Chapter 7

PROVIDING SITUATIONAL AWARENESS FOR PIPELINE CONTROL OPERATIONS

Jonathan Butts, Hugo Kleinhans, Rodrigo Chandia, Mauricio Papa and Sujeet Shenoi

Abstract A SCADA system for a single 3,000-mile-long strand of oil or gas pipeline may employ several thousand field devices to measure process parameters and operate equipment. Because of the vital tasks performed by these sensors and actuators, pipeline operators need accurate and timely information about their status and integrity. This paper describes a real-time scanner that provides situational awareness about SCADA devices and control operations. The scanner, with the assistance of lightweight, distributed sensors, analyzes SCADA network traffic, verifies the operational status and integrity of field devices, and identifies anomalous activity. Experimental results obtained using real pipeline control traffic demonstrate the utility of the scanner in industrial settings.

Keywords: Pipeline control systems, situational awareness, ROC protocol

1. Introduction

Imagine flying a modern aircraft with 10% of the instrument panel indicators providing erroneous data. Now imagine controlling a pipeline running from the Texas Gulf Coast to New York City with approximately 100 million pounds of liquids or gas, but with 10% of the field devices either non-operational or of dubious integrity.

Based on our experience, this is sometimes the situation with sensors and actuators in oil and gas pipelines. The sensors measure key process parameters such as pressure, temperature, flow and compositions. The actuators perform vital tasks such as opening and closing valves, operating pumping stations and tripping circuits. Pipeline operators must be able to trust their SCADA devices [14]. Unfortunately, few, if any, tools are available for verifying the status and integrity of SCADA systems.

This paper describes a SCADA network scanner intended to provide oil and gas pipeline operators with a comprehensive view of network topology along with detailed information about the configuration, status and integrity of SCADA devices and communication links. The scanner architecture incorporates a command module and database located in the control center and sensors positioned within SCADA subnets. The sensors passively monitor traffic and send information to the command module. The command module configures sensors, interacts with the database and provides event updates to operators. The database organizes, correlates and archives data collected by the sensors.

Tests using real pipeline control traffic demonstrate that the scanner can remotely verify the status and integrity of SCADA devices, profile normal SCADA operations and identify anomalous activity. The current implementation is targeted for ROC [3], a popular pipeline control protocol; however, the modular design readily accommodates other SCADA protocols.

2. ROC Protocol

The Remote Operation Controller (ROC) Protocol is used extensively in the oil and gas sector for pipeline operations. ROC is a proprietary protocol maintained by Emerson Process Management [3]. It is used primarily in Emerson products; however, other vendors often implement the ROC protocol to facilitate interoperation with Emerson equipment [11, 12, 16].

The ROC protocol uses a request-response paradigm for message transmission between a master terminal unit (MTU) and remote terminal units (RTUs) [2]. The MTU sends request messages to outlying RTUs to gather monitoring data or to specify control actions. The RTUs collect discrete and analog sensor data and maintain actuator settings specified by the MTU. Response messages are generated by RTUs after direct requests from the MTU. The MTU resends a request when it does not receive a timely response from an RTU. This communication model addresses congestion control and transmission error recovery.

ROC devices maintain control specifications and flow measurements within a database. The data elements, called “points,” represent single input or output values [1]. Each database parameter is uniquely identified by a parameter number and point type. A request message from an MTU specifies a function to perform and the associated parameter number and point type. The receiving RTU performs the operation for the specified parameter and sends the desired measurement or a confirmation that the control action was performed. The ROC protocol specifies access mechanisms for the database configuration, real-time clock, event and alarm logs, and historical (archived) data.

Figure 1 shows the structure of a ROC message. A message contains a Destination Address followed by the Source Address. The addresses are split into two components: Unit ID and Group ID. The Unit ID is a unique one-byte address for each host in the system. This address is user configurable, with Unit ID 0 reserved for “broadcast within group” and Unit ID 240 used as the “direct connect address.” The Group ID specifies the group to which a device is assigned. This address has a default value of 2, but is user configurable and

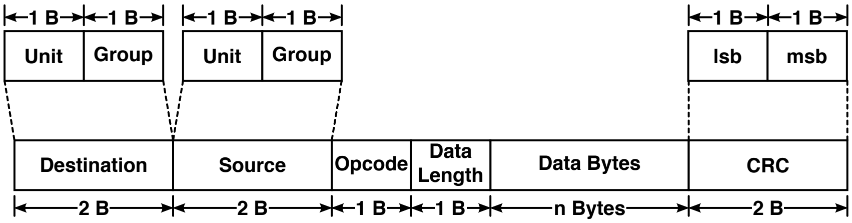

Figure 1. ROC message structure.

can be used to segregate broadcast groups. When a ROC device receives a message, it examines the destination Unit ID and Group ID. The message is accepted and processed if the two destination IDs match the configured device IDs or if the message is a broadcast message with a matching Group ID.

The Opcode field in a ROC message indicates the operation to be performed by the receiving RTU. The operations include configuration modification, retrieval of stored readings and alarms, direct sensor input reading, writing to outputs, acknowledgement of a report by exception, and store-and-forward messaging. An RTU responds to an MTU request with a message containing the original opcode and the results of the operation. When an RTU encounters an error condition (e.g., a request for unavailable data), it responds with Opcode 255 to indicate that the message contained a valid cyclic redundancy check (CRC) but requested invalid parameters.

The Data Length indicates the number of bytes in the Data Bytes field. The Data Bytes field is variable in length and contains the parameters for the operation requested by an MTU or information returned by an RTU. For example, an MTU may use Opcode 167 to request an RTU to measure the Analog Input#6. The RTU responds with Opcode 167 and places the data for the requested point type and parameter number in the Data Bytes field. The ROC protocol specifies the data format and available point types and parameters for each opcode.

The CRC field contains an error detection code to verify message integrity. The standard GPLib CRC routine [4] with the polynomial $x^{16} + x^{15} + x^2 + 1$ is used to compute the 16-bit value. When a device receives a message, it calculates the CRC value and verifies that it matches the value in the CRC field. The message is discarded if the two do not match.

The ROC protocol permits an RTU to generate a "report by exception" message (identified by Opcode 224) when certain conditions occur (e.g., when a sensor value exceeds a predetermined threshold). Upon receiving such a message, the MTU queries the stored alarms and sends a message with Opcode 225 that acknowledges the report by exception message.

3. Scanner Architecture

The SCADA scanner is designed to provide situational awareness for large pipeline systems. Real-time traffic analysis can be very difficult for traditional

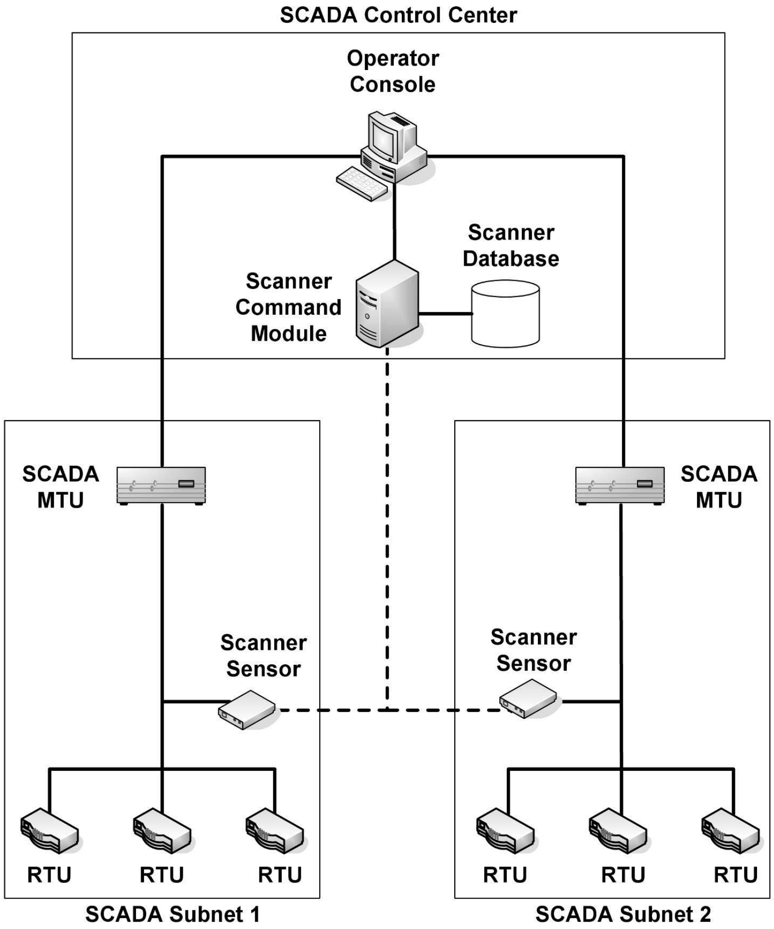

Figure 2. Scanner architecture.

IP networks, mainly because of high traffic volumes, changing network topologies, the range of protocols and applications that are supported, and the relative unpredictability of network traffic patterns and content [10]. On the other hand, even SCADA systems with thousands of devices have low traffic volumes, static topologies, limited protocols and applications, and highly predictable traffic [8]. These attributes make it feasible to implement a real-time scanner that analyzes SCADA network traffic, verifies the operational status of field devices and identifies anomalous activity.

A SCADA system for pipeline operations is typically organized as multiple subnets, each with a master (MTU) and multiple slaves (RTUs). Operators monitor and control pipeline operations from a control center (Figure 2).

The scanner architecture incorporates a command module and database located in the control center and sensors positioned in the various SCADA subnets (Figure 2). The sensors passively monitor traffic and send information to the command module. The command module configures the sensors, interacts with the database and provides event updates to operators. The database organizes and correlates data collected by the scanner sensors. Ideally, the scanner components are dual-homed and use a dedicated communications network so as not to interfere with SCADA operations. However, if bandwidth is not an issue, the scanner sensors may be configured to use the SCADA network for communications.

Modern SCADA systems often employ TCP/IP for device communications, mainly to leverage the flexibility and cost of commodity LAN and WAN technologies. Consequently, the scanner is designed to operate in an IP-based environment. Because the majority of the scanner functionality resides at the application layer, the scanner can be readily modified for use in different communication environments.

This paper focuses on pipeline operations and the ROC protocol. However, the scanner sensors are designed to be modular and to support "plug and play" operations for other protocols. For example, to support Modbus network scanning, it is only necessary to incorporate a Modbus protocol module in the scanner framework. The following sections describe the scanner sensors, command module and database.

3.1 Sensors

Scanner sensors deployed at field sites operate in the promiscuous mode, enabling them to capture and examine traffic in their subnets. The sensors gather information about device functionality, state and network topology; and identify anomalous traffic (e.g., erroneous and malicious messages and unexpected traffic volumes). The sensors are designed to be implemented using inexpensive embedded devices (e.g., Gumstix [5]).

Each sensor maintains a local table with the attributes of devices in its subnet (address information, device functionality and communication patterns). Sensors "learn" about devices and attributes by examining captured traffic. Initially, the local sensor tables are empty; as the sensors parse traffic, new information is gleaned and stored in their tables. Alternatively, a configuration file with device attributes may be uploaded to each sensor by the command module.

Whenever a new entry is added to a sensor table (e.g., a new device address), an alert is sent to the database, which records information pertaining to the alert. The sensors also send periodic status updates to the database to provide situational awareness.

3.2 Command Module

The command module is accessed by plant personnel via an operator console. It provides facilities for managing alerts, reviewing scanner status, configuring sensors and storing historical information related to pipeline control operations in the scanner database.

The command module also serves as the front-end to the database, which it queries constantly for new alerts and changes to SCADA device state and configuration. The query results are passed to the operator console. A human operator processes alerts and examines SCADA device data. Additionally, the operator can view the status of sensors, e.g., subnet data, alerts generated, time of last update and status (active/disabled).

Each sensor has a unique configuration file for scanning its subnet. A sensor may be configured to examine specific device functions and roles, time-out periods and traffic rates. For example, one of the RTUs in SCADA Subnet 1 (Figure 2) produces minimal traffic and an alert should be generated by the sensor when the traffic rate exceeds 20 messages/second. On the other hand, an alert should be generated for the MTU in SCADA Subnet 1 when the traffic rate exceeds 400 messages/second. Note that the command module can upload a new configuration file to a sensor or disable a sensor in real time.

3.3 Database

A relational database maintains historical information about SCADA devices, scanner sensors, network traffic and alerts. Figure 3 shows the eight database tables. The Scanner Traffic table contains the traffic attributes that generate alerts and/or database updates. The Sensors table holds information about the scanner sensors. Four device tables (Devices, Device Types, Device Opcodes and Device Alerts) maintain information about SCADA devices. The remaining two tables (Alerts and Alert Codes) contain information about the alerts generated by the scanner.

4. Scanner Functionality

The ROC scanner provides information about device functionality, device roles, communication patterns, and anomalous process behavior and SCADA network activity. This information about the operational status of the SCADA network and devices provides pipeline operators with vital situational awareness. The ROC scanner generates alerts about anomalous activity by comparing network traffic against normal (profiled) traffic.

4.1 Creating System Profiles

Local tables are maintained by sensors to profile traffic and device operations. Table 1 shows a sample sensor table. When a ROC message is received by a sensor, it examines the source IP address and source device address to see if they exist in its table; if not, an entry is added to indicate that a new device is communicating in the subnet. The sensor also examines the destination IP address and destination device address in a similar manner.

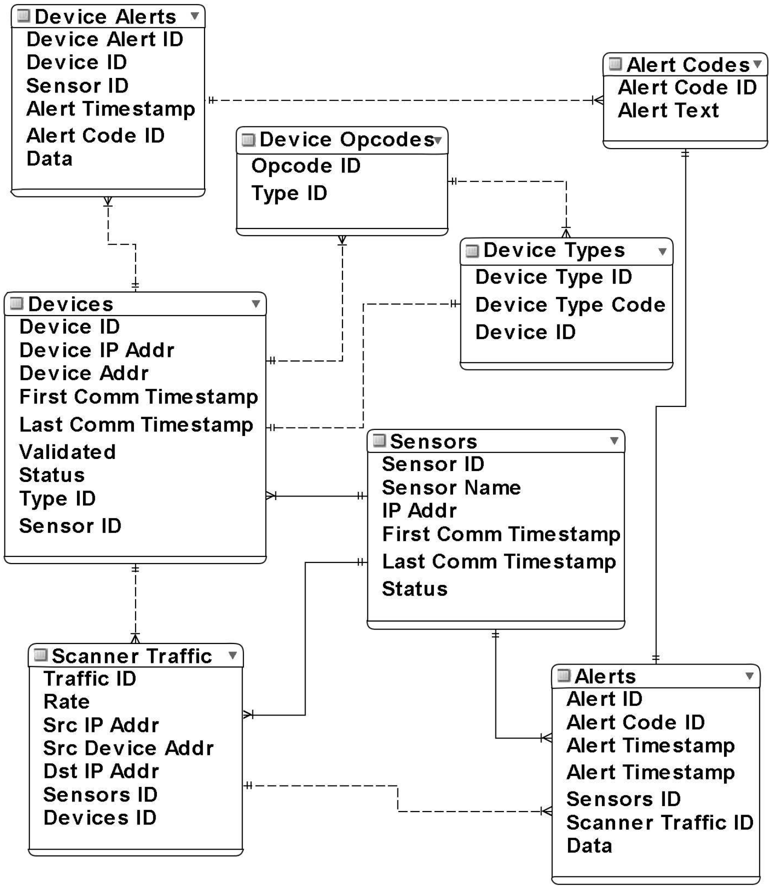

Figure 3. Database relations.

Next, the opcode is examined to determine if the sending device has previously sent a message with the code; the sensor table is updated if the opcode has never been used by the device. The Communication Relations column in the table identifies entities with which devices have communicated. The Opcodes and Communication Relations columns help determine device roles. For example, Device 1 in Table 1 has sent messages to all the devices in the subnet and has used opcodes related to every device in the subnet. Therefore, it can be inferred that Device 1 is an MTU. Devices 2 through 4 use certain opcodes and only communicate with Device 1; thus, these devices are functioning as RTUs.

The Rate column lists the number of messages per second that have been sent or received by a device. This provides an indication of the traffic rate for each device in the subnet. The Last Communication column displays a time stamp and an ordinal date (`ddd`) to identify when the last message originated from the corresponding device.

Table 1. Sensor table data.

Device	**IP Addr.** **Device Addr.**	**Opcodes**	**Comm.** **Assosc.**	**Rate**	**Last** **Comm.**
Dev 1	192.168.10.10 0xAB00	0 2 6 8 11 105 166 171	Dev 2 Dev 3 Dev 4	118 msg/sec	23:14:53 244
Dev 2	192.168.10.20 0x4A05	0 2 6 105	Dev 1	33 msg/sec	23:11:14 244
Dev 3	192.168.10.21 0xFC04	0 2 8 166	Dev 1	64 msg/sec	23:13:29 244
Dev 4	192.168.10.22 0xD607	0 105 171	Dev 1	29 msg/sec	23:14:53 244

The table entries provide a profile of the known state of the subnet. This profile identifies the devices, the functions they implement and their roles, and the communication patterns.

4.2 Generating Alerts

This section discusses the steps involved in processing messages and generating alerts (Table 2). The sensor parses a message to analyze the various fields. If new device and IP addresses are observed, an alert is sent to indicate that a new device is communicating in the subnet. If a new device address and an already existing IP address are observed, a possible spoofed IP address or configuration change alert is generated. Similarly, a possible spoofed device address or configuration change alert is sent when an existing device address and a new IP address are encountered.

SCADA devices are configured to use specific TCP communication ports [6]. Therefore, valid traffic should use the designated ROC communication ports and should conform with the ROC protocol. An alert is sent if a non-standard protocol message is received on a ROC port or a ROC message is received on a non-standard port. Note that alerts are not sent for non-ROC messages received on non-standard ports because these messages are ignored by all ROC devices.

The opcodes of ROC messages that use ROC communication ports are then checked to verify control actions and device functionality. An alert is sent when an opcode is encountered that has not been used previously by a device or that has not been configured as a valid code. Note that numerous alerts are generated when the sensors are first turned on. If this is a problem, the number of alerts generated on start-up can be reduced significantly by using a configuration file that contains information about the baseline state of the system.

Table 2. Generated alerts.

Conditions	**Alert Message**
device address is new==true ip address is new==true	New device
device address is new==true ip address is new==false	Possible spoofed IP address or configuration change
device address is new==false ip address is new==true	Possible spoofed device address or configuration change
roc comm port==false roc msg format==true	ROC message on non-standard port
roc comm port==true roc msg format==false	Non-standard message on ROC port
roc comm port==true roc msg format==true valid control operation==false	Unexpected control operation
exceed rate threshold==true	Traffic rate exceeds threshold
exceed device time-out==true	Device has stopped communicating

Next, traffic rates are computed for the source and destination devices specified in the messages. This is accomplished by maintaining message counts for devices and aggregating the numbers of messages over time. An alert is sent when the traffic rate exceeds the threshold of any of the communicating devices as specified in the configuration file. Finally, a time-out period (in the sensor configuration file) specifies the length of time a SCADA device can go without communication. Each sensor periodically computes the time difference between the current time and last communication time for devices in its local table. An alert is sent when this time difference exceeds the time-out period.

This message processing logic detects several scenarios: (i) an unauthorized system communicating on a subnet; (ii) an attempt to spoof an address or device; (iii) a denial-of-service attack; (iv) a reconnaissance probe performed by an attacker (e.g., port scanning, network mapping, device opcode identification); (v) an attempt to send improper control messages; and (vi) a device performing an unauthorized operation (e.g., a rogue device operating as a master).

5. Experimental Results

This section describes the experimental results obtained for a simulated pipeline control system. The simulation models the control operations of a major gas pipeline company. The experiments, which used real ROC traffic, were designed to evaluate the ability of the scanner to accurately profile system attributes and to identify unauthorized operations (including malicious activity).

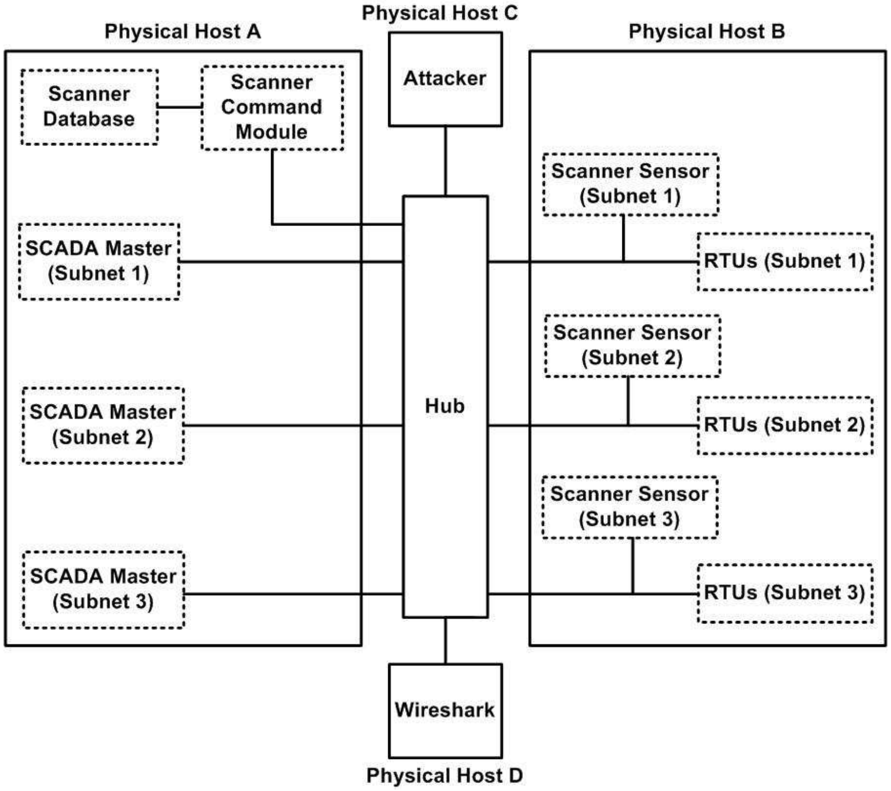

Figure 4. Experimental SCADA testbed.

5.1 Experimental Testbed

Figure 4 illustrates the virtual SCADA testbed used in our experiments. The testbed has three subnets, each with one MTU and ten RTUs; a sensor is positioned in each subnet. The scanner components (control module, database and sensors) and the SCADA system use a common network for communications. SCADA traffic used in the experiments was obtained from an operational pipeline system utilizing Fisher ROC devices.

The boxes with solid lines in Figure 4 denote physical hosts and the dashed boxes indicate virtual machines. Physical Hosts A and B are 2.0 GHZ notebook PCs with 2 GB RAM running Windows XP Service Pack 2. All the virtual machines are VMWare [15] images of SCADA devices and scanner components (three MTUs, three RTUs, three sensors, one command module and one database). The VMWare images for the MTUs, RTUs and scanner sensors use Arch Linux 2.6.22 and are each assigned 256 MB RAM. The database runs MySQL 5.0.51 on Windows XP Service Pack 2 and is assigned 512 MB RAM. The command module also runs on Windows XP Service Pack 2 with 512 MB RAM. The MTUs and RTUs are run on different physical hosts to ensure that

SCADA traffic is visible to the sensors. The IP device addresses are assigned statically as in real-world pipeline operations.

Two additional hosts (Physical Hosts C and D), both 2.0 GHZ notebooks with 2 GB RAM running Windows XP Service Pack 2, are used. Physical Host C sends malicious traffic; it uses the `hping2` utility [13] to craft and inject attacks. Physical Host D uses Wireshark [9] to capture network traffic for validation purposes. All the physical hosts are interconnected using Ethernet network interface cards via a 10/100 Ethernet hub.

The local sensor tables are set to empty at the beginning of each test, requiring the sensors to "learn" the SCADA system attributes. Each sensor is configured for a device traffic threshold of 400 messages/second and a time-out period of 1 minute.

5.2 SCADA Network Profiling

Several tests were conducted to evaluate the ability of the scanner to accurately profile SCADA devices and operations. During each test, the sensors examined twenty minutes of pipeline control traffic between the MTU and RTUs in their subnets.

The MTUs and RTUs were identified almost instantly by the sensors and the corresponding new device alerts were generated. After a new device alert was received and correlated with the correct device, the alert was cleared to ensure that the device did not generate additional new device alerts.

Alerts were also generated for new opcodes used by SCADA devices. Practically all the alerts (and sensor table updates) occurred during the first three minutes of sensor operation. A few alerts related to new opcodes were generated during the remaining seventeen minutes of traffic analysis.

The tests were run four times with different traffic. In every case, the scanner accurately identified the SCADA devices and their functionality. Additionally, the device communication relations and device roles were identified correctly.

Table 3 presents the communication patterns identified during SCADA network profiling. The IP addresses have been altered for reasons of sensitivity; however, the addresses presented are representative of the network topology. The network topology and device roles are easily determined using the address information and communication relations. For brevity, some of the devices identified as RTUs are not included in Table 3. Note that Device 1 communicates with Devices 2–11 on the same subnet while Devices 2–11 only communicate with Device 1. Thus, Device 1 (and Devices 12 and 23) appear to be functioning as MTUs while the other devices are RTUs.

5.3 Malicious Activity Detection

Additional experiments were conducted to evaluate the ability of the scanner to accurately identify malicious activity. Traffic corresponding to four attacks was interspersed with regular network traffic. The attacks involved: (i)

Table 3. System profiling results.

Device	IP Address	Comm. Relations	Role
Dev 1	192.168.10.10	Dev 2–Dev 11	MTU Subnet 1
Dev 2	192.168.10.20	Dev 1	RTU Subnet 1
...	...	...	...
Dev 11	192.168.10.29	Dev 1	RTU Subnet 1
Dev 12	192.168.40.10	Dev 13–Dev 22	MTU Subnet 2
Dev 13	192.168.40.20	Dev 12	RTU Subnet 2
...	...	...	...
Dev 22	192.168.40.29	Dev 12	RTU Subnet 2
Dev 23	192.168.70.10	Dev 24–Dev 33	MTU Subnet 3
Dev 24	192.168.70.20	Dev 23	RTU Subnet 3
...	...	...	...
Dev 33	192.168.70.29	Dev 23	RTU Subnet 3

a spoofed device; (ii) network reconnaissance; (iii) a rogue master; and (iv) denial of service.

Two instances of spoofed devices were executed repeatedly with different device addresses. The first involved MTU messages with existing IP addresses (of RTUs) but new device addresses. The second involved MTU messages with new IP addresses but existing RTU device addresses. In every case, the sensors correctly sent alerts indicating the presence of spoofed devices.

Two reconnaissance probes were conducted on the SCADA network. The first attempted to identify the ROC communication ports. This probe involved sending legitimate ROC messages and incrementing the communication port value until a response was received. In every case, an alert was generated that a valid ROC message was sent on a non-standard communication port.

The second reconnaissance probe emulated an `nmap` scan [7]. ICMP and TCP messages were crafted for host discovery and open port identification. The sensors detected the anomalous messages and raised alerts that non-standard ROC messages were being sent on ROC communication ports. Also, during both the network reconnaissance probes, the machines that generated the messages were correctly identified as new devices.

The rogue master attack involved sending fabricated messages to RTUs. One set of messages requested RTUs to clear their event sequences (Opcode 132). Another requested RTUs to set a new date and time (Opcode 8). The sensors correctly raised alerts about the new master device and anomalous function codes for the associated RTUs.

Table 4. Summary of malicious activity and alerts.

Attack	**Details**	**Alerts Generated**
Spoof 1	Rogue device communicates using an existing IP address	Possible spoofed IP address
Spoof 2	Rogue device communicates using an existing device address	Possible spoofed device address
Recon 1	Port scan uses ROC message to determine ROC communication ports	(i) New device (ii) ROC message on non-standard port
Recon 2	Network scan attempts to discover topology and open ports	(i) New device (ii) Non-standard ROC message
Rogue MTU 1	Rogue device functions as an MTU to clear data in an RTU	(i) New device (ii) Unexpected control operation
Rogue MTU 2	Rogue device functions as an MTU to write data to an RTU	(i) New device (ii) Unexpected control operation
DoS 1	Excessive traffic prevents device from functioning properly	(i) New device (ii) Traffic rate exceeds threshold
DoS 2	Device is taken off-line	Device has stopped communicating

The final test involved two denial-of-service attacks. One attack sent large volumes of traffic to an MTU; the other physically took RTUs offline. In the first instance, alerts were generated when traffic rates exceeded the configured thresholds; also, the attacking machine was identified as a new device. In the second instance, an alert that an RTU had stopped communicating was issued when the communication time exceeded the specified time-out period.

Table 4 summarizes the malicious activity and the corresponding alerts generated during the experiments. The correct alerts were generated in a timely manner in all the tests.

6. Conclusions

Oil and gas pipeline operators do not have adequate means to verify the state and integrity of the thousands of widely dispersed SCADA devices used for pipeline control. A lack of situational awareness about the behavior of SCADA systems can complicate – if not degrade – pipeline control operations. Limited situational awareness also makes it more difficult to detect and respond to the effects of unexpected incidents and malicious acts.

Our distributed scanner provides vital situational awareness about SCADA devices and control operations. The scanner can remotely verify the status and integrity of SCADA devices, profile normal SCADA operations and identify anomalous activity. The current implementation is targeted for ROC, a popular pipeline control protocol; however, the design readily accommodates other SCADA protocols via plug-in modules.

Acknowledgements

This work was partially supported by the Institute for Information Infrastructure Protection (I3P) at Dartmouth College, Hanover, New Hampshire, under Award 2006-CS-001-000001 from the U.S. Department of Homeland Security.

References

[1] ArWest Communications Corporation, Supervisory Control and Data Acquisition (SCADA), San Jose, California (www.arwestcom.com/?s=resources&p=scada), 2008.

[2] Emerson Process Management, ROC Protocol User Manual, Bulletin A4199, Houston, Texas, 2007.

[3] Emerson Process Management, St. Louis, Missouri (www.emersonprocess.com), 2008.

[4] C. Frayn, Genetic Programming Library (GPLib), University of Birmingham, Birmingham, United Kingdom (www.cs.bham.ac.uk/∼cmf/GPLib/index.html), 2006.

[5] Gumstix, Portola Valley, California (www.gumstix.com), 2008.

[6] Information Sciences Institute, RFC793: Transmission Control Protocol, University of Southern California, Marina del Rey, California (www.faqs.org/rfcs/rfc793.html), 1981.

[7] Insecure.org, Nmap Reference Guide, Palo Alto, California (nmap.org/book/man.html), 2005.

[8] T. Kilpatrick, J. Gonzalez, R. Chandia, M. Papa and S. Shenoi, Forensic analysis of SCADA systems and networks, *International Journal of Security and Networks*, vol. 3(2), pp. 95–102, 2008.

[9] U. Lamping, R. Sharpe and E. Warnicke, Wireshark User's Guide: 27121 for Wireshark 1.0.0 (www.wireshark.org/download/docs/user-guide-us.pdf), 2008.

[10] S. Northcutt and J. Novak, *Network Intrusion Detection*, New Riders, Indianapolis, Indiana, 2003.

[11] OPC Foundation, Matrikon OPC Server for Fisher ROC Plus, Scottsdale, Arizona (www.opcfoundation.org/Products/ProductDetails.aspx?CM=1&RI=8538&CU=1), 2008.

[12] ProSoft Technology, Fisher ROC Communications Module (3150-ROC), Bakersfield, California (www.prosoft-technology.com/prosoft/products/for_rockwell_automation/protocol/custom/fisher_roc/3150_roc), 2008.

[13] S. Sanfilippo, `hping2` (www.hping.org), 2006.

[14] R. Shayto, B. Porter, R. Chandia, M. Papa and S. Shenoi, Assessing the integrity of field devices in Modbus networks, in *Critical Infrastructure Protection II*, M. Papa and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 115–128, 2008.

[15] VMWare, VMWare Server Virtual User's Guide (VMware Server 2.0), Palo Alto, California (www.vmware.com/pdf/vmserver2.pdf), 2008.

[16] Wonderware West, Wonderware Universal Server, League City, Texas (www.standard automation.com/products/universal-server), 2008.