Global Information Assurance Certification Paper

PROTECT CRITICAL INFRASTRUCTURE COMPUTER SYSTEMS WITH WHITELISTING

GIAC (GSEC) Gold Certification

Author: Dwight Anderson, dwight_anderson@selinc.com
Advisor: Hamed Khiabani, Ph.D.

Accepted:

Abstract

Protection of critical infrastructure computer systems from malware and zero-day exploits is imperative because critical systems provide for the well-being of the public by supplying valuable public resources, such as water, gas, and electricity. This paper explains whitelisting and its use to protect computers that provide a human machine interface into systems that control critical infrastructure operations, such as those found in utilities. Whitelisting is discussed in detail as well as how it differs from traditional antivirus software, or blacklisting, and why whitelisting lends itself for use on critical infrastructure computers more favorably than traditional antivirus software. This paper also discusses misunderstandings in the use of whitelisting while supporting and promoting the use of both whitelisting and blacklisting as a means to increase the security posture of computer systems and even protecting them from zero-day and other malware exploits.

1. Introduction—Critical Infrastructure Systems Operation and the Differing Need for Security

Today there tends to be a misunderstanding regarding the operational aspect of critical infrastructure systems. For example, systems that provide us with on-demand or instant access to electricity, gas, and water operate with a much different set of operational conditions than systems that provide us Facebook or email. For example, informational systems that provide us with email provide the information with data confidentiality as a higher priority than data integrity; lower on the list is availability. These form the well-known security triangle or acronym of CIA (Confidentiality, Integrity, and Availability). For informational systems, this tends to match expectations; namely, systems that serve email assure that privacy (confidentiality) is first, integrity is second, and last the email is available, or has been received within a reasonable amount of time, possibly minutes. However, critical infrastructure systems that support the safe and reliable delivery (availability) of electricity, water, or gas flip the CIA around 180 degrees, and delivery occurs in near real time or milliseconds. The traditional CIA flips around to AIC because "availability" of electricity, water, or gas via a critical infrastructure is the highest priority. That is, we expect to have electricity, water, or gas when we either flip a switch or turn a valve. Then after delivery, these must not undergo modification of their content; specifically, the critical infrastructure must assure integrity of delivery. The general public should not know how much any individual consumes of these things. The expectation is that these items be delivered and available. For example, first we expect our water to be available, and then we expect it to be clean and pure. But there is less expectation over the amount of consumption of water.

Over the past few decades, critical systems that deliver things like electricity, gas, and water increased automation as a means to increase reliability and dependability. For example, they must be robust. (Langer, 2012) The improvement in safety and reliability from automation did not occur as a result of government regulations but because the general public grew to expect it. Imagine if the power or water were off for a period of

time and yet neighbors down the street had theirs available. This would be fairly intolerable and most likely would result in complaints. Advancements in interconnectivity and automation of these systems helped reduce downtime and serviceability from hours to minutes. (Guzman, 2002) More recently, over the past few years, the systems in many areas of the country increased the serviceability for critical consumables to seconds. If and when there are interruptions to the delivery of power, gas, and water, they are often associated with unexpected traumatic forces from catastrophic natural events, such as hurricanes, tornadoes, earthquakes, or fires. It might surprise some readers that in the United States, it was only three decades ago that a simple lightning strike could easily cause the power to be off for an hour or longer while work crews located and restored power by replacing line fuses. Today, automation provides intelligence so that the infrastructure is able to help self-identify, isolate, and even remediate without the need to dispatch work crews.

The technical advancements that deliver intelligent or smart-grid systems which increase the reliability and dependability of electricity are the result of a number of technological advancements. The technical advances in increased communications speed, increased communications interconnectivity as well as increasing computational power provided significant advances in systems distributing intelligence and in decision making. Not only did the above trends allow subsecond response times to provide self-diagnosis and repair, they also took less space and often cost less money. For example, the equivalent intelligence in the microprocessor chips found in the early 1980 Apple computers or IBM PCs are now in home refrigerators even dishwashers. The benefits are when power outages do occur, the electricity may turn off or blink on and off, but then afterwards, system automation allows the electric power to heal itself. During this "blink" in time, the fault is isolated to a very narrow number of customers and electricity is rerouted and restored. (Allen, 2008)

Unfortunately, the same advancements that provide more robust delivery of critical infrastructures also increase the risk to these systems of a cyber attack. Because there is more reliance on computers and interconnectivity of these systems, they, therefore, become easier targets for persons who would like to wreak havoc. This is the backdrop for the focus of this paper. As security experts, we cannot simply add security

measures without understanding the net effect these measures may have on the overall safety, reliability, and operation of critical infrastructure systems. It does not help if when adding in security measures, that the net effect is that electricity, water, and power become less available, less reliable, and less dependable but are more secure. It seems today many researchers tend to think that solving security issues is to simply drop in security measures—namely, just applying those security measures that have long been available in information technology. But it is imperative to understand that the flipping of CIA to AIC and for communications to operate in real time as well as be deterministic requires that any security measure installed to protect basic infrastructure systems must be of good design and undergo rigorous testing to ensure there is no negative impact to the real-time and deterministic operation of the critical infrastructure.

Before getting too deep into application whitelisting and the security for computers deployed on critical control systems, this paper requires a brief discussion and background on supervisory control and data acquisition (SCADA) and the use of computers that provide a human machine interface (HMI) or graphical depiction into the SCADA process.

Basically SCADA systems provide a means to automate the control of processes by sequentially polling for measurement data (data acquisition) and then comparing the polled data against a set of pre-programmed values (supervisory control), and if needed, the SCADA system changes the output based on the difference. For example if a process requires heating water in a boiler to a specific temperature, the SCADA system might measure the output of a temperature sensor, and then if the temperature is too low, the SCADA system will close a relay to turn on the heating element in the boiler until the temperature reaches the desired set-point or temperature. In this simple example, there may be other sensors, such as a level sensor or even a pressure sensor that allows the SCADA system to know if there is too much or too little water and/or how much pressure there is in the boiler. The SCADA system is able to periodically poll all the various sensors in order to supervise and control the process via periodic acquisition of various data points. In general, SCADA systems will monitor tens to thousands of boilers or processes. Often the SCADA system will display a graphical output of the entire suite of

processes, typically on multiple displays. Also, the SCADA system will typically store the history of the measurements and changes in a file server known as the data historian.

The benefits of SCADA are that it allows for easier and greater monitoring and supervision of an entire industrial process from a central location. In many industrial processes, there are hundreds to thousands of data points that a SCADA system might continuously monitor and control. Prior to SCADA, many processes utilized a manual process or semi-automatic process where people or strip chart recorders operated across the industrial control complex. By automating the process with SCADA, it relieved the burden of placing persons in unfavorable working conditions and provided safer and better quality working conditions. SCADA streamlined and accelerated process decisions, allowing for cost and time reductions even conserving the size of the staff required. By lowering costs, SCADA provides greater affordability and a higher availability of resources to larger groups of people. As an example, consider the processes required to clean and transport drinkable water. If SCADA were not available, then many people would not have immediate access to this resource. It is an unfortunate fallacy that hackers think by attacking these systems they are in some way helping to increase security, when in fact they are increasing the costs and therefore reducing availability of these resources to the very people they think they are protecting.

Of the many automation systems in use on critical infrastructure control systems, none play as important role as do computers operating as human-machine interfaces (HMIs) often deployed as a means to graphically depict SCADA operations. HMI computer systems are often located in key locations and provide the means for an operator, engineer, or technician to view of status, trends, and even make changes to a control system. HMI computers provide an easy means to visualize supervisory control and data acquisition (SCADA) systems that control the transportation of gas, electricity, or water. Computers that act as HMIs do not undergo a great deal of change. Namely, the operating system and application software residing on an HMI computer do not undergo or require changes in applications nor do they require new installation of software.

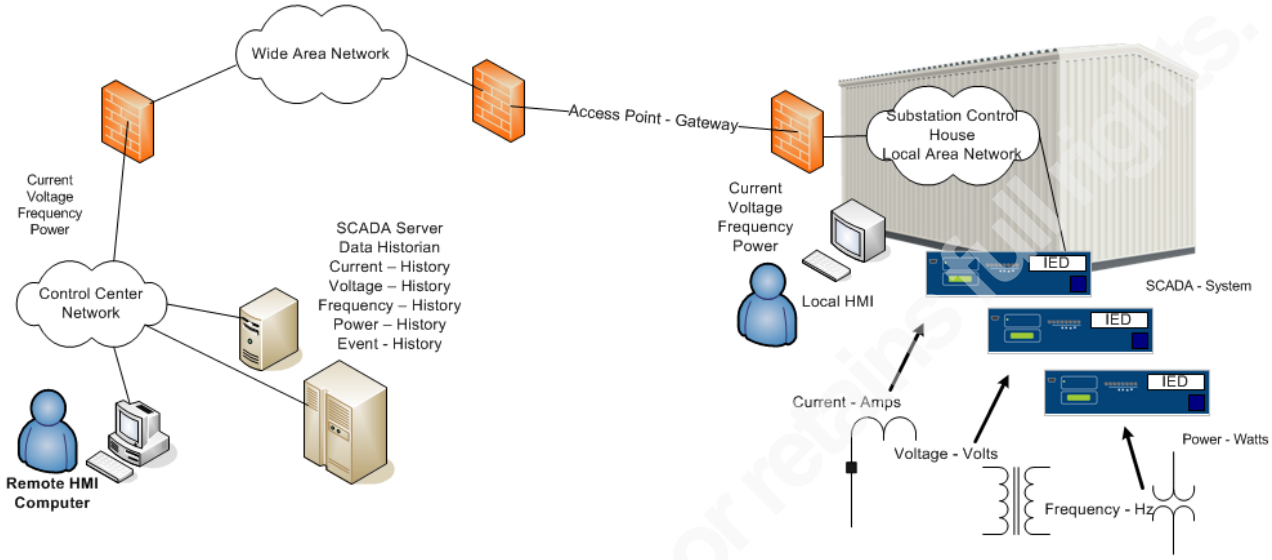

Figure 1 Figure 1 HMI Computers Interfacing Into Electric Substation SCADA System

Shown above in Figure 1 is an example depicting two HMI computers that provide a graphical interface into a power substation SCADA system. In the example shown the intelligent electronic devices (IED's) poll for voltage, current, frequency, and power of transformers stepping up and down electricity from high to lower voltages. The function of SCADA in this example helps provide basic measurements for balancing of load and demand for the larger electrical power system. The figure depicts a substation control house with a local HMI computer that provides a local engineer with the ability to view the status of the SCADA system. Likewise a remote HMI computer connects into the network and receives the same data. This provides a second engineer the means to remotely view SCADA operations as well. Also noted on the diagram is a SCADA server and data historian. All these computers needs protection from malware as the interface with the SCADA system.

2. Enter Application Whitelisting

Application whitelisting is a method that examines the contents of a computer and creates a list, "whitelist," of all the programs that are permitted to run. The application

whitelist also oversees the operation of any program requesting permission to run, and if the program is not on the list, it does not run. The application whitelist assumes that the state of the computer and application programs that are installed and operational will not undergo significant change, specifically, that the computer will not have a new version of a photo editor installed and later in one month, a new version of a music player. Application whitelisting expects the programs on the computer to be relatively static.

One way application whitelisting generates a known good list is by creating a hash digest of all software applications. It uses this hash list to determine if the application has permission to run. If the hash matches what is contained in the list, it will run. If the software is not on the "list" or its hash value is not correct, the program has no permission to run and will generate a security event.

The way that application whitelist programs generate their permitted list is similar to the way software hash verification works. For example, software manufacturer's webpages often provide either an MD5 or SHA1 hash value. The hash value is a cryptographic or complex mathematical result that provides a method for a user to compare the value they calculate on a program to the value that the manufacturer provides as a result. Both values should be the same, and by verification, a user can assure themselves that no alteration occurred on a program prior to installation. The advantage to taking this approach to compare software is that the process is much faster. Otherwise, a user would need to compare line item by line item over each step of a program to see if there were any alterations. There are a number of freeware tools available on the web that let you calculate the hash value of a program. It might be an interesting extracurricular exercise for a person to try two different tools and calculate the hash value for the same software program. The resultant hash value should be identical. (Slavasoft, 2001) In like fashion, the application whitelist conducts a survey and generates a set of hash values for the programs residing on a computer. If a computer program does not provide the same hash value as expected by the application whitelist, the program will not have permission to run.

The example below demonstrates how to compute a SHA-1 hash value for a software program using the SlavaSoft, Inc., FSUM tool. Other tools are available that

perform the same function. The example program file is the Notepad++ installer program a freeware tool available from Notepad++. (Ho, 2011)

The software manufacturer lists the following SHA-1 digest on its webpage for the npp.6.6.3.Installer.exe program file :7ade7f5aa3a16c93238f125ae7020c09ef8b5df6

By opening and running FSUM against the installer program using a Microsoft® Windows® command line interface for example:

"C:\>fsum –sha1 npp.6.6.3.Installer.exe"

The FSUM tool lists the following SHA-1 hash value as:

7ade7f5aa3a16c93238f125ae7020c09ef8b5df6

The hash values match and therefore provide assurance of the integrity of the program prior to installation. If the values did not match, there would be no reason to run the program. In fact, it would be extremely important to determine why the values do not match. In a similar way, this manual process of verifying the hash value of a program is akin to what application the whitelist provides for the entire computer. Application whitelist software may use more than just a hash value of the program. It may also consider the size of the program, the path of the program, or other considerations in making a determination that the program may run.

The application whitelist is the polar opposite of antivirus software in that antivirus software examines programs to see if they contain a known virus. As part of the process, the program is allowed to run in a special cordoned-off area known as a "sandbox," and if the program contains code from a virus and it attempts to make unauthorized changes in the sandbox, the originating program is stopped and either quarantined or deleted. Antivirus software requires regular, often daily, updating of virus signatures in its effort to deal with viruses. In order to be effective, the computer must regularly update the database, and this means it must have access to the Internet.

Application whitelisting operates under the principle of deny by default. Namely, no program is allowed to run unless it explicitly has permission. Antivirus software, on the other hand, permits any program to run. It assumes all programs may run unless it

sees something that behaves or looks like a virus. Besides examining a program, antivirus software also schedules a time to conduct a full scan of the computer to look for viruses. It is up to the administrator to tell the antivirus software what it must do in the event that it finds something. There can be two issues for control systems actively running antivirus software. The first is there can be a very large need to use processing power during the time it scans. During this time, the HMI operation or even the data reporting back to an operations center may take longer than normal. The second issue is if the antivirus software accidentally quarantines or worse deletes a needed file because the antivirus software incorrectly determined that a file contained a virus.

To explain this difference a bit further, take for example, an HMI computer in use in a power substation. Further, assume the HMI computer is using application whitelisting. That means the computer will not permit any newly installed software to operate. Suppose, in this example, an employee or contractor attempts to install an MP3 player on that computer. They may transfer the MP3 player from a USB stick to the computer. But the MP3 player will not operate because the whitelist will not find the program on the permitted list.

Conversely for antivirus, if the employee installs the MP3 player on the HMI computer, they are able to operate the software as long as the software did not contain a virus.

In general, there would most likely be other protection measures preventing the MP3 player from being installed on the computer. For example, there might be specific group policies and/or read and write privileges. Even the third-party antivirus software or even application whitelist software may prevent the movement of program files from the USB stick to the computer.

From an operational perspective for control system computers, it is advisable to lock down the computer to the greatest extent possible and only allow those programs to run on the computer that are necessary for the computer to do its job. This is a core concept and has been discussed in multiple domains but applies here as given in the Access Control domain of the CISSP CBK, the security practice of least privilege. (Tipton, 2007) Locking down systems provides a minimal amount of security for

computer and application software and are good steps to take. This minimalist security philosophy works well alongside application whitelisting, but moving from antivirus to application whitelisting requires change, and change in the industrial control system workspace takes longer than other segments.

Surprisingly in August 2010, the National Security Agency (NSA) provided a "How To" document that explains step-by-step how to create a software restriction policy on a Windows® XP computer. The software restriction policy operates as an application whitelist for the computer. It is unfortunate that it is a relatively unused and unknown method to secure Windows-based computers. Even many IT administrators are not familiar that there are software restriction policies that apply to the Windows operating system. The NSA paper is a good first step for the reader to try out, for zero cost, and become familiar with the use of application whitelisting. (NSA, 2010)

3. Administration of Application Whitelisting Systems

One of the more problematic issues for a computer that operates on an isolated network is there is no means for obtaining regular or periodic updates. For traditional antivirus software, the update requirements can be as often as one update per day. One way to update isolated computers with antivirus software updates is to utilize a proxy file server and manually transfer updates to an isolated computer. But this can take a significant amount of labor and resources, e.g., driving out to a remote location like a power substation. This effort also presents problems with tracking and managing the portable storage media that contain the updates. Any portable media that moves from one environment to another increases security risks. In general, for a remote substation computer located hours away, it is not practical to update such a computer on a daily basis or even weekly basis. Again, this problem makes the use of application whitelisting more attractive because it does not need regular or periodic updates to secure a computer. Application whitelisting, therefore, lends itself very well to isolated or air-gapped networks, even more so if the computers are located in geographically disparate locations. Once a whitelist forms and is operational on a computer, there are little if any updates required. There are some chances that an update will need to occur, for example, to update the application whitelisting software. Also, if the operating system or

application software has an update, it is good practice to update these as well. But one benefit of application whitelisting is that it provides immunity and extends the time between updates since no code is able to run on the computer unless it has previously received permission. That means any malware, including zero-day malware, may attempt to take advantage of an unupdated computer, but the malware is not able to run. The malware that is attempting to exploit the lack of an update remains nonoperational or dormant. In dealing with the issue of "dormant" malware, it is important to note that some whitelisting software packages prevent installation as well as modification of any programs onto the computer. Application whitelisting administration requires allowing updates and patches while maintaining the security of the computer. Specifically, the application whitelisting software should provide a mechanism for provisioning that allows updates of appropriate programs that are running on the computer and not just the whitelisting software itself.

Examining the log files, is extremely important for application whitelisting administration. Expect to periodically review the log files on an HMI system running either whitelisting or antivirus software. It is important that the log files have some form of traceability back to the application whitelist. That means any event that the application whitelist prevents should be clearly identified in the event log. It is even better if the event provides a reason along with the time and date for the event log. One issue with isolated or air-gapped computers is that the logs from these computers cannot propagate from the HMI computer to a central collection point. For example, some application whitelist software support event notification via the Syslog protocol. Therefore, it is important on isolated computers to periodically examine or manually transfer the event logs.

Regarding the administration of some application whitelisting software, when a program needs a patch or update, for this to happen, the whitelist software may require disabling the whitelisting software and re-calculating the hash. In order to disable the whitelist, there must be some mechanism to secure the system during the time that the whitelist is disabled. Please note that not all application software requires disabling in order to perform an update. A solution at this point in time is to use antivirus software and conduct scans: one scan prior to the update and one scan after the update, and then

re-enable the whitelisting software. One example solution is to create a USB thumb drive with antivirus software. Make sure to update the thumb drive with the appropriate and current signatures or definitions. The first scan assures there were no viruses present on the computer prior to an update. The second scan shows the update itself did not introduce viruses. The two scans, one prior and one after an update, provide a clear document trail and process, and reduce risks. Note that this process still does not guarantee that a zero-day exploit previously loaded on the computer may not wake up and take action upon the disabling of the application whitelist.

The other consideration when the application whitelist is dormant is if a zero-day exploit were on the computer, it might germinate or run at the time. In this case, the whitelist is not able to restrict its operation as it is in the "update" mode. This means at this point in time, a zero-day malware can propagate. And it is important to note that for a zero-day exploit, even a dual scan with an antivirus software will not flag it as malware. A possible answer in dealing with zero-day exploits is to review log files and conduct an audit of files and folders installed onto a computer. Then by conducting a comparison of these results against a previous audit or logs, you may find evidence of an unauthorized change. The method is to use application whitelisting software that does not enter into a disabled state for an update or it does not allow for installation of files except by an approved system administrator. Information in the log files help by allowing comparison to file sizes or even the dates. Third-party software packages can automate the process for analysis as some provide this type of assessment. Namely, they allow the conducting of audits that compare previous results against one another. In this way, it might be possible to see anomalies that could be the harbinger of a zero-day exploit.

After installing the application whitelist onto a computer, it is a relatively simple process for the whitelisting software to hash the applications and memory. The process examines the files and creates a mathematical digest for the contents of the computer's C: Drive as it creates the whitelist. An administrative issue is the amount of time it takes to process the hash. The amount of time is dependent on the number and size of the volume(s) as well as the speed of the processor. In general, it can take between 45 minutes to two hours in order to create the whitelist.

Dwight Anderson, dwight_anderson@selinc.com

It is important to remember that the application whitelist software assumes that that the computer is clear and free from malware. There are a number of process steps to assure that the computer meets this criteria. As discussed earlier, the first step is to verify that the hash values of all the programs downloaded and added to the computer match those values provided by the vendor. Second is to scan the computer with an updated antivirus software prior to running the whitelist. These steps give assurances that the computer is free and clear of viruses.

4. Testing Application Whitelist

Testing an application whitelisted computer is not very difficult and should occur right after installation and periodically thereafter to assure the whitelisting software is operational. A simple test for this is once the computer is whitelisted, attempt to add and run an executable. Namely, install and attempt to run an executable that was not previously whitelisted. Any attempt to run the program should receive an error and should generate a log entry. In order to test that both the antivirus software and the whitelist software are protecting the computer, consider using the EICAR anti-malware test file. The EICAR test file, when unzipped, may quickly flag or alarm the antivirus program that you have introduced a virus onto your computer. Prior to working with the EICAR test file, take appropriate steps to follow your company's policies. It is advisable to notify your security personnel ahead of time. Then for whitelisted computers, attempt to run the EICAR file. It should not run, and it should generate an event log item. (EICAR, 1998) Informing your security or IT department of your activities along with finding an acceptable place to keep the test file helps avoid the scenario where one's own system and anti-malware software send out alarms and emails to various strata of your organization.

The following are some considerations regarding the methods that might be employed to attack a whitelisted computer system. The attack method depends a great deal on what type of system and whitelisting software is running. For a typical PC running Microsoft operating systems, the easiest way to attack a system is to have physical access. Once you have physical access, then it is a matter of booting up into safe mode and attempting to disable the whitelist. Without physical access, it is not certain at

this point in time that there are ways to compromise a whitelisted computer. Possibly gaining administrative access to the computer allows access to the whitelisting software, and then it might be a simple matter to disable such. Again, the end goal would be to shut down the whitelist, and then make the payload a part of the whitelisted programs or even have the whitelisting software appear as if it were operational but, in fact, is not. The downside to this action is that all the white listed networked systems would require compromise for a payload to operate and propagate. For some application whitelisting software, each whitelisted computer has its own set of hash values so these could not be copied and duplicated onto other computers. Given this situation, an attacker would have to compromise all computers in the path.

The number of platforms supported by application whitelisting software is fairly extensive and includes Windows and Linux operating systems. One software vendor, Savant for example, supports all of the following Windows platforms:

- Windows 7 (32- and 64-bit versions)
- Windows Server® 2008
- Windows XP
- Windows XP Embedded
- Windows Server 2003
- Windows 2000
- Windows WEPOS
- Windows Server 2000

5. Application Whitelisting and Zero-Day Malware

A number of application whitelisting software vendors state boldly that the software prevents the operation of zero-day exploits. This section of the paper will provide some details into the reasons behind these statements.

Zero-day malware is a program that takes advantage of a previously unknown vulnerability or bug found in a program or operating system. The 2011 Derbycon

conference dispelled the myth that zero-day exploits took intense research and highly technical abilities. During the conference, Terry McCorkle and Bill Rios shared a presentation about finding over 100 vulnerabilities in 100 days. (McClorkle, 2011) Further in 2012 at Digital Bond's S4 conference, McCorkle and Rios provided a training session on finding and exploiting HMI vulnerabilities. The net result of both of these presentations was that finding previously unknown vulnerabilities in poorly designed HMI software is not difficult. During the conferences, McCorkle and Rios demonstrated that it is not too difficult to have the system run their code (malware) based on exploiting the vulnerability. In fact, they had all participants attending the conference finding zero-day vulnerabilities in a matter of hours.

Keep in mind that all these vulnerabilities had no known antivirus signatures, and the programs that took advantage of the exploit would not be uncovered or quarantined by the antivirus software. As a follow-on exercise, not part of the conferences, application whitelisting did prevent the above zero-day exploits from running. Namely, there was no means to add the exploit code into whitelisted computer programs. Any attempt to run the changed program generated a whitelist event. Other papers, such as from Andrew Ginter, chief security officer at Industrial Defender Inc., at Foxborough, Mass., make very similar statements. (Iverson, 2010) But application whitelisting is one tool of many, and it is not good to assume that it will provide 100 percent protection against zero-day exploits. It just adds an additional layer of protection.

Protection against zero-day exploits is extremely important for large SCADA systems that often use servers and workstations. In these large SCADA systems, the file servers often use Windows 2003 or 2008 R2 operating systems. The workstations often use Windows XP or Vista operating systems. The file servers operate as data historians. Namely, they provide the repository for the historical data about the process. The workstation operates as the HMI and gives a graphical depiction of the process along with the most current values of the polled data. HMI displays also provide alarming when a process reaches an out-of-limit condition. Multiple workstations allow multiple persons to observe the process as well as provide the ability for one workstation to zoom into a process and to yield greater detail on the state of the process. In some cases, the data historian or file server operates from a geographically separate location from an HMI

workstation. For example, a data historian may be located at a command and control center while the HMI workstation is located miles away in a substation. The servers and workstations often have no Internet connection and remain on their own local-area network or subnet. With no security on the workstation or file servers, it is possible for an attacker to use a man-in-the-middle-style attack and change the process results in the workstation that feeds back fake data to the command and control center. In this scenario, the SCADA system may give the appearance that the process is operating within the control limits but in fact the process may be completely out of control.

Since a SCADA system's file server or workstation are static or do not get new or additional changes to the software, they are good candidates for application whitelisting. The application whitelist effectively locks down the data historian and HMI workstation; therefore, it would not allow malware or even zero-day programs to operate. The application whitelist would prevent the workstation from changing the data sent back to the SCADA command and control center, therefore thwarting a man-in-the-middle style attack. If a zero-day program attempted to operate on the HMI workstation, the application whitelist could send an event or alarm that notifies the command and control center that it detected an attempt to make a change to the HMI software. This is an extremely valuable security tool that needs serious consideration and application in SCADA systems.

6. Application Whitelisting With Non-Windows-Based Network Devices

Up until now, this paper has focused mostly on whitelisting on HMI computers that use a Microsoft Windows operating system, but what about the use of whitelisting with other operating systems, and more importantly, what about network devices? For example, what about Linux-based operating systems, and what about network switches and firewalls? The short answer is "yes," it is possible to use whitelisting with these non-Windows-based network applications with some considerations. A number of third-party application whitelisting software support Linux-based operating systems. For example, McAfee Embedded Control and Coretrace Bouncer support application whitelisting for

Linux-based systems. Also, there is support for Apple Mac even VM-based operating systems such as from Bit9.

McAfee—www.mcafee.com/us/products/embedded-control.aspx
Coretrace (Lumension)—www.lumension.com/coretrace
Bit9—www.bit9.com
Savant Protection—www.savantprotection.com

The same principles apply for Linux- and Mac-based systems as needed to secure Windows-based systems. Namely, there is the assumption that the device will not undergo change, which is generally true for control systems. Upon deployment or commissioning, the device will remain static. The software will generate a hash digest of the Linux- or Mac-based system and software programs. Subsequently, the hash values determine if the program or code has permission to run.

Most recently, application whitelisting for embedded systems has been incorporated into network devices, such as from Schweitzer Engineering Laboratories, Inc. (SEL) called exe-GUARD™. The development of the devices were part of the U.S. Department of Energy's Cybersecurity for Energy Delivery Systems (CEDS) program. The program develops cooperative projects between private and public enterprises to increase the security of critical infrastructure operations. In this case, exe-GUARD™ development was between Dominion Virginia Power (DVP), Sandia National Laboratories (SNL), and SEL. The program generated commercial application whitelisting for embedded network devices, specifically an Ethernet security gateway and firewall as well as an Ethernet-to-serial port server. The devices use application whitelisting on embedded operating systems to improve the security of critical infrastructure systems. (Schweitzer, 2014)

7. Conclusion

Dwight Anderson, dwight_anderson@selinc.com

Whitelisting is just one tool of many in a security professional's tool belt. The purpose of this paper is to encourage others to look at and evaluate the use of whitelisting, especially for those persons involved in securing industrial control systems. There are other security measures that a security professional should also consider as well, such as the use of firewalls or intrusion detection systems. Also, it is not the intent of this paper to suggest that whitelisting negates the need for antivirus software. Each security measure provides greater defense-in-depth and increases the security posture of an enterprise as well as for a critical infrastructure system.

Areas that might be of interest but were out of the scope of this paper include showing how well other security tools interact with application whitelisting software. For example, with the use of Microsoft's Enhanced Mitigation Toolkit known as EMET, there seems to be good interoperability and synergism using this tool with application whitelisting. A follow-on project might be to examine how well whitelisting and EMET protect a computer system's memory from a heap-spray style attack, for example.

Also of importance is to understand the interaction between application whitelisted computers and security tools. For example, if a company uses a Tenable Nessus vulnerability scanner, what is the interaction between that software and Nessus in providing a report or list of vulnerabilities? Initial testing seems to indicate that Nessus would not be able to access the computer and conduct a vulnerability scan. In order to do so, it would require whitelisting programs that the application whitelist prevents. This would be true of almost all vulnerability scanners, so it generates an administrative conundrum. An answer is to conduct the scan without the whitelisting software operating. Identify the vulnerabilities, and update accordingly, and then re-enable the whitelisting software.

The other area this paper did not explore but is of interest is delving into whether or not solving the compatibility issue between the application whitelisting software and a third-party HMI software package decreases the effective whitelist operation. Of concern is that the application whitelisting software must open up a security hole in order to get the third-party software operational.

Dwight Anderson, dwight_anderson@selinc.com

Next might be the determination of how well and quickly the manufacturers of application whitelisting software respond to security researcher findings. Specifically, do they have a process in place, and how long does it take to obtain a fix if there is a problem? Do they have an easy method to contact someone, and what is their disclosure process? This paper explained how whitelisting fits fairly well into static systems, such as those found in control system applications and provides much greater security.

8. References

Langner, Ralph, (2011, September). *Robust Control System Networks: How to Achieve Reliable Control After Stuxnet.* Momentum Press LLC, New York.

Guzmán, J. B. Mooney, G. Benmouyal, N. Fisher (2002, April). *Transmission Line Protection for Increasing Power System Demands,* 55th Annual Conference for Protective Relay Engineers, College Station, Texas.

Will Allen, (2008, September). *Effects of Wide-Area Control on the Protection and Operation of Distribution Networks,* Schweitzer Engineering Laboratories, Inc.

Harold F. Tipton and Kevin Henry, (2007). *Official (ISC)^2 Guide to the CISSP CBK*, ISC^2 Press/Auerbach.

United States of America, National Security Agency, (2010). *Application Whitelisting Using Software Restriction Policies*, retrieved March 9, 2014, from Information Assurance Directorate, Vulnerability Analysis and Operations, Systems and Network Analysis Center website: http://www.nsa.gov/ia/_files/os/win2k/Application_Whitelisting_Using_SRP.pdf

EICAR, (1998). *European Expert Group For IT-Security*, originally as, European Institute for Computer Antivirus Research (EICAR), Anti-Malware Test file, retrieved April 15, 2014, from http://www.eicar.org

Terry McCorkle and Billy Rios, (2011, September). *100 Bugs In 100 Days*, Retrieved April 10, 2014, from website: http://www.irongeek.com/i.php?page=videos/derbycon1/mccorkle-and-rios-100-bugs-in-100-days-an-analysis-of-ics-scada-software

Wes Iversen, Managing Editor, (2010, October). *Automation World, Defending Against The Next Stuxnet*, Retrieved April 18, 2014, from website:

Dwight Anderson, dwight_anderson@selinc.com

http://www.automationworld.com/energy-management/defending-against-next-stuxnet

Schweitzer Engineering Laboratories, Inc., (2014, January). *SEL Announces Embedded exe-GUARD™ Antimalware for Critical Communications Devices*, Retrieved April 19, 2014, from website: https://www.selinc.com/news.aspx?id=103431

Dwight Anderson, dwight_anderson@selinc.com