Investigating the Applicability of a Moving Target Defense for SCADA Systems

Cordell C. Davidson, Joel Dawson, Paul Carsten, Mark Yampolskiy, Todd R. Andel
University of South Alabama
*{ccd1421, jad1324, pwc1221}@jagmail.southalabama.edu, {*yampolskiy, tandel}@southalabama.edu

As computer attacks become more sophisticated and the rise of advanced persistent threats, moving target defense (MTD) is a new strategy being developed to reduce attacker success and provide for system resiliency. This work has been focused on standard IT infrastructures. In this paper we put forward how current MTD techniques may be applicable to the SCADA environment.

Security, Network Security, Moving Target Defense, Diversity Defense, Software Vulnerability Mitigation

1. INTRODUCTION

Supervisory Control and Data Acquisition (SCADA) systems are used to monitor and control industrial plant and equipment that is often part of a national infrastructure.

Reliability and availability is extremely important to these systems. Once they are up and running reliably, any changes to the hardware or software could introduce problems that affect reliability. It is impractical and expensive to take these systems down except in very infrequent and planned intervals. This often means that many of these systems rely upon hardware and software systems that are years or even decades in age. Even well-known software vulnerabilities may go unpatched for years as reliability and uptime of the system may be deigned to be more important than threat incurred by the software's security risk. Many systems are known to be vulnerable to common attacks.

This paper examines Moving Target Defense strategy and the applicability, if any, for use with SCADA systems given the requirements and constraints inherent in such systems.

2. MOVING TARGET DEFENSE

The Federal Cybersecurity Research and Development Program defines Moving Target Defenses as those strategies

> …. that aim to substantially increase the cost of attacks by deploying and operating networks and systems in a manner that makes them less deterministic, less homogeneous, and less static" (Executive Office of the President NSTC, 2011).

A Moving Target Defense (MTD) is a defensive security technique of periodically changing the attack surface of protected systems in order to make it more difficult for an attack to suceed. These techniques do not resolve any particular vulnerability but, instead, seek to make the vulnerability less probable of being successfully exploited.

In essence, a Moving Target Defense makes the assumption that perfect security is an illusion and thus focuses effort on having systems that provides resiliency in an insecure environment. They seek to provide systems that are defensible instead of perfectly secure. An exploit may still be possible on the defended system but the probability of a successful attack may be markedly reduced.

A MTD can be designed to probabilistically protect a host, a network, or a particular program and is additive to passive, active, and other defenses that may already be in place.

Moving target defenses can be implemented through varying mechanisms:

Address Space Layout Randomization (ASLR) seeks to randomize the memory layout of a processes address space to make it more difficult for an adversary to take advantage of known absolute and relative addresses (Shacham et al., 2004).

Instruction Set Randomization (ISR) introduces the idea to modify the underlying instruction set of a protected program so that any injected code is invalid and cannot be executed (Barrantes et al., 2003).

Data Space Randomization (DSR) is a technique for scrambling or encrypting data that is used in a program. Unlike instruction set randomization, data

randomization seeks to encrypt or randomize user data stored on the heap or the stack. Program instructions are not encrypted (Cadar et al., 2008).

IP Hopping is a MTD networking defense that seeks to periodically and randomly alter the IP address of a host or group of hosts in order to confuse an adversary and disrupt any attack (Dunlop et al., 2011).

Configuration Randomization (CR) is a MTD implemented by modifying, in a seemingly random manner, the initial and load time configuration parameters that govern changeable aspects of a program or operating system. By manipulating which group of parameters is set along with the values for each particular one, and by periodically changing which parameters are loaded and what values they are set to, the attack surface of the host system can be altered in such a way that a Moving Target Defense is deployed (John et al., 2014).

3. MTD AND SCADA SYSTEMS APPLICABILITY

The moving target defense paradigm has been demonstrated to be a useful model for the development of security tools that can mitigate malicious attacks. However, the field is young and there remains much more research to be done. An area that has not received much, if any, attention is the applicability of MTD techniques to SCADA systems.

Simply migrating current IT solutions that have been designed for use with common corporate and governmental computers and networks do not take into consideration the unique requirements and constraints that are inherent in SCADA systems (Krutz, 2005) such as the following:

(i) Loss or even the interruption of data may be intolerable.
(ii) Systems must be extremely resilient and reliable.
(iii) Require deterministic times in control loops.

A failure of the first requirement could lead to extensive damage or even the loss of life.

The second requirement is necessary because these systems often cannot reboot for routine patches and other maintenance. It may be months or years between software patches or hardware upgrades yet the system must continue to perform its' task without failure.

The third requirement, in many ways, is the core constraint that the other requirements are based upon. These machines monitor and control real-time systems or are themselves real-time systems that require deterministic times in their control loops. Delays in receiving or processing data cannot be tolerated beyond a particular threshold that may vary between systems. This often negates the use of virus software protection, encryption, and other common tools. Anything that causes a delay could be regarded as unnecessary or dangerous and not be allowed.

In a regular IT system the first lines of defense are to patch known vulnerabilities on a regular basis, run virus software, encrypt important data, and implement other security measures that may not be allowed on a SCADA system.

Moving target defenses, in contrast, seem to have some suitability to this type of environment. Once a MTD has been designed, tested, and applied to a SCADA system it can continue to provide some measure of probabilistic protection regardless of how long it goes between changes. Obviously, there is a trade-off between the frequency of the system reconfiguration and its impact on both the system performance and security.

Below, we discuss the applicability of existing MTD techniques to various components of SCADA systems. Table 1 summarizes our premise for each moving target technique and the components of a SCADA system to which it may apply.

Table 1: *MTD Techniques / SCADA Components* APPLICABLE (A), CONDITIONALLY APPLICABLE (CA), NON-APPLICABLE (NA)

MTD	**SCADA Component**			
	PC/HMI	*MTU*	*RTU/PLC*	*Field Units*
ASLR	*A*	*A*	*CA*	*CA*
CR	*A*	*A*	*CA*	*CA*
DSR	*A*	*A*	*CA*	*CA*
ISR	*NA*	*NA*	*NA*	*NA*
IP Hopping	*CA*	*CA*	*CA*	*CA*

Address Space Layout Randomization (ASLR) for use on a host operating system assumes that initial efforts would be directed at protecting the main control center. Often Linux or Windows are the operating systems in use. Both of these have ASLR built into them. As long as the program to be protected has been compiled with the proper compiler and linker flags, any application can have host level address space layout randomization easily applied. Vendors of HMI and other control center software may be required to provide an updated application, but once that has been tested and installed, no further changes are required. Every time the application is loaded, re-randomization of memory locations will occur. Granted, this may not be very often but even the initial load will move memory from the static

locations that had been used before. With address space layout randomization an adversary cannot successfully direct a canned attack against a known vulnerability in the software. They must now spend additional time and effort that was not previously required.

ASLR at the individual program level requires that each protected program arrange its own movement of stack, heap, libraries, functions, and other memory management in order to provide a finer level of control than given by host based ASLR. Though the commitment of highly skilled developers to provide a custom program may be required, once developed these programs should provide the same type of benefits as a host based ASLR.

The main control center of a SCADA system may employ several different computers (HMI, MTU, data historian) running a version of Windows, UNIX, or Linux with the capability of ASLR. It is unlikely that a RTU/PLC or the controlled field units will have this capability built in. It may be possible to develop a program for some of these devices to implement ASLR at a program level but this seems a bit impractical and premature. The safer approach is to test with the more powerful machines at the control center and let any success trickle down to the remote sites if, at that time, it is deemed feasible.

Address space layout randomization implemented on PC's at the control center is expected to add an extra level of while still meeting SCADA requirements of (1) no loss or interruption of data, (2) reliability and resiliency, and (3) determinism. If ASLR can be implemented on a particular RTU/PLC then it would be expected to also meet these requirements.

Data Space Randomization (DSR) would be implemented in a similar fashion to ASLR for an individual program. A great deal of expertise is required for custom modification of an application. This could be a significant undertaking. Still, once the program has been completed and tested to ensure that it is both correct and that it does not negatively impact the three SCADA requirements, it can be installed and should provide a similar level of movement as discussed with ASLR. This MTD methodology looks promising for protecting individual programs with a high security risk. The control center is the most likely area to find such programs though it may also be used in any program that runs on a system that can afford the time to XOR the data against a key.

Instruction Set Randomization (ISR) is likely to impact performance too greatly to be effective at this point in time. Though hardware based version using a FPGA as a register might increase speed enough to be used on a machine in the control center, this is a technique still under development (Andel et al., 2014) and not yet amenable to integration into any SCADA system component.

Configuration Randomization (CR) could be attempted on any SCADA component that has initial or runtime configuration parameters that can be modified enough to cause an effective change in the attack surface of the host operating system or an individual program. Since it is likely that the OS or program cannot be reloaded very often, we would expect little movement based solely upon initialization parameters. If there are significant runtime parameters that can be safely changed, then movement would be increased. Randomization with this method could also be used for any program that runs on more than one machine in the SCADA system. For example, if many of the remote sites run the same program and that program can be configured many different ways while maintaining full functionality and meeting the three identified SCADA requirements, then even if each individual program cannot be reconfigured often at each particular site, a MTD defense of the system as a whole is provided by configuring each program at each site in a different manner. An attacker would have to vary their attack at every location which adds time, cost, and additional chance of discovery. This would also be the case if implemented across different SCADA systems. Each one would be somewhat different thus requiring individual attention by an adversary and negating many generalized attacks.

IP Hopping might be effective but should be thoroughly tested to ensure that there are no delays in communications and that each node can continue to communicate as necessary even when the address is changed. If this can be accomplished, then this scheme could be used in every part of a SCADA.

Some types of moving target defenses implemented within a SCADA system may be provided more easily than others. Though we are still gathering data concerning PLC's and embedded systems, there is an expectation that it may be possible to implement several different MTD techniques on ones that meet the specific requirements of a particular moving target defense. However, a modern operating system on a workstation or server enables the easiest and most diverse implementations of MTD in SCADA systems.

4. SUMMARY

Supervisory Control and Data Acquisition (SCADA) systems are the backbones of our national infrastructures yet, for various reasons, lack many of the security features of a common business network. Providing additional security has become

a priority but the particular safety, reliability, and functionality needs of each system precludes simply adding these aspects of security. It will be many years before the current SCADA generation is updated with the security features that are already needed today.

In researching other methods of mitigating these security needs, we looked into the feasibility of implementing a Moving Target Defense (MTD). These types of defenses do not seek to provide perfect security. The assumption is that no such thing exists. Instead, it seeks to make any known or unknown vulnerability less likely to be exploited. The key issue to be addressed is whether a particular moving target defense can be implemented without affecting the core safe, reliable, deterministic, and correct operations of the SCADA system.

Our investigation raised suspicion that the additional processor load normally associated with Instruction Set Randomization (ISR) was likely to interfere with determinism requirements of a SCADA system. No further investigation into this technique with SCADA is planned at this time.

Data Space Randomization (DSR), Configuration Randomization (CR), Address Space Randomization (ASLR), and IP Hopping may be feasible as security implementations for specific programs or network segments. Each of these will require a significant investment in time and money to develop though the additional security provided may overshadow the costs. However, once they are developed and thoroughly tested to ensure no ill effects to the SCADA system as a whole, then they will provide some level of defense even against zero day attacks.

We believe that Address Space Layout Randomization (ASLR) at the host level appears to be the most promising avenue for initial adoption in a SCADA environment. Modern operating systems have ASLR built-in and require only that each program be compiled with the appropriate compiler and linker flags. Once this is done, protection is provided automatically by the operating system. Whenever the program is reloaded, the memory layout will change thus achieving a moving target defense. Even though reloading the program may be uncommon in the SCADA system, it will still change the program's memory locations from the static and easily determined locations it currently loads. In addition, every other program running on different machines will have a different memory layout. A generic attack against vulnerabilities in the program is not likely to succeed. Note that similar protection is provided even against zero-day attacks.

As next, we plan to verify our premise using simulated SCADA system as well as apply and evaluate selected MTD techniques to PLCs in a laboratory setting. Furthermore, we plan to document the effects of various MTD techniques on the overall system security. For the detailed description of changes and their effects, we plan to use Cyber-Physical Attack Description Language (CP-ADL) (Yampolskiy et al., 2015), which is specifically dedicated to reflect causal relationships between changes and effects.

5. REFERENCES

Andel, T. R., Whitehurst, L.N., McDonald, J. T. (2014) Software Security and Randomization through Program Partitioning and Circuit Variation*. Proceedings of the 1st ACM Workshop on Moving Target Defense*. ACM.

Barrantes, E., et al. (2003) Randomized Instruction Set Emulation to Disrupt Binary Code Injection Attacks, *Proceedings of the 10th ACM Conference on Computer and Communications Security.* ACM.

Cadar, C., Akritidis, P., Costa, M., Martin,J., Castro, M. (2008) *Data Randomization*. MS Research. http://research.microsoft.com/apps/pubs/default.aspx?id=70626. Accessed Sep 23, 2014.

Dunlop, M., Groat, S., Urbanski, W., Marchany, R., Tront, J. (2011) MT6D: A Moving Target IPv6 Defense, *Military Communications Conference*, 2011 - MILCOM 2011. IEEE., 1321-1326.

EOP NSTC. (2011) *Trustworthy Cyberspace: Strategic Plan for the Federal Cybersecurity Research and Development Program.* http://www.whitehouse.gov/sites/default/files/microsites/ostp/fed_cybersecurity_rd_strategic_plan_2011.pdf. Accessed March 19, 2015.

John, D., Smith, R., Turkett, W., Canas, D., Fulp, E. (2014) Evolutionary Based Moving Target Cyber Defense, *Proceedings of the 2014 Conference Companion on Genetic and Evolutionary Computation Companion*. ACM, 1261-1268.

Krutz, Ronald L. (2005) *Securing SCADA Systems*. Wiley Publishing.

Shacham, H., Page, M., Pfaff, B., Goh, E., Modadugu, N., Boneh, D. (2004) On the Effectiveness of Address-space randomization, *Proceedings of the 11th ACM Conference on Computer and Comm Security*. 298-307. ACM.

Yampolskiy, Mark, et al. (2015) A language for describing attacks on cyber-physical systems,. *International Journal of Critical Infrastructure Protection* Volume 8, 40-52.