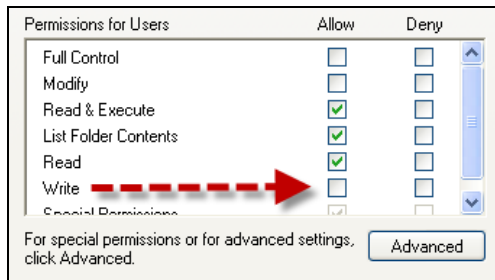


Fig. 10. Setting access control permissions

The testing showed no negative impact in the performance of the distribution automation system, yet it provided anti-malware protection.

VIII. FURTHER STUDIES ON UPDATING WHITELISTING SOFTWARE AND RESTRICTION POLICIES

Application whitelisting is a relatively new tool; however, it needs more testing. How would a person update the application whitelisting programming infrastructure? This is an especially difficult problem in a control system environment. It might be beneficial for application whitelisting to allow creation of an auditable or baseline document of the system configuration. The document would allow a user, at a later point in time, to assess if the application whitelisting system configuration underwent changes.

Documentation regarding who, what, and when the application whitelisting was modified is an important tool for an audit document. Related to this topic is the process to update the infrastructure for application whitelisting. Microsoft tends towards an automated solution that is often bundled into its operating system service patches. A power utility must test that the fixes do not negatively impact safe and reliable operations. This concept aligns well with a signature phrase adopted by former President Ronald Reagan of a Russian proverb, “doveryai, no proveryai” (Russian: Доверяй, но проверяй). “Trust, but verify.” Trusting and verifying the update or patch before installing it into a substation is an important concept for critical infrastructure system administrators to follow.

IX. CONCLUSION

Application whitelisting software restriction policies are a new tool that holds promise to secure control system computers that are stable or fixed in their configurations. A third-party software package provides easier configuration and greater coverage by protecting the administrator.

Administrators want confidence that their systems operate with known and verified software. Application whitelisting provides a promising technology to guarantee that only authorized code can operate.

Application whitelisting is a cybersecurity tool that has the potential to increase the security posture for the substation environment, while reducing computational overhead and congestion, providing a new countermeasure to deal with malware.

X. ACKNOWLEDGMENT

The author gratefully acknowledges Eric Cosman of Dow Chemical for his help and guidance on application whitelisting.

XI. REFERENCES

- [1] P. W. Barnes, “Defending Windows With Application Whitelisting,” September 2009. Available: <https://patrickwbarnes.com/blog/2009/09/defending-windows-with-application-whitelisting/>.
- [2] Microsoft TechNet, “Using Software Restriction Policies to Protect Against Unauthorized Software,” May 2004. Available: <http://technet.microsoft.com/en-us/library/bb457006.aspx>.
- [3] Microsoft TechNet, “What’s New in Security for Windows XP Professional and Windows XP Home Edition,” February 2003. Available: <http://technet.microsoft.com/en-us/library/bb457059.aspx>.
- [4] EICAR: European Expert Group for IT-Security, “The Anti-Virus or Anti-Malware Test File.” Available: <http://www.eicar.org/>.

XII. FURTHER READING

National Security Agency: Information Assurance Directorate: Vulnerability Analysis and Operations: Systems and Network Analysis Center, “Application Whitelisting Using Software Restriction Policies,” June 2010. Available: http://www.nsa.gov/ia/_files/os/win2k/Application_Whitelisting_Using_SRP.pdf.

XIII. BIOGRAPHY

Dwight Anderson received his BS in electrical engineering from Steven’s Institute of Technology. He is now a project engineer for Schweitzer Engineering Laboratories, Inc. (SEL) in Pullman, Washington. Prior to joining SEL in 2005, Dwight worked 20 years for Hewlett-Packard as an aerospace and defense business development manager and systems engineer, working on projects ranging from electronic warfare countermeasures to SCADA system programming. He holds the Global Security Essentials Certification (GSEC) from Global Information Assurance Certification (GIAC) and is a Certified Information Systems Security Professional (CISSP).