CYBERSECURITY
FOR SCADA SYSTEMS

SECOND EDITION

PB
PENNWELL BOOKS

Contents

Chapter 4

Chapter 5

Chapter 6

Chapter 7

Chapter 8

Chapter 9

Figures

be sent to the RTUs and the set of messages that the RTUs could generate together define a *communications protocol*. In the 1960s, vendors of SCADA systems had to design and build their own RTUs and thus also defined their own (proprietary) communications protocol(s). In certain industries today (primarily the electric utilities), there are still RTUs utilizing some of these old, obsolete protocols—often referred to as *legacy* protocols.

In the 1960s, the *universal asynchronous receiver transmitter* (UART) chip had not yet been invented, and microprocessors were just beginning to be invented, so it was up to each vendor to decide the format of their protocol messages (i.e., how many bits in a message). To simplify the electronic design of the RTUs, most vendors elected to send all available numeric (analog) or binary (status) data in single, long, many-bit messages (fig. 1–2). This would mean messages of 74, 96, 123, or some other extended number of bits. In essence, a response to a polling message from the SCADA host was the transmission of the current values of all of the inputs (analog and status/digital), sent as one long message. These early protocols have very few message types and variations (all of which were built into the RTU hardware), and data values were either single bit or an 8 bit binary integer.

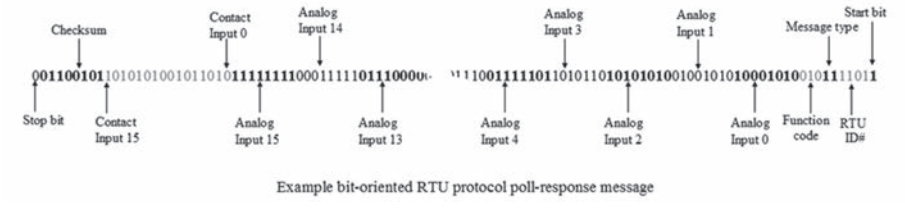

Fig. 1–2. Example bit-oriented message format (starting and ending portions only, owing to actual large number of bits required in a full-length message)

These *bit-oriented* protocols fell out of favor with the invention of the UART chip (and microprocessors), but as previously mentioned, some of these legacy bit-oriented protocols remain in limited use today. These types of protocols usually require the use of specially designed interfaces that can receive and generate the necessary long-bit-sequences. Specially programmed single-board computers are often used for this task. Most 'serial' RTU protocols still used today are based on constructing the messages using some integral number of 8 bit octets/bytes which are suitable for asynchronous serial transmission via UART circuits. Although they normally don't come as a standard interface anymore, computers today can still be equipped with RS-232 serial ports (called 'COM:' ports in the Windows® operating or designated as "ttyS0, ttyS1, etc." in a Linux operating system all of which employ UART circuitry to make them function. Protocols based on messages that use an integral number of octets are generally called *character-oriented* protocols. You will also occasionally hear these two different types of protocols referred to as *synchronous* and *asynchronous* protocols, but this is not technically accurate. In fact,

service provider sets up a system and provides each participating utility a Web-based portal that provides them with displays of their own information (but isolates them from the information of others.) The SCADA vendor provides the equipment to the utilities and assists them with the initial configuration and installation of RTUs and workstations and setting up of their basic displays, alarms, and control screens. Each field site needs to have power and network connectivity, which can be arranged through local telecommunication providers. In some instances, the communications may be via cellular connectivity or possibly even via satellite link. The point is that by sharing the cost of running and maintaining a SCADA system among a fair number of utilities, this can be profitable for the SCADA service provider and cost-effective for the various utilities. This is a form of SaaS (SCADA as a Service).

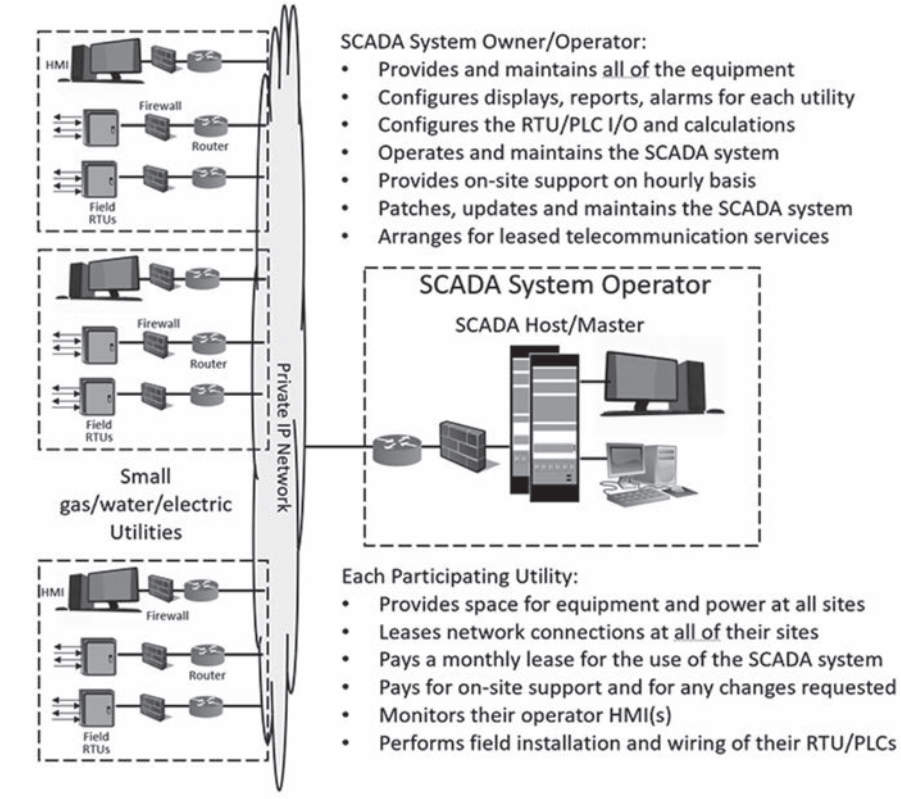

Fig. 1–9. SCADA as a service

These days, there is a lot of noise made about the evolving and emerging Industrial Internet of Things (IIoT) and how it will impact automation technologies (of which SCADA is a principle one). A lot of the IIoT discussion revolves around embedding intelligence into the lowest levels of automation, which for SCADA

Regulatory and sequence control

In the pipeline and water/wastewater industries, the RTUs installed in the field were often located at physical locations around the process where there was a need for some level of local regulatory control and sequence logic. Basic SCADA systems are *supervisory* control systems, meaning that the decisions to take a control action are made at the host level and then dispatched to the RTU to execute. This is fine as long as the communications bandwidth is sufficient and reliable. If gas pressure needs to be controlled at a delivery point and the target pressure needs to be adjustable based on conditions, then control adjustments to the pressure-regulating valve may need to be made every second or faster. With the low-baud-rate serial communication schemes used by most SCADA systems (not to mention channel sharing across multiple RTUs), it is typically not possible to read the gas pressure, send it to the host, make a control adjustment calculation in the host, and send a control output command to the RTU all within a second. In the past, in order to provide local regulation for such applications, it was typical to have some form of instrumentation and control panel at the site, to perform the regulatory control, and the RTU would merely interface with this panel via analog, pulse, and contact I/O points (fig. 2–7), typically to provide a reading of the measurements and to make setpoint adjustments to the instrument performing the *PID control*.

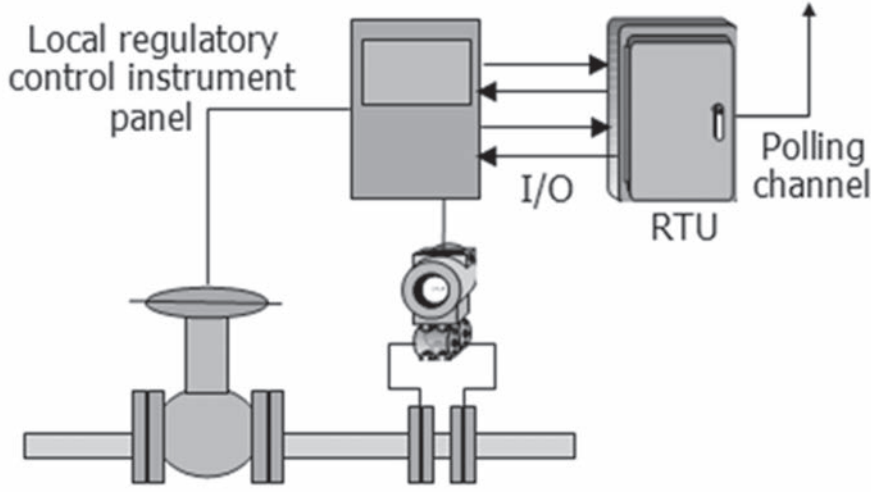

Fig. 2–7. Supervisory control of local regulatory control panels

In the 1980s, microprocessor-based RTUs had become reasonably powerful, and at the same time, a revolution was taking place in the process control industry. Traditional analog instrumentation and control panels were being replaced with computer-based technology. Specifically, the *distributed control system* (DCS) had been introduced and, for reasons both financial and technical, was

local network. (If it is across a LAN, then the IP datagrams themselves are carried as data in the link layer protocol of the LAN [such as an Ethernet frame].)

One feature that these two transport protocols add is the idea of *ports*. Computers on an IP network have unique (IP) addresses (and in fact that can have more than just one), but there can be many different programs concurrently running on a computer and attempting communication with other programs running on other computers. Port numbers (a 16 bit integer value) were created to uniquely identify each communicating program. (Just like a street address for a building can include a specific apartment number so that the letter gets to the right person at the address, an IP address gets more specific with the addition of a port number, so that messages get to the right program.) There are over 65,000 possible port numbers as they are represented by a 16 bit integer (0 to 2^{16}-1). Unlike IP addresses, we have not run out of those numbers. For common applications (services) that are found on most computers, a standardized set of so-called *well-known port numbers* have been assigned (port numbers between 1 to 1023). Returning to the postal service metaphor for the moment, a port number is like specifying the specific person at an address to whom the letter should be delivered. A *well-known port number* is equivalent to using an addressee such as "Accounting Department" or "Human Resources" in place of a specific person's name. Since most companies have such departments, you can get your mail delivered, even if you don't know a specific name of a person in that department. Using a well-known port number is how you get connected to a computer's email server, Web server and most other common services. Unfortunately, because they are well-known, hackers can attack these ports and attempt to exploit identified vulnerabilities in these system-level services.

IP addresses (until IPv6) were just a 32 bit binary number. (That number is usually broken into four octets with each octet written as a decimal number separated by periods: e.g., 128.156.12.33.) The number actually has two parts: a network ID and then a unique computer/host ID within that network. You can think of this like having an area code and phone number. Other people may have the same phone number as you, as long as they have a different area code. The people who initially cooked up the Internet couldn't envision there ever being more than 2^{32}—1 computers connected on a single network worldwide. They were so very wrong. We ran out of unique IP addresses a few years ago, and there have been several temporary workarounds employed to keep things going. In IPv6, the address is a 128 bit binary number (and that should last us a while). When an IP message (actually called a *datagram*) arrives at a computer, the message headers contain the IP address and port number of both the destination computer/program and the sender computer/program. (Actually, it may arrive as multiple pieces called *fragments* if the original datagram was large, and each fragment will have that same delivery information.) This information is like the return address on a letter, the original purpose of which was so that you could send a return message using that address/port information. Today, that sender information might also be used to decide if the message should be accepted or discarded (that is called *packet filtering,*

Utility Type	Description/Use	Usage Frequency
Computations and calculations	Define, modify, and delete calculated values, adjust calculation parameters, assign re-calculation intervals	Frequently during initial system commissioning and then again when system expansions or changes are made. Otherwise not used.
Logging	Assign alarms and events to logging, remove from logging, print logs, purge logs, assign printing and archive directory to logs, define log retention intervals, define operator and other user actions to be logged, define other activities to be logged, remove activities from logging	Frequently during initial system commissioning and then again when system expansions or changes are made. Some functions may be used on a regular basis for system administration purposes
Data exchange	Set up values and status and controls to be exchanged, assign placement and scaling of values in exchange messages, add or remove points, controls, and functions offered to the exchange partner	Frequently during initial inter-system link commissioning and testing and then again when system expansions or changes are made. Otherwise not used.
Redundancy synchronization	Establish the trigger events for fail over to backup, establish schedule for data update of backup computer, enable or disable fail over function, enable or disable data synchronization functions	Used during the initial commissioning and testing of the system. Not normally used thereafter.

The SCADA functions of a SCADA system product normally provide for a wide range of operational configuration flexibility, since a given vendor's SCADA product needs to be able to accommodate a wide range of application and industry variations. Many of the SCADA utilities exist for the purpose of setting up customer-specific and industry/application-specific configuration tables that direct the actions of the generic SCADA software modules (fig. 4–4). For example, the number of polling channels, the number of RTUs per channel, baud rate per channel, and protocol to be used on each channel are all customer-specific configuration parameters that would need to be defined in order for the SCADA system to perform its RTU polling duties. For a SCADA system of any reasonable size there is a massive amount of information that must be defined to make the system work. A single analog input can require a dozen or more parameters to be set, and a SCADA system might have many thousands of analog inputs. Everything from giving inputs names and descriptions to defining how they are to be displayed and even how they are to be historically trended must be defined.

Configuration activities go well beyond merely setting up the polling channels. It is necessary to describe each and every input and output that the SCADA system is to process for every one of the RTUs. The description for each I/O signal (point) includes assigning a *tag name* to the signal, defining what type of signal it is, providing all of the processing and alarm-checking information necessary to manipulate the point, defining the frequency at which the point is to be processed, the actions to

business applications, but these systems are not directly involved in monitoring and controlling the distributed process via the SCADA system. Of course, there are also small SCADA systems that consist of a desktop PC, a printer, and a master radio, all sitting on a desk in the corner. Pipeline and electric transmission utilities tend to have the big, fancy control rooms. Small water utilities and rural electric cooperatives (RECs) tend to have the desktop SCADA systems.

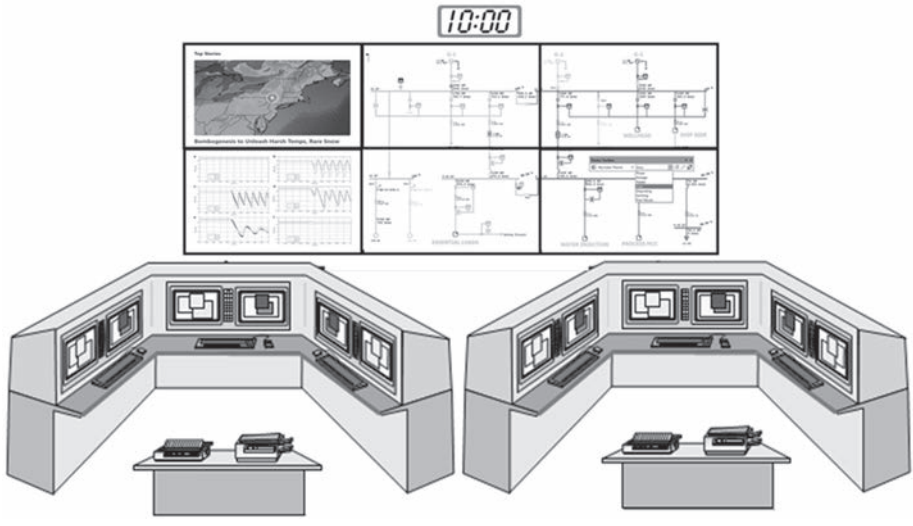

Fig. 5–1. Example SCADA system control room console design

The purpose of all this equipment is to give operational personnel a real-time view of the state of the process they are monitoring and to provide them with a means for initiating control actions and responding to alarms and events, whenever necessary. Obviously, we don't want unauthorized (or untrained) personnel walking into the control room and touching the operator consoles and potentially sending commands out to operate field equipment. So that only authorized personnel have access to the system functions, physical isolation and physical protection of the equipment is common, particularly for the control room (because as we have mentioned, the operator consoles may have no protection other than physical protection because they are always operational and don't require a login or password). As was previously noted, SCADA systems include some type of access-control mechanism for non-operational personnel, whether it is just the user account/password scheme of the underlying operating system or that plus additional SCADA access controls as well. In the vast majority of cases, this is an ID/password scheme, whereby either each user or each category of user is issued a unique ID/password pair that enables (and disables) the functions and features authorized for this user or category of user. The problem with such protective measures is that people don't change or properly protect their passwords, they

7

Identifying cybersecurity vulnerabilities

Threats and Threat Agents

When discussing the possibility of a hacker, a malware infection, or a terrorist attack on a SCADA system, we are really referring to the probability (or likelihood) of an attack occurring and the consequences resulting were it to be successful. An attacker, regardless of type and motivation, will be looking for the weaknesses in your defenses and will attempt to exploit those weaknesses (vulnerabilities) to carry out an attack. It is important to take reasonable actions to protect your systems, particularly if a successful attack upon them could cause loss of life, injury, or substantial damage, not to mention any additional financial impact. But what is the probability of an attack and the possibility that it could succeed, and who are the potential attackers?

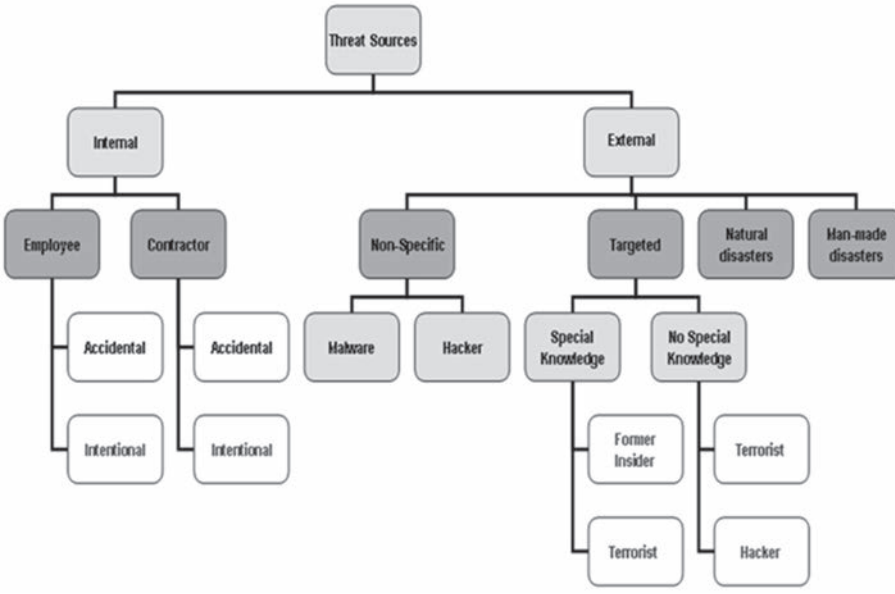

Fig. 7–1. Taxonomy of potential threat sources to a SCADA system