INL/CON-08-13946
PREPRINT

Idaho National Laboratory Supervisory Control and Data Acquisition Intrusion Detection System (SCADA IDS)

2008 IEEE International Conference on Technologies for Homeland Security

Jared Verba
Michael Milvich

May 2008

Idaho National Laboratory Supervisory Control and Data Acquisition Intrusion Detection System (SCADA IDS)

Jared Verba, Idaho National Laboratory, PO Box 1625, Idaho Falls, ID, 83415-2604, (208) 526-6120, jared.verba@inl.gov
Michael Milvich, Idaho National Laboratory, PO Box 1625, Idaho Falls, ID, 83415-2604, (208) 526-6109, michael.milvich@inl.gov

Abstract*—Current Intrusion Detection System (IDS) technology is not suited to be widely deployed inside a Supervisory, Control and Data Acquisition (SCADA) environment. Anomaly- and signature-based IDS technologies have developed methods to cover information technology-based networks activity and protocols effectively. However, these IDS technologies do not include the fine protocol granularity required to ensure network security inside an environment with weak protocols lacking authentication and encryption. By implementing a more specific and more intelligent packet inspection mechanism, tailored traffic flow analysis, and unique packet tampering detection, IDS technology developed specifically for SCADA environments can be deployed with confidence in detecting malicious activity.*

1. Introduction

Traditional information technology (IT) measures of monitoring and securing Supervisory, Control and Data Acquisition (SCADA) and Control System networks perimeters have used Intrusion Detection System (IDS) appliances or software that examine traffic on the network and wait for events to occur. Current IDSs rely on signatures and anomaly detection to find malicious packets, scanning attempts, and denial-of-service, and man-in-the-middle (MITM) attacks. These systems are a good start for securing a SCADA network; however, another degree of granularity is required to understand how the systems operate and how to protect them and the infrastructure they operate. This paper describes common IDS technologies, where they work and where they need to be improved, to cover this new vulnerable SCADA environment.

2. Current Technology

Two broad type of intrusion detection technology are deployed today: those that detect anomalies in the network and those that match signatures. Anomaly-based IDSs usually fall into three separate categories: behavioral, traffic pattern and protocol [1]. These systems classify network activity as either normal or anomalous and rely heavily on heuristics to make a determination on the traffic's intent. A signature-based IDS can only detect malicious traffic or network misuse if a signature has been explicitly created for detection. Signature-based IDSs rely on a series of patterns that malicious traffic may or may not match. When a signature- or anomaly-based IDS discovers a match or classifies a piece of network traffic as suspicious, alerts are generated and sent to either a central logging facility or logged into a file locally. This approach to securing networks works very well for popular network protocols when used in common IT infrastructure. Malicious operating system attacks, web exploitation, and virus-laden packets are detected easily and quickly with current IDS technology.

Vendors have recognized a gap in the applicability of IDS technology to control system environments, and have started to create additional modules for their products targeted at the common protocols used in SCADA and control system protocols including Modbus®, DNP3 (Distributed Network Protocol), and Inter-Control Center Communications Protocol (ICCP) [2]. While these additions represent progress, these IDS products still lack the SCADA protocols found at the innermost networks segments—the protocols that make the system function [3]. Handling these protocols presents a larger challenge, as they are proprietary and are often undocumented or ported from insecure serial protocols to an Internet protocol (IP) network stack.

Signature- and anomaly-based IDS technology was originally developed for the IT realm. The techniques applied in that environment have been slowly migrated to SCADA networks. These methods work to an extent but cannot reliably detect a sophisticated attacker. Current anomaly-based IDSs use heuristics to ensure traffic conforms to a known baseline. [4] This method works well if a well-defined baseline is generated. IDS technology based on a signature matching has its own strengths and weaknesses also. It can detect suspect or malicious packets, but is limited by the signatures defined. Current IDS technologies have a chance at detecting anomalous activity (network scans, malformed packets, operating system-level attacks), but the lack of authentication in the SCADA environments means there is nothing to stop an attacker from forging real and correct commands or data feeds. This method can be used to confuse a system or leave it in a critical state without employing those attack techniques that IT-driven IDSs were developed to detect.

3. Future SCADA IDS Technology

To leverage existing intrusion detection technology for successful deployment into a SCADA network a next generation IDS must adhere to three simple principles;

- Able to match signatures and map specific network traffic.
- Monitor traffic and generate flow analysis on each link while understanding the protocols used.
- Evaluate traffic inconsistencies in session traffic and determine if that traffic has changed from the last time it was sampled.

Many of these environments have legacy systems integrated into them that cannot be upgraded, patched, or protected by traditional IT means. These points are critical to protecting systems that rarely use encryption, authentication, or bounds checking integrated into the network communication. It is possible to build a SCADA specific IDS to meet these goals.

4. Signature Matching

Signature- and anomaly-based IDSs use deep packet inspection to look inside the application layer of most IT-centric protocols to verify the data is within normal bounds and not malicious. This includes malformed data packets, network scans, and command shell prompts returning from exploited machines. This technology needs to be extended in the SCADA world to look inside the protocols that make these systems work.

Current SCADA protocols are usually undocumented, proprietary, and used primarily to move data and commands around the internal SCADA network. Before large network architectures were implemented into the SCADA environment, current protocols were sent over serial lines, modem lines, and any other point-to-point technology that was available. These protocols were developed to move data effectively, be processed quickly, and have minimum overhead on the already taxed processors of the time. Bounds checking and authentication were unnecessary extras not implemented within these simple protocols to improve efficiency.

As large networks were rolled out, the older, point-to-point protocols were converted for use on packet switched networks by simply pre-pending transmission control protocol (TCP)/IP headers. This allowed the use of the legacy protocols on modern architectures and over greater distances. Traditional signature- and anomaly-based IDS implementations may be able to detect a malformed packet crafted by an unskilled attacker, but that packet is unlikely to result in a direct action on the SCADA network. Packets that are correctly formed and contain valid data within them or data that would overflow an internal buffer and have a direct effect are a larger concern. An IDS designed to understand SCADA protocols would be able to monitor for correct packets between hosts and alert on transactions that would be outside of a protocols bounds. Overflows or improperly formatted packets could result in exploitation of a network device.

Using current deep packet inspection techniques to extract protocol state information would allow data including commands and point status to be interpreted and compared for improper format or data outside designated bounds. This type of detection is particularly effective against an attacker that is fuzzing a SCADA protocol in order to gain more knowledge while exploiting the system. Fuzzing the protocol, or sequentially testing fields within the protocol, can result in crashes and overflows from poor bounds checking and data handling of the older protocols. Figure 1 shows an example of a normal packet that consists of a header, message type and a message length. Data follows the length field and a correct packet never has length over a specific size so as to not overflow an internal static buffer inside the receiving application or device. The incorrect packet looks like a legitimate data packet; however the length is well beyond the length for a correct packet. If this packet were sent, the internal static buffer would be overwritten outside the normal bounds and a classic overflow condition is exploited. An IDS looking at layer 4 and higher of the network stack for correct field lengths within the packet would catch fields potentially exceeding the static buffer length, alerting to potential buffer overflows within packet.

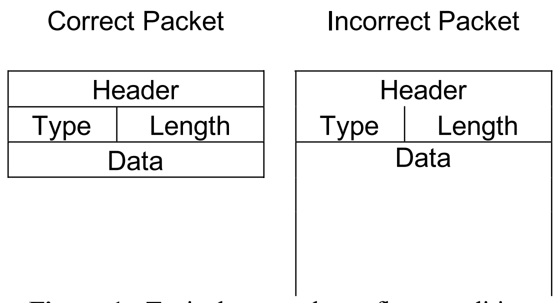

Figure 1 - Typical protocol overflow condition

5. Network Traffic Flow Analysis

IDSs in a SCADA network have at least one advantage over IDSs deployed in the corporate environment. SCADA networks are static, both in the sense of network topology and the tasks performed on the network. Servers and networking hardware are infrequently added to a system once it goes live; they also serve one single purpose, which is running SCADA software. Therefore it is possible within a SCADA network to map out all of the network traffic flows. This network map would contain information about which hosts talk to each other, which includes the ports, protocol (TCP or User Datagram Protocol [UDP]), and the host that initiated the connection. This information can then be programmed into an IDS that will watch the traffic flows of the SCADA network and trigger an alert on any traffic flows that do not match the normal traffic patterns.

An example of a simple SCADA network is shown in Figure 2, and the corresponding rules are found in Table 1. Each valid connection, shown in black, has an entry in the IDS that allows the connection to occur without triggering an alarm. The invalid connections, shown in red, will be collected in the default rule that will trigger an alert. In the normal course of operation, a Human Machine Interface (HMI) or an operator console should have no reason to connect to the front end processor (FEP). In this scenario, such a connection would cause the flow-based IDS to send out an alert.

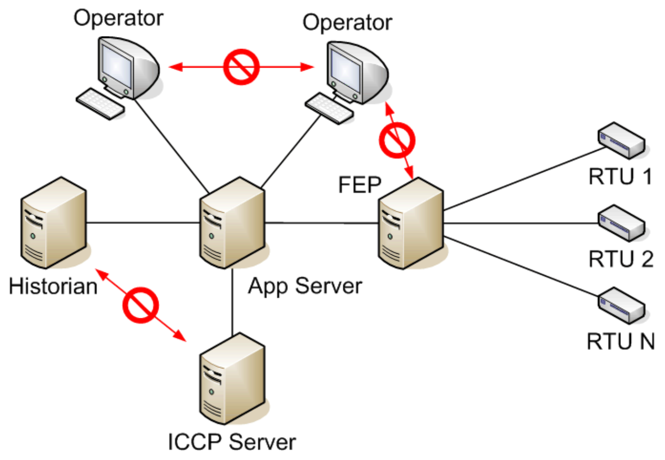

Figure 2 - Typical SCADA Network

Table 1 - Sample Traffic Flow IDS Configuration

Source	Destination	Protocol	Action
Operator	App Server	HMI	Allow
Historian	App Server	Data API	Allow
ICCP Server	App Server	Data API	Allow
App Server	FEP	Control API	Allow
FEP	RTU1, RTU2, RTU3	DNPv3	Allow
*	*	*	Alert

A flow-based IDS will catch any attacker that attempts to scan the network, no matter how slow or which obfuscation techniques are used. A network scan enumerates hosts and the network services running on the targets of the scan. Since rules are in place defining which hosts can communicate, scanning the network to discover hosts would trigger alerts as previously undefined connections are attempted as shown in Figure 3. When enumerating network services, a scan will attempt connections to ports not specifically defined, resulting in an alert from the IDS. For example, if an attacker forges an IP address, their forged IP address will not match one of the predefined rules and an alert will be generated by the IDS.

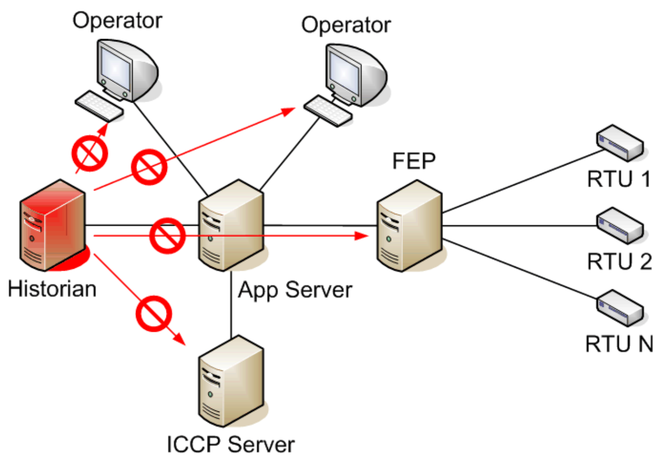

Figure 3 - Attacker Scanning the Network

In addition to detecting network scans, the traffic-based IDS will detect if a host tries to bypass parts of the system. For example, the FEP is a simple device that translates protocols and forwards commands onto remote devices. It will process and forward commands received from the application server, as well as any commands it may have received from the ICCP server. Because the ICCP server has no operational reason to talk directly to the FEP, the traffic-based IDS will detect this otherwise valid action and trigger an alert.

The configuration for the traffic-based IDS can be created by statically analyzing the hosts and the services running on them. It would also be possible to create these rules automatically by observing a correctly functioning system for a period of time and recording the traffic flows of a system in a normal state. This method is similar to anomaly-based IDS, which uses a baseline for the network to detect anomaly activities.

6. Network Data Inconsistencies

The previous IDS strategies try to detect anomalous connections (Traffic Flow Analyzer) and malformed packets (Deep Packet Inspection) but will not detect man-in-the-middle (MITM) attacks that can occur on a compromised server. Current IDS technologies will also miss detection of syntactically valid commands but are not correct or possibly malicious for the system. For example, if an attacker has compromised the FEP. From this location the attacker can send commands to the Remote Terminal Units (RTU) to carry out a malicious action. All of the commands transmitted by the attacker are syntactically valid, but are probably incorrect for the current state of the system. A traffic flow analyzer would have a rule to allow it, and the deep packet inspector would see that the command is formatted correctly; however, there is still the question of whether or not the command is valid.

It should be possible to answer this question and detect many of these MITM attacks by building a real-time database of the points and commands transmitted on each network stream. By comparing these databases, it should be possible to detect the inconsistencies that a MITM attack would create.

SCADA networks are regimented in how point data, data from remote hosts, and commands are distributed throughout the network. The point data is collected from the remote devices and is funneled through the FEP. The FEP translates the point address, optionally performs a scaling operation on the data, and forwards the resulting point data to the application server. The application server stores the point data in a real-time database. The data is then provided to its clients, such as the operator consoles (HMI), data historians, and ICCP servers. Operator commands are processed in the reverse manner. When initiated by the operator, a command flows from the operator's computer to the application server. The application server then forwards the commands to the FEP, which forwards the commands to the RTU. In addition, the application server can send commands on its own, based on its own internal logic and state machine.

As a consequence of the SCADA architecture the commands and data in the system exist in multiple locations at nearly the same time. In the simplest case, data travels from the RTUs, to the FEP, and onto the application server. It is then sent out to the operators, the ICCP server, and to the historian. This provides at least 5 copies of the data. By watching the traffic streams and building a per-network stream database of point data and commands, it would be possible to compare the data and watch for inconsistencies that will show up in the data streams when an active MITM attack occurs or if an unauthorized command is sent by a host that does not have the authorization to generate commands on their own.

For example, if an attacker managed to compromise the FEP, he would be able to send commands to the remote devices independent of the application server and would be able to forge point data returning back to the application server to cover his or her tracks and blind the operators, as shown in Figure 4. In this example the compromised FEP sends a command to RTU2 to open a breaker (Bk1). Being a simple device, the RTU will trust that the command is legitimate and will then open Bk1 before sending back a status update indicating that Bk1 is open. The attacker can either drop this status update or forge a status update to the application server reporting that Bk1 is closed. There are two data inconsistencies that appear in this scenario. First the FEP sent a command to RTU2 to open Bk1 without a corresponding command from the application server. FEPs are not intelligent devices and should never be used to send a command without receiving a corresponding command from the application server. The second inconsistency shows up when the attacker either drops the status update or forges the status update to the application server. With a per-network- stream database, it should be able to detect both of these inconsistencies and generate an alert.

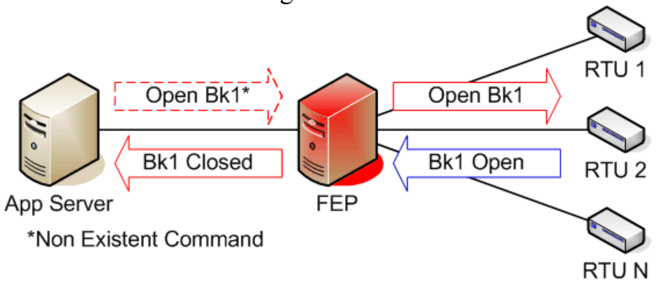

Figure 4 - Compromised FEP Sending Rouge Commands

An attacker with control over the FEP could also filter incoming commands and drop commands that might interfere with his or her attack, as shown in Figure 5. In this case the attacker dropped the command to open Bk1 on RTU2. Because the RTU2 never got the command to open Bk1, its status update will always say that Bk1 is closed. In order to complete the illusion, the attacker must also forge status updates to the application server indicating that Bk1 was indeed opened. This again creates two data inconsistencies that can be detected by maintaining and comparing per-network-stream databases. The First inconsistency is that the application server sends the FEP a command to open Bk1. The FEP does not have the intelligence to make a decision and drop the command, so the fact that the command shows up in the network stream from the application server to the FEP but not in the network stream from the FEP to the RTU2 indicates that something is wrong and may indicate a MITM attack. The second inconsistency is the point value on the network stream between the FEP and RTU2, which shows that the breaker is closed, and between the FEP and the application server, which shows that the breaker is open. Either of these inconsistencies can indicate the presence of an attacker performing a MITM attack and can be detected by comparing the databases for each network stream.

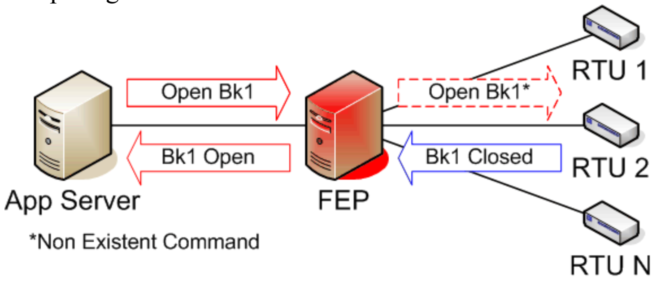

Figure 5 - Compromised FEP Droppings Commands

If an attacker has compromised the application server, detecting a MITM is not as simple as watching for inconsistencies in commands. The application server is intelligent and can initiate independent actions, and it can also drop commands from the operator that are deemed to be incorrect or could cause an unsafe condition. In the case of a compromised application server, the anomaly tracking IDS would rely on point-data inconsistencies rather than watching commands. For example, if the attacker decided to open Bk1 on RTU2, as shown in Figure 6, he would send a command to the FEP to open Bk1; the FEP, being a simple translator and forwarding device, will forward that command to RTU2. RTU2 will respond saying that Bk1 is now open, and the FEP will relay the corresponding status update to the application server. The attacker, wishing to cloak his or her actions, will then mutate the data and report to any consumers of the data, such as the operator and historians, that Bk1 is still closed. In this example there are four network streams: FEP to RTU2, application server to FEP, operator to application server, and historian to application server. Two of the four network streams have the correct status value for Bk1, which is FEP to RTU2 and application server to FEP, and the other two network streams have the incorrect status value for Bk1. This leads to a network data inconsistency that can be detected.

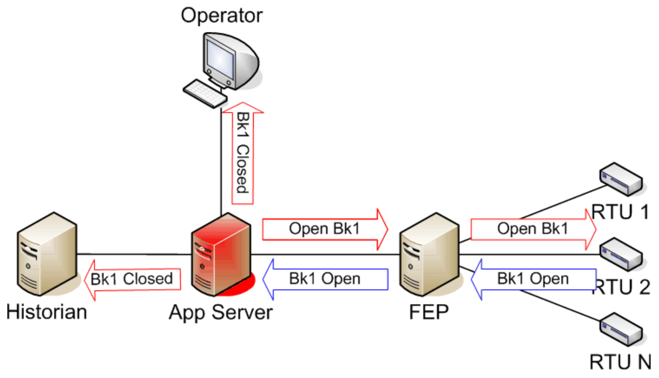

Figure 6 - Compromised Application Server Falsifying Information

The previous examples all dealt with host-based MITM attacks, where an attacker had compromised a host and used that host's network privileges. This was performed in accordance with the assumption that the traffic flow-based IDS would be able to detect most network-based MITM attacks. Even if it cannot detect the attack, the network data inconsistency IDS would still be able to detect inconsistencies that appear in a network-based MITM. This is shown in Figure 7, which is an example of how an attacker compromises the FEP and dropping commands. The same inconsistencies will show up. The only difference is that there is an extra network stream; thus, there would be three databases to look at instead of two.

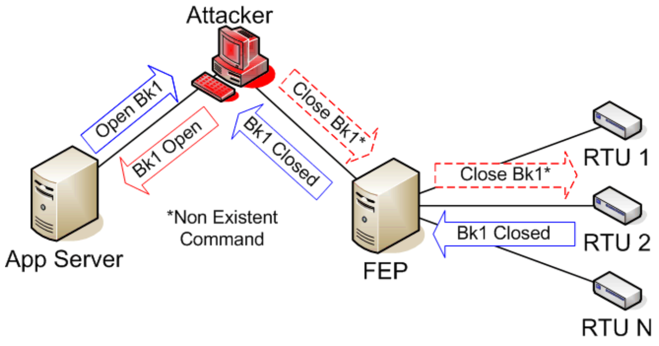

Figure 7 - Network MITM Attack

7. Conclusions

Current IDS sensors have difficulty working within a SCADA environment due to the lack of protocol understanding, which a traditional IDS would use to identify malicious network traffic. To address the legacy, proprietary traffic, newer and more granular methods that leverage the static nature of the network need to be developed to protect these critical, unsecured environments. A system implementing signature matching, flow analysis, and data inconsistency detection tuned specifically for SCADA and process control environment should be developed.

Tuned signature matching on the protocol can detect fuzzing and potential exploitation vectors not addressed in current IDSs. Analyzing the data and command flows within a SCADA network can be used to detect unauthorized commands or MITM attacks between devices. While each detection methodology has its own strengths and weaknesses in protecting the network, a hybrid approach is able to cover the gambit of common attacks against a SCADA system.

8. References

[1] Bejtlich, Richard. The Tao of Network Security Monitoring, Pearson Education Inc, 2005

[2] "Nessus SCADA Plugins." 04 Feb 2008, Digital Bond. Retrieved March 04, 2008, http://www.digitalbond.com/index.php/category/nessus-scada-plugins/

[3] Peterson, Dale. "Intrusion Detection and Cyber Security Monitoring of SCADA and DCS Networks." Presented at ISA Automation West. 2004. Retrieved March 04, 2008, http://www.isa.org/filestore/Division_TechPapers/GlassCeramics/TP04AUTOW046.pdf

[4] "The Science of Intrusion Detection System Attack Identification." 2002, Cisco. Retrieved March 04, 2008, http://www.cisco.com/warp/public/cc/pd/sqsw/sqidsz/prodlit/idssa_wp.pdf