Industrial Control System Security - Getting Better, Getting Worse

2019 NSF Cybersecurity Summit for Large Facilities and Cyberinfrastructure
Phil Salkie, Jenariah Industrial Automation - phil@jenariah.com - 301-859-0500

Reading Period
What's ICS/SCADA, and why should I care?

ICS - Industrial Control Systems

ICS is a catch-all term meaning any automated system which controls physical functions by manipulating mechanical equipment. It can mean the robots which assemble automobiles, or the heating system in a building, or the chillers at a data center. As building systems become more complex and management of plant and equipment gets more automated, more and more ICS equipment is being placed into service in new ways throughout our scientific and technical facilities. This equipment is often not designed with a robust security model in mind, designs are not implemented by engineers skilled in threat analysis, and sophisticated computer systems and communications networks are installed without the input or knowledge of the IT staff.

PLC - Programmable Logic Controller

PLCs are rugged, dedicated-purpose computing systems which use specialized programming languages and industrial-grade interface hardware to directly control physical devices and machinery.

OIT/HMI - Operator Interface Terminal / Human Machine Interface

HMIs are ruggedized display panels with embedded CPUs and multiple communications ports, designed to display data that resides in PLCs and to allow machine operators to make changes to that data. They are generally programmed using separate GUI software from that used for the PLC, and they tend to have a shorter working lifespan than PLCs due to their exposure to human interaction and their use of components like backlit displays and membrane keys. Note that there are vendors producing combination HMI/PLC devices which may present advantages or disadvantages depending on the exact usage situation.

DCS - Distributed Control System

DCS generally refers to systems where the processor is far removed from the points of control, and all input and output signals are transmitted via some form of networking back to the processor. These control systems are most often found in large "Process" control applications - think large tanks in fields, vats in buildings, pumps and sensors, all connected by miles of piping. The programming for these systems is most often a graphical representation of data flow, much like a data flow diagram from the realm of software engineering, or a programming flowchart.

SCADA - Supervisory Control And Data Acquisition

SCADA is a term which is used differently in the computer security field than it is in the controls industry. Security specialists often use "SCADA" to refer to any ICS equipment which isn't a standard commercially-available computer component like a destop PC or a laptop. In the controls industry, it specifically refers to hardware and software of any sort which is not directly involved in controlling the equipment - but rather controls the PLCs which control the equipment. That "Supervisory Control" can mean simple monitoring via web pages, downloading of recipes, collection of log data, or all of the above and more.

Sensors
A sensor is any device which translates real world information into signals that a controller can work with. That can be a simple switch saying that a door is closed, a transmitter which reports the temperature of a liquid or the pressure inside a pipe, or complex equipment measuring the optical clarity of a stream of treated water or reporting on radio-frequency emissions from a system under automated test.

Operators
Could refer to the user of a control system, or a simple operating device like a pushbutton or a selector switch (such things are collectively referred to as “panel operators”)

Actuators
Any hardware which a control system uses to mechanically move something. This could be a motor, an air-operated rod cylinder, or a mechanical gripper, just to name a few.

Telemetry
Remote measuring systems - a telemetry device allows a control system to obtain information from something far away, for example letting a water pumping system know the current level of a distant storage tank.

Custom Hardware
Embedded Controls
Ways to refer to controls hardware which doesn’t use a pre-packaged solution like a PLC or HMI. Often these systems were first prototyped using off-the-shelf controls hardware, then once the requirements were fully determined, a custom hardware and software solution was produced for volume sales. These are the trade-offs involved in producing one-off custom solutions (like the control system for a building or a packaging line) versus large volume solutions like the controls for a dishwasher or a laboratory chiller.

In one-off systems, the engineering time involved in making things work is significant in proportion to the cost of the controls - so it makes sense to spend money on controls hardware which minimizes the time required to build the system. Systems such as these often become difficult to support over time, as component availability, access to specialized accessory hardware and software, and even the closing of the original machine builder may become difficult issues to surmount.

In volume produced systems, engineering time is amortized across the large number of devices that are sold - possibly only equating to pennies per unit. Therefore, investing engineering time up front to make the per-unit cost as low as possible is a definite win over the lifecycle of the product. Systems such as these tend not to allow much site-specific customization, the software which is resident in them is usually a closely-guarded trade secret, and if replacement electronics components are no longer available, it is often necessary to completely replace entire equipment rather than just replacing the controls.

Fun Fact: There were programmable controllers before there were desktop computers. ("PC" meant "Programmable Controller" before it meant "Personal Computer", then was changed to "PLC" for "Programmable Logic Controller" after the IBM-PC came out.)

Not-So-Fun Fact: Some of those controllers are still out there and running.
They may have been programmed on dedicated hand-helds, with their programs stored on cassette tape, or hand-written on paper with pre-printed boxes for entering the data. These paper documents may be the only existing copy of the operating logic.

Fun Fact: There were PLCs before there were communications network standards.

Not-So-Fun Fact: Some of those controllers are still out there, running on dozens of different networking hardware layers and hundreds of software communications protocols.
Names you may run into are: Modbus, CANBus, ProfiBus, FieldBus, InterBUS, CC-Link, DeviceNET, HART, CIP, Ethernet/IP, DF-1, DH-485, MELSECNet, BACnet, LON, ZigBee, SRTP, and many, many more.

Some of these protocols allow an external host to read and write anything stored in the controller's data memory, some allow an external host to read and write the controller's password or the logic program, some allow an external host to flash new firmware into the controller's operating system memory. Some protocols have undocumented functions and features which could be exploited. In 2019, manufacturers have begun the process of supporting fully encrypted communications between devices, with strong authentication and multiple user access levels - currently, this is an expensive optional add-on, unlikely to be used widely until years from now.

Fun Fact: Desktop PC systems are designed with a four year product lifecycle.
One often sees PCs which are ten years old, and occasionally encounters a system that’s over 20 years old. (Win98 SE, anyone?)

Not-So-Fun Fact: Controls systems are designed to be installed in capital equipment hardware which is amortized over twenty or thirty years. It is common to see controllers which were installed thirty or forty years ago, and one can occasionally see operating machinery with controls built before integrated circuits were common.

Fun Fact: The advent of the Desktop PC forced standardization of communications systems, including connectors, hardware layers, and higher level protocols.

Not-So-Fun Fact: Controls systems have used a bewildering variety of connectors, pinouts, standards, and protocols - often making changes to existing designs seemingly solely to break compatibility with existing systems. You may require specialized cables, programming interface boxes, or plug-in cards for your computer - you may even find that modern computing hardware is incapable of communicating with a given system, and you’ll need to find a ten-year-old laptop with a built-in RS-232 port or a PCMCIA card slot.

Fun Fact: Modern PC motherboards support PCIe 2.0 X1, PCIe 2.0 X16, PCIe, and (possibly) PCI busses. They also support USB 2.0, USB 3.0, and (possibly) USB 3.1, USB-C, Lightning, and Firewire.

Not-So-Fun Fact: Control systems have used S-100, VME, STDbus (8,16, and 32), ISA-8, ISA-16, PC/104, PC/104-16, PCI, PCIe, and a host of other proprietary busses either based on existing standards of the time or developed entirely in isolation.
Programmable Controllers generally do not use standardized busses, but have proprietary data paths and connector sets. Dozens of bus architectures are currently in use in controls systems worldwide today.
If Desktop PCs are part of a control project, it is more than possible that they are using a bus architecture which is not available on new motherboards, and are using plug-in cards which cannot be purchased new (and may even be difficult to get repaired.)

Fun Fact: Modern computer systems use flash memory for long-term storage of data which is read more than it is written. (BIOS, etc.)

Not-So-Fun Fact: Programmable controllers may store firmware in mask-programmed ROM, UV-EPROM, or EEPROM/Flash Memory. They may store user programs and data in battery-backed SRAM, capacitor-backed SRAM, EEPROM/Flash Memory, UV-EPROM, or some combination of these. If vulnerabilities are known, patching may not be possible.

Many control systems have some sort of replaceable lithium battery to hold up SRAM data storage, or run a real time clock. While the shelf life of these batteries is 10+ years, when they are actually being used to support the memory (i.e. when sitting on a shelf as a spare part) they can deplete in less than a year. Also, batteries aren't just for CPUs - often there are special-purpose communications or motion control modules which have their own batteries which need to be tracked and changed.

The retention times of flash memory and EEPROM varies widely based on temperature and the amount of time since it was last powered - and, since the time frames we are dealing with are long in comparison to the existence of the components, largely theoretical at this juncture.
As some of these parts have been sitting for upwards of a decade, we have started seeing failures due to flash memory "bit rot".

Fun Fact: Your facility almost certainly has multiple industrial control systems which are running building infrastructure like chillers, water pumps, and emergency power generation.

Not-So-Fun Fact: There's probably no central inventory of those systems, no backup and recovery plan, no spare parts, no preventative maintenance, nobody who claims responsibility for any of those areas, and no budget to make any of that happen.

DHS ICS-CERT Training, available for free in Idaho Falls, ID
(I attended, and found it interesting - the red/blue exercise is some serious business.)

https://ics-cert.us-cert.gov/Training-Available-Through-ICS-CERT#workshop
Hands-On Format - Technical Level

ICS Cybersecurity (301) - 5 days

This event will provide hands-on training in discovering who and what is on the network, identifying vulnerabilities, learning how those vulnerabilities may be exploited, and learning defensive and mitigation strategies for control system networks. The week includes a Red Team / Blue Team exercise that takes place within an actual control systems environment. The training provides the opportunity to network and collaborate with other colleagues involved in operating and protecting control system networks.

Note that this course is not a deep dive into training on specific tools, control system protocols, control system vulnerability details or exploits against control system devices.
This event consists of industrial control systems cybersecurity training and a Red Team / Blue Team exercise:

- Day 1 - Welcome, overview of the DHS Control Systems Security Program, a brief review of cybersecurity for Industrial Control Systems, a demonstration showing how a control system can be attacked from the internet, and hands-on classroom training on Network Discovery techniques and practices.

- Day 2 - Hands-On classroom training on Network Discovery, using Metasploit, and separating into Red and Blue Teams.

- Day 3 - Hands-On classroom training on Network Exploitation, Network Defense techniques and practices, and Red and Blue Team strategy meetings.

- Day 4 - 8-hour exercise where participants are either attacking (Red Team) or defending (Blue Team). The Blue Team is tasked with providing the cyber defense for a corporate environment, and with maintaining operations to a batch mixing plant, and an electrical distribution SCADA system.

- Day 5 - Red Team/Blue Team exercise lessons learned and round-table discussion.

- Prerequisites: Each attendee should have an understanding of ICS networks and IT network details. **Every student attending this course should bring a laptop computer (with a DVD drive).** The user must be able to boot the laptop to an operating system from the DVD. If using a DVD is not an option the user may run the operating system in a VM such as VMware Player, VMware Fusion or Oracle VirtualBox.

- This course is presented at a facility in Idaho Falls, Idaho, USA configured specifically for the aspects of the course. A Certificate of Completion will be provided at the conclusion of the course. Refer to the ICS-CERT calendar for a schedule of this training option.

Industrial Control System Security - Getting Better, Getting Worse

2019 NSF Cybersecurity Summit for Large Facilities and Cyberinfrastructure
Phil Salkie, Jenariah Industrial Automation - phil@jenariah.com - 301-859-0500

1) Introduction to ICS/SCADA - A brief history of ICS

Originally it was all relays - controls systems didn't use tubes for reliability reasons - then they started migrating to solid state logic and have been migrating ever since. PLCs were designed to be a hardware and software simulation of mechanical relay designs. The "programming language" presented to the user was a graphical representation of an electrical wiring diagram. (And often still is.) PLCs were based on 8-bit or 16-bit microcontrollers, and had their user-facing configuration based around the limitations of the underlying chips. Some newer designs maintained instruction and data level compatibility with older units, some switched to newer processors and hardware and incorporated hosts of new features in their software design, breaking direct compatibility with older devices and requiring an engineering effort to port older designs.

PLC networking was designed to be deterministic and reliable. Time delays due to data transmission were known and could be designed around in situations where those delays affected the mechanical operation of equipment. Most networks were closed protocols, requiring special hardware to transmit and receive data, which made interfacing with competitors' equipment difficult or not cost effective. Some networks (Modbus, Profibus, CANbus) were developed on more open platforms, and while some still required special hardware, the specifications were available and designs could be licensed and validated. Modern systems often combine these dedicated networks with various forms of Ethernet networking to make a hybrid communications system.

HMIs were originally pushbuttons, switches, gauges, and lights. Later on, small displays with membrane keypads were used to provide a simple device allowing the setting of time delay values, temperature setpoints, or cycle counter presets. Those gave way to small graphic screens with touch capability or membrane keys, then to color graphics screens of various sizes. The most advanced could be a 21" touchscreen showing many pages of operational data, menus, recipe selection, alarm and warning displays, and system event logs. Some newer HMIs can move data between systems which normally could not communicate at all, and thus can be used as data firewalls and protocol translators in addition to their normal purpose as an operator's interface to the PLC.

SCADA systems started out as minicomputer installations which could gather and store data from PLCs, transfer production quotas to them, and log alarms and events for external analysis. As microcomputers became available and increased in power, the capabilities of SCADA systems have increased alongside, to the point where the line between "Control" and "Supervisory Control" can get blurry. Add to this the profusion of control and HMI software systems which run on commodity X86 PC hardware, and it's no wonder that terms have developed different meanings in different subspecialties. One potential miscommunication
to avoid is that a specification describing the security requiredfor a SCADA system - a controls contractor may not automatically assume that those specs refer also to the system's PLC and HMI hardware and software.

What's so special about this hardware?

Slow, Reliable Processors, Ruggedized Hardware
ICS hardware is designed to work on factory floors, and to have a product life-cycle which matches the life span of the equipment which it controls.

Extended temperature ranges, conformal coatings, electrical isolation and noise immunity.

Direct connectability to many physical devices (sensors, buttons, lights, solenoids).

Compatibility with voltage levels found in automation equipment - 5VDC, 12VAC, 12VDC, 24VAC, 24VDC, 110VAC, 220VAC, and more are used for on/off ("Digital") signals.

0-10VDC, 4-20mA, thermocouple level and RTD level signals among others are used for continuous("Analog") data signals.

Background OS layer Creates A Virtual Machine
Single-Threaded - Deterministic Code Execution
ICS was using VMs before they were cool.

PLCs present a programming language and a virtual CPU to the user, while the actual OS handles all the background tasks of I/O, timing, counting, and communications. The virtual CPU is generally handled as one task in an ordered cyclic scan, so the time required to execute a given virtual instruction is known and does not vary with system load. In a simple PLC system, the physical CPU might first handle network tasks, then runs all functions related to timing and counting, performs a physical input read, runs the virtual CPU's program, sends out the physical outputs, and repeats those cycles in an endless loop.

Deterministic Networking
To complement the known time requirements of the instruction set, many PLC networks were designed to have fixed execution times, so that when a signal was triggered at one end of the network, the minimum and maximum possible times until the signal was processed at the other end were well documented and could be accounted for in the user's program. Stochastic (random) networks like Ethernet present special challenges when applied to industrial automation, and often the dedicated networks are preferred for installations where reliability and timing are critical.

Harvard Architecture - data cannot be executed as code
The virtual CPU the user sees has a data space, but the code that is run by that CPU is not anywhere in that space. The virtual CPU cannot modify the code it will run, and (barring unknown bugs in the system's firmware) nothing the virtual CPU can do will cause the actual CPU to crash or execute instructions incorrectly. (A user program _can_ cause the virtual processor to stop or halt on error, by exceeding a watchdog timer, accessing non-existent hardware modules, or dividing by zero.) This removes a whole class of security vulnerabilities such as viruses, trojans, and the like from consideration, but can confer a false sense of the system's invulnerability to engineers.

Specialized Programming Languages: List, Ladder, Function Block, Structured Text
Originally PLCs were programmed using a sort of simple boolean logic assembler language which was designed to be able to be hand-generated from a hand-drawn version of a simplified electrical relay logic drawing, sometimes referred to as a "ladder" diagram because it had a vertical power rail on the left, a vertical ground rail on the right, and logic interconnections in between, like rungs. Later equipment allowed input of the ladder diagrams directly into a special-purpose programmer, then PC software allowed ladders to be entered as text lines, then as graphical depictions of the original "ladder" drawing. It's likely, at this point, that a large percentage of those who program PLCs have never drawn an electrical circuit or wired a relay control system - but still program the controllers using a language that's derived from electrical wiring. Realizing this, there's been a push (primarily in Europe) to standardize on languages which either use a building-block format with interconnecting lines (Function Block Programming) or a BASIC-like text language to program controllers. Many PLCs now allow programming in more than one of these languages, with varying levels of interchangeability between them. Each has its advantages and disadvantages, such as speed of program entry, clarity/readability or lack thereof, and familiarity to the available pool of programmers.

Instructions to Emulate Hardware - Timers, Counters, Cam Switches, PID Loops
PLCs have built-in instructions to directly handle complex functions such as timing (wait 0.07 seconds, then turn on the output, and keep it on until I tell you to stop), item tracking (follow these boxes down the belt, remember which are empty), rotational triggering (turn on this light between 0 and 40 degrees), and temperature control (hold the temperature at 27 Degrees C, compensating for external heat fluctuations, sudden changes of temperature, and variable heating or cooling power from the equipment). Many of these things seem simple, but are fiendishly difficult to implement well on a computer which is simultaneously processing many computational threads in parallel.

Hot Swappable Components
Larger PLCs often have some ability to swap out failed components without stopping the control system. Some processes are not easy to start and stop at will (think steel mills or chemical processes) and being able to hot-swap failed components is a huge benefit.

Known state of hardware at start, stop, and failure
PLCs are guaranteed to have all their outputs in a known state (usually "Off") at power up, when stopped, and if there is a failure of the CPU. This is a critical and often overlooked difference between PLCs and control systems run by personal computer, Raspberry Pi, or even dedicated microcontroller hardware.

Ever wonder how all the fireworks got launched right at the start of the display, rather than waiting their turn and launching one by one like they were supposed to? Likely because the homebrew control scheme someone put together turned every output on for a few milliseconds at power-up, and nobody noticed that because they didn't actually run the tests with actual fireworks - just a bunch of LEDs. But the milliseconds of "On" at boot were enough to light **all** the fuses, and then there was no turning back...

What's been happening in the PLC/SCADA space in the last year?

Urgent/11 - don't color me surprised...

For many years, security analysts have been concerned that the combination of older software components, hardware vendors' opaque development processes, and the widespread use of numerous closed-source third-party software libraries were a serious problem just waiting for a time to occur.

Earlier this year, a group of 11 critical security flaws in the TCP/IP stack of a very widely used embedded real-time operating system (RTOS) called VX/Works (and possibly other predecessor systems which used the same TCP/IP stack) were disclosed. While the vulnerabilities are high severity, the real issue is less related to the exploits themselves, and more to the realities of the ICS landscape.

We know some vendors who have built systems based on the affected products, but certainly don't have a comprehensive list of impacted manufacturers or devices, and likely never will.

While firmware patches have been issued for some systems manufactured by certain vendors, many others have been classified as "will not fix." Almost certainly, orders of magnitude more systems are just abandoned by their manufacturers, or the issue is being ignored by the vendors.

Patching of ICS systems is sometimes easy, but more often logistically nearly impossible. Systems have been tested to perform a certain function, and installing a patch may introduce regressions which cause improper behavior. End-users may have validation protocols in place which would require a complete re-test of a patched system, causing downtime and lost productivity - even with a demonstrable threat, a customer may prefer to attempt external mitigations rather than modify a running ICS system. Adding to these problems is the fact that many systems are ROM-based, and patching would require soldering - so hardware needs to be replaced, and that replacement hardware may not even exist.

On the positive side, many of the systems which may be vulnerable are running software that's in flash memory or in mask-programmed ROM, so they're not going to be able to have a persistent modification made to its operating firmware. Another odd benefit here is that since one primary vector for Urgent/11 attacks to penetrate firewalls is via a compromised DNS server or DNS MITM attach, the fact that a majority of ICS systems don't ever use DNS (since they operate entirely within limited networks and don't access the internet for updates or communications) acts as a safeguard against compromise.

That said, if an attacker can remotely cause vulnerable equipment to lose TCP/IP connectivity or even have a full system failure requiring reboot, there's a potential for lots of very expensive damage to be done.

MQTT - The Industrially Designated Internet Of Things

Just as we see in the consumer and office landscape, there is a push towards a profusion of small, low-power devices which communicate over Ethernet, WiFi, or a variety of dedicated radio protocols. While it is undeniable that there are times and places that wireless sensors are the absolute best method for retrieving a particular data item, the security questions they raise are too large to ignore.

A very common small-sensor protocol is MQTT (The "MQ Telemetry Transport"), first published in 1999. Over the years, it has been expanded to include user authentication, and it can be encapsulated in a secure transport layer such as SSL - but many small devices which use MQTT don't have the processor power required to perform encryption in a timely manner. Authentication's nice, but if you can eavesdrop on an unencrypted channel, it's mostly window-dressing.

Binary protocols such as MQTT are vulnerable to implementation errors - length fields, difficulties in properly handling of UTF-8 strings (a maximum length of 64K in MQTT, larger than the RAM available to many implementations), and the need for correct bit handling all leave doors open to errors which would not occur in a text-based protocol. Partial implementations of the specification also lead to potential mishandlings of data, and although there is now an authentication mechanism, it is not often used - more ways for information to leak, or improper values to be injected.

While all these potential issues exist in various consumer devices, if a drive-by attacker can read a home's WiFi password through a mis-configured network of light bulbs there's a limit to the amount of damage that leak will allow. (I admit it may be very, very bad for a given home-owner, possibly even their neighbors.) If, however, a poor implementation of wireless sensor data allows an attacker a foothold onto a corporation's internal WiFi network, that may be the toehold that allows exfiltration of corporate data or the injection of serious desktop malware. The ability to overpower a wireless signal with a false, externally generated value could be used to cause all manner of physical havoc - think overfilling liquid storage tanks, causing generators to run out of fuel, or disrupting highway signals.

In prior years, many signals of these sorts were sent over dedicated twisted pair wiring, using protocols which, while insecure, were protected by the need for physical access to the wires. Now, as more and more devices are being placed on corporate networks and even on the open Internet, these signals are much, much more vulnerable to malicious modification than their RS-422 point-to-point hardwired predecessors.

ICS Firewalls - Vendors are Stepping Up To The Plate

As processor power, memory, and storage has increased on desktop and server computing systems, it has similarly increased in firewall devices. What was previously only available to the largest of the large systems is now packaged in a Rack Unit or two - this is allowing commercially available firewalls to perform protocol-aware deep packet insepction for protection of even the smallest business networks.

ICS is benefiting from these advances now - vendors are producing firewall devices specifically aimed at the ICS/SCADA market segment, with built-in parse tables for multiple industrial protocols and the ability to define rule sets which allow, for example, read-only access to certain registers from a given IP address range, while blocking all write attempts. This allows an IT department to install robust monitoring and protection of ICS data channels without requiring any modification of existing ICS system programming, a definite win for the year 2019 in ICS.

Even better - concepts which were pipe dreams in our industry ten years ago are now a relatively inexpensive reality. Several brands of ICS protocol aware firewalls are equipped with “learning” modes, where you can place a firewall device in the middle of a data channel, then instruct it to observe, classify, and record the traffic and identify all the specific protocol transmissions that have occurred. During this time, you endeavor to exercise the ICS data channel as fully as possible, calling up all the screens on the HMI or SCADA displays, changing the changeable values and putting them back, etc. Once that’s done, you tell the firewall appliance to alert on any messages which don’t fit the detected patterns, or block those transmissions - no understanding of firewall rules required. The glimmer of hope here is that we will start to see controls vendors offering and installing these devices as part of ICS cabinets, because they don’t require a deep understanding of packet filtering - they’re more “plug and play”.

The Cloud - Some Good, Some Bad

Everyone’s seen situations where migrating a service to the cloud has been a huge win - using a cloud provider for email and dropping the amount of spam by 90%, or using cloud storage to share files among distributed user bases. We’ve also seen cloud services cause havoc - accounts deleted, vendors disappear, services get obsoleted. It’s made most of us interested, but cautious, aware of the good and the bad, and looking for the right balance.

More and more ICS services are becoming cloud-available, as well. Remote monitoring of processes with a smartphone app, entire cloud-based SCADA systems, cloud-based data archiving, and cloud-facilitated remote access systems are all gaining traction in the industry. The obvious problems each of these present are hard to overstate, but the benefits each can bring have the potential to be vast - so it’s up to us to help our clients navigate these waters evaluate the vendors and services, and decide when something should go cloud, when something should be “cloudy”, but hosted internally, and when things are best left entirely local. The decision tree here uses the same questions that cloud services for general business would invoke:
“What will we gain by using the system?”
“What do we risk losing by using the system?”
“What options do we have if the service is temporarily unavailable?”
“How do we ensure we can continue operating if the service disappears without warning?”

One item I'm actually pleased to see go cloud is remote access to PLC/HMI systems. For a decade or so, there's been a push to move ICS equipment from stand-alone, air-gapped networks onto corporate networks. There are advantages and disadvantages to this - but one of the biggest disadvantages is that if remote access to a control system is to be provided on a corporate network, that means whoever has remote access has some sort of presence on the company's business net.

In 2019, after some high-profile intrusions and incidents have made the news, we've started to see customers moving away from ICS network integration back to the idea of physically separated ICS networks, with cloud-based systems for remote data access. These are basically NAT routers with a cloud centered registration and authentication system - a mix of NAT, an access-restricted equivalent to Dynamic DNS, and a VPN system on the user's side. Think "TeamViewer" for controls networks, and you won't be far off base. While there's always the potential for improper access to a controls system through one of these devices, they're physically more readily controlled than VPN access through a business firewall. Power to the device can be controlled through the PLC system itself, allowing access to the ICS network only when the system's engineers authorize it, and only for a limited time. It's also much easier to physically disconnect the one external network cable if you want to disable remote access, or just manually power the access router off when you're not allowing access.

As always, just saying "We're going to air-gap the networks" is no panacea - there are plenty of ways for that to go wrong - but what it is is another layer of "defense in depth", and what's important is that the ICS networks not be a "shadow IT department", but be a known entity, able to be monitored, but not made into an easy, hidden entry point to the business network.

Some Standards Emerge

While there are still hundreds of ICS communications protocols, there have been a few which have become de facto (or actual) standards:

Modbus was one of the first serial protocols which was openly published by its manufacturer and permitted for royalty-free use. That made it a sort of least-common-denominator of protocols, not the best for any given purpose, but that which pretty much every controls device could speak some subset of.

MQTT, mentioned earlier, is a published standard allowing lightweight messaging between sensors and controls systems. More and more controllers and displays have built-in MQTT messaging stacks, and we'll see it put into use in good ways and bad ways in the coming years.

CIP, the Common Industrial Protocol, is a standardized version of Rockwell's "Ethernet/IP" communications protocol. This is a complex intercommunications system which allows transfers by tag name (Send me the value of "Main Room Temperature"), which adds a level of self-documenting to the network traffic. While CIP has had encryption as part of the standard for several years, only this year has hardware become available to permit even limited use. As more vendors provide encryption as a possibility, it will become something which can be added to our controls system specifications, and maybe cause us to breathe a bit easier. (Until that point, the increased use of CIP is causing traffic which would have been on a more secure twisted pair to be put onto Ethernet networks - which just makes me nervous.)

Notes On ICS Protocols

Protocols generally have no encryption, and generally have minimal or no authentication means.

ICS/SCADA comms is generally VERY predictable, has a limited number of PRE-DEFINED IP addresses. Makes alerting easier - IF you even know the ICS network is there, and monitor it.

Air-gapping and network segmentation is becoming more popular after a decade of "Integrate Everything On One Big Happy LAN". As more intrusions occur, IT is becoming more aware of possible poorly-guarded entry points and unpatchable ICS equipment's vulnerabilities. The operational benefit of ICS Network integration with Business LAN is being overshadowed by the large risks to the ICS equipment and the risks to the network in case ICS equipment can serve as a point of access.

For more generalized needs like wide-area access to production data, consider interposing a stand-alone device to serve web pages, push data tables to a cloud provider, and/or send emails from the ICS equipment to a central collection point.

As ICS networks are once again being divorced from business networks, how do we permit external access to the ICS network? VPN devices from numerous vendors (Ewon, Rockwell, Secomea, KEB, etc.) allow cloud-based connection to remote devices - like a hardware box that's got TeamViewer running on it. This allows ICS networks to have an easily controlled, encrypted, authenticated tunneling system which lets external users access the network as needed, while helping to minimize the potential for intruders to gain entry to the ICS network or monitor/affect any data in transit.

Some Security Hardware Vendors

ICS Protocol-Aware Firewalls

Tofino / Belden - product line includes deep packet inspection with built-in awareness of ICS protocols, some units have “learning” mode to observe network traffic, then automatically build filter rules based on the traffic which has been scanned.

Nozomi / Fortigate - less advanced ICS capabilities than the Tofino units, but integrated into Fortigate’s full-line business firewall / sandbox / AV / AP security hardware/software platform.

Moxa - protocol conversion and firewalling systems, allow pickup of data in one protocol and forwarding of data in another, allows limiting of accessed data.

Red Lion - HMI displays, protocol conversion systems. Not specifically sold as firewalls, but the availability of multiple protocols, filtering, and forwarding makes a way to add security enhancements while performing other functions.

Cisco - they’re noticing that ICS protocols are a thing, and are adding many pre-defined filters to their protocol-aware deep-packet-inspection firewalls.

Remote Access Device Vendors

Ewon (Many cloud devices for different requirements, including 4G cellular access points)

Rockwell (Tight integration with Allen-Bradley controller systems)

Secomea (Both cloud and user-hosted systems, European vendor)

KEB (An offshoot product of their wind-power industry controls)

Numerous different versions of the NAT-VPN based remote access hardware, different use cases may point to different vendors, but they’re all pretty much in the same space.

ICS Protocols of LESS concern for intrusion/tampering
This is based on the transport layer being local, twisted pair, coaxial, etc.
(These are the more common ones, by no means an exhaustive list.)

Modicon Modbus RTU / ASCII (Very widely supported inter-vendor comms method)

Phoenix Profibus (Mostly for remote I/O and sensors talking back to PLCs)

Foundation Fieldbus H1 (Again, mostly remote I/O)

Rockwell DF1, DH-485+ (PLC to PLC, HMI to PLC, multi-drop twisted pair.)

Mitsubishi CC-Link (High speed twisted-pair, more popular in Asia, many 3rd party devices)

Siemens MPI / PPI (Multi-Point, Point-To-Point - PLC to PLC, HMI to PLC, twisted pair)

ControlNet (Rockwell's high-speed sensor network)

BACnet/MSTP (Building Automation - Remote I/O and data gathering from smart devices)

HART (Data transmission over 4-20mA analog control loops - sensor programming)

CAN (Automotive data networking)

AS-i (High speed sensor network from Siemens)

ICS Protocols of MORE concern for intrusion/tampering
Routable Ethernet, Wireless

Modbus TCP (Same as Modbus RTU, but over TCP. No authentication or encryption)

Ethernet/IP, CIP (Supports authentication and encryption, but that's almost never used)

BACnet/IP (Building automation protocol over Ethernet, no authentication or encryption)

Fieldbus HSE (High Speed Ethernet version of Fieldbus, no authentication or encryption)

MQTT (Small sensor telemetry, authentication but no encryption, can be Ethernet or wireless)

Zigbee (Wireless sensor telemetry, no authentication or encryption)

ICS/SCADA Network-Based Attack Types
(In rough order of likelihood)

Denial of Service - can an attacker make the communications path, and thus whatever is depending on that pathway, unusable?

Exfiltration of Data - can an attacker obtain sensitive data from a system?

Improper Control - can an attacker modify system data without authorization?

Denial of Service, Next Level Down - can an attacker, addressing only the communications path, cause the _Control System_ to spend so much time handling comms that it can no longer effectively operate the physical hardware it is meant to control?

Lockup - can an attacker turn off or crash the controller, HMI display, or SCADA system, requiring a restart to return it to service?

Program Damage - can an attacker erase or corrupt a device's control program so that the device cannot be restarted without engineer intervention?

Data Modification - can an attacker interpose themselves in a data stream and modify values being sent between two devices?

Program Modification - can an attacker modify the control program to do something damaging or nefarious? (think: Stuxnet)

Firmware Damage - can an attacker erase or corrupt a device's operating system firmware to the point that it cannot be restarted without engineer intervention or component swaps?

Firmware Modification - can an attacker change a device's firmware with modified versions which will appear to operate correctly, but will have hidden functions such as data exfiltration or attacks on other devices? (Think: Cisco router firmware mods by state actors)

ICS/SCADA Information Security Failure Modes - Not Network-Based
(In no particular order)

Loss of PLC or HMI program at power outage due to failed lithium battery.
Inaccessible PLC/HMI program backups due to media failure.
Requirement for unavailable obsolete hardware to backup or restore PLC/HMI software.
Not realizing that particular ICS components are also programmable, not just the CPU module.
Not realizing that some devices store configuration values external to the program memory, and that those values may not be included in any backups that were made.
Lost or never provided system passwords.
Lost or never provided PLC/HMI programs, maybe along with original vendor company closure.
Hardware components no longer available or able to be repaired.

ICS - The Threat Landscape

Security is more than "Cyber"
Security is an awfully broad term. For $5, you can buy a TSA approved luggage lock, which anyone can find a key for with a single Google search. Or you can spend upwards of five grand, buy a bullet-resistant ultra-high security door lock with a registered pick and bump resistant RFID key. Or spend any amount in-between.

Part of our task is to determine which systems get what level of security, and in what form or forms. Controls system security means anything from:

Preventing unauthorized access ("**Physical**")

Improving power conditioning and reliability and/or adding surge and lightning protection ("**Electrical**")

Fixing issues which cause the equipment to be misused or improperly controlled by the users ("**Operational**")

Ensuring that software is backed up, spare parts are available, and we have (or can obtain) everything necessary to repair a failed system in a timely manner. ("**Maintenance**")

Collecting observational and diagnostics information on the controls system and providing for reporting of deviation ("**Monitoring**")

Securing the digital networks which are used in the operation and supervision of these controllers ("**Data** or **Cyber**")

Security on multiple levels is part of a program of **Defense In Depth**, the various things we do at each stage can help prevent an incident or they may slow down an intruder or delay an inadvertent action enough so they are detected and blocked before damage is done.

Cyber may be last on that list, but it is important nevertheless
If we look at the things which can cause an industrial control system to fail, and the degree of difficulty required to access and cause trouble to the majority of ICS devices over an ethernet network, the other types of security would seem to be much more likely points of failure. However, as networking and data communications becomes more and more a part of the ICS and SCADA landscape, we're prone to increasing the number of data pathways which are possible points of intrusion. While a desktop PC or server often has just a single ethernet port as its primary attack surface, a control system can consist of dozens of devices on multiple types of data network, each with different physical layers, data characteristics, and security concerns. Systems which control scientific experiments, data center cooling and emergency power, or building automation make prime targets for skilled intruders - especially as their

security situation can be easily overlooked, they can be accessed in unexpected ways, and their equipment continues to appear on “Click and Crack” frameworks like Metasploit.
An important point to remember is that in many cases, PLC systems have had Ethernet ports tacked onto them more as a marketing necessity than due to actual engineering requirements. It’s 2018, so everything has to have an Ethernet port - whether or not it really serves a function. As we’re all no doubt aware, network security is a complex business, and the various protocols which have evolved over the years are the result of years of “learning the hard way” and “standing on the shoulders of giants”. When a PLC manufacturer implements an Ethernet port, they’re not likely to graft in a processor running Net BSD - they’re going to implement something they wrote themselves with code cribbed from someplace on the net, and they’ll get it working well enough to satisfy marketing, and then they will go back to what they were working on before. This leads to some very odd implementations, like ethernet ports which do nothing more than encapsulate the data from a serial port into a telnet session, or ethernet cards which do packet inspection to determine what protocol is being sent, regardless of what port it arrived on. There’s “security” which is implemented by having the client request the PLC to send its password in clear text for verification, or PLCs which will just allow a client to ask “Is 0000 the password?” “How about 0001?” over and over, so that brute-forcing takes just hours.

There will be systems which can be easily secured, systems which can be updated, or firewalled, or isolated - and there may be systems which you can't touch at all, and the only thing that's possible is to set up monitoring and alarms on their network traffic. There are some systems which, to be honest, aren't worth the time and effort it would take to secure them. You’ll be able to back some systems up, some may be password protected or have the software installed in a way that makes retrieval of the program impossible. Systems for which spare components are no longer available will make their way to the top of the pile when considering what devices to earmark for replacement.

Management resistance to patching and mitigations
Once a controls system is installed and its operation is verified, there tends to be resistance to making any modifications to the operating software or the system’s firmware. After a system has been shown to do the things it was installed to do, making a change of any sort runs the risk of interrupting production or causing unforseen operational problems. In the worst case, a firmware update could be installed which contained a regression that could cause software which previously functioned properly to stop working. If that firmware could not be rolled back, the system might be completely down until a fix was engineered - and that’s often just not a risk worth taking. The end result is that ICS devices are rarely patched against discovered vulnerabilities, even if patches are made available (which is itself a rarity.)

Multiple attack surfaces - including inadvertent “attacks”
It’s tempting to ignore all the networks which don’t directly connect to plant Ethernet, because they’re airgapped or they don’t alow reprogramming of the control system - but even simple PLC-to-PLC networks such as Modbus can wreak havoc if misconfigured. Most of the controller-level networks have no security models at all, and often the security tools which do exist aren’t implemented. A network such as modbus can, by default, permit reading and writing of every word of data in a programmable controller’s memory - a device on that network which is not necessarily malicious, but perhaps just misprogrammed, can overwrite crucial operational data in a remote system with no logs or indications that the event even happened. Because many of these simpler networks are used by SCADA systems, a change to a building management

computer or addition of a master temperature controller on one of these networks can suddenly cause equipment on the other side of the facility to mysteriously misbehave.

Security degrades over time

As controls systems remain in place, continuing to do the things we expect them to, we often don't notice maintenance issues which impact their overall security. Their CPU modules may have batteries which support RAM memory through power-downs - those batteries have a service life which can be a year, or ten years, depending on how often the system is powered down. Sometimes other modules have batteries, as well - I've seen a system where one battery was changed like clockwork, and the other seven were ignored for fifteen years because nobody knew they were there. Networks can become compromised, as new equipment is added, intelligent devices are replaced but not properly configured, or users add connections for their own different purposes.

Control systems are designed and purchased with the expectation that they will serve as long-term capital equipment. You may expect to trade up a laptop in two or three years - not so a building's heating system. ICS hardware is generally up to that expectation - vendors usually give years of advance notice before phasing out controls hardware, and there's usually some sort of replacement/upgrade path that's relatively painless. Also, you're not alone in having older hardware and needing to keep it running - a robust industry has developed in repairing and maintaining a supply of older ICS components. A much larger problem is the support hardware and software that's required to program a controller CPU, screen, or PLC module.

At this point, a ten-year-old system may use programming software that only runs on Windows XP, and requires a built-in RS-232 port. A twenty-year-old system (many of which are in constant use today) could easily have programming software that runs under MS-DOS. Some older systems can be maintained with more modern software, but there may be issues with transferring and verifying the user programs from the older software to the newer.

While ICS hardware vendors tend to be around for the long haul, the firms who integrate that equipment into control panels and program it may be much more short-lived. If you do not have your own backups of the controller's program, or if you find that they were on 5 1/4" floppy disks which have since degraded, you may reach out to the original system integrator and find them gone, or that they're still there but had a disk crash ten years ago. At that point, you may be left with a control system which you can pull an undocumented program from, or that's password locked and you don't have the password. Now it's time to find a vendor who can come in and produce new software from whatever operational specifications you have, or start specifying a replacement control system and scheduling the downtime required to change it out and validate its operation. Neither of those options are preferred, inexpensive, or quick.

Phase 1 - Inventory

What's out there? Knowledge of existing systems - Build a Controls System Inventory List

For each system, initially you need to know:

Name (for your reference) - be sure panels are labeled!
Number of panels/boxes/stations in the system
Physical location of each panel
Level of importance to business operation (a system may be more than one of these)

Mission Critical - Telescope Positioner, Data Center A/C, BioReactor, Data Collection
Building Infrastructure - Elevators, Building A/C, Sewage
Backup Systems - Fuel for Emergency Gens, Power Transfer
Important but Bufferable - Production Chromatography, Building A/C, Generator Fuel System, lighting controls (These are systems which can be shut down for a while, can batch up their output, or which may be able to run in a manual control mode for a while during maintenance.)
Monetary cost if offline - solar power, microturbines, Regulatory monitoring
Non-Critical - Lawn sprinklers, fountains, ancillary lighting, secondary solar collectors, offline experiments

Known level of connectivity to Internal Network, Internet, Wireless, and POTS
(Don't overlook serial connections to other devices and PCs, wired telephone modem connections, nor wireless connections to sensors and internet, including 3G/4G data modems. Watch out for devices which bridge networks together inadvertantly, and remember that sub-components like OITs and UPSs can have connectivity too.)

Rough date system installed
PLC Brand (If present)
Operator Interface Terminal Brand (If Present)
SCADA Software Brand (If Present)
OS of SCADA PC (If Present)
Are custom controller boards installed?
All known passwords to every system device

Also, if readily available:

Name & contact info of System Integrator
Name & contact info of Panel Builder (UL File number can help with this)
Are operation and maintenance manuals on hand?
Are panel drawings on hand?
Are spare parts on hand?
Is programming software and cabling on hand?
Are PLC, OIT, SCADA program sources on hand? How about data tables and recipes which are necessary to operate the equipment, but may not be backed up?

Phase 2 - Triage

Which systems will get the most attention soonest?
What's the rough budget for mitigation?
What's the rough timeframe for mitigation?
What level of defense will each system get?

Threat profiles:

Random intruders from the internet

Random intruders from the house network

Targeted intruders (industrial espionage) from internet or house network

Internal espionage via network or physical access

Extreme espionage (state actors) via diverse means

Non-malicious employees with physical access accidentally pushing buttons, moving valves

Protection against power/weather events

Ability to recover from control component failures

Protection against loss of communications

Systems which are not connected and aren't going to be connected may be able to receive a lower priority, depending on their location and their function. Remember - if there's something that's unconnected but mission critical, you may be better off to **increase** its connectivity in order to improve supervisory monitoring, especially if it's in a remote location which isn't normally manned, it's mounted outside, etc.

The result of the Inventory and Triage phases should be a list of all controls systems which are under your authority, how many panels each system has, where they are, what sort of controllers they have, how they're currently connected, how important they are to your mission (in various ways), the general age of the system and its software components, what level of documentation you have, what avenues of support may be available to you, what sorts of threat you think each system deserves to be protected against, and a feeling for the timeframe and the money you have to do the work.

Notice there's no "Penetration Testing" or "Verify Presence Of Possible Vulnerabilities" or even "Run NMAP Against The Controls System Network" in this procedure. Control systems can be very brittle, and the vulnerabilites can be severe - picture the ability to read or overwrite, via an unsecured network or serial connection, any byte of disk space on a PC. Now picture that you have no backups of that PC, and it just happens to host your main data store. That's the level of vulnerability which is built in to most implementations of the Modbus protocol, spoken by almost every PLC and OIT in the world.

Now add to that the fact that some controllers will respond to protocol commands on any TCP port - meaning that you can send Modbus requests on port 25 (usually reserved for email), or programming requests on port 502 (Modbus TCP). All the more reason that even a more or less "safe" scan can cause havoc on controllers.

Once you've got a feeling for what systems you want to address first, you can start on a more detailed analysis of those systems.

Can it be powered down/stopped?
Some systems have remained powered since install, others are powered down regularly.
If PLC battery has failed since install, powerdown could lose PLC program and recipe data.
Power cycling could also affect the process and may need very specific advance scheduling.

Backup Battery in PLC?
Has it ever been changed? Can it be changed while system remains powered?
Does the PLC alert if battery is low?

Is there a backup/reserve system?
Has failover been tested? Is it known currently to work?
Is the reserve system complete control, or partial?
Is human intervention needed to transfer over to the backup system, or to continue full control when backup is running?

Do we have the programming tools and cables for all components?
Some older programming software tools are difficult to find.
Some require older operating systems (XP, DOS) or laptops with built-in serial ports (not USB).
Some newer software won't work with controller programs which were built on older tools.
Most older PLC systems require special cabling.

Do we have software for all components?
PLC CPU, PLC Function Cards, Display Screens, 3rd Party Modules
(Not always easy to determine if cards get software loads, or to verify the software that's been loaded into the cards.)

If not, can we pull software from it?
(Many older PLCs have easily defeated passwording, we can often use that to our advantage to recover password-protected software.)
If we pull the software, it may only be the raw program, with no supporting documentation.

Can we verify that the software we have is what's running now?
Not all PLCs can do that. Some of the ones which can will only show that there is a difference, not give you a line-by-line breakdown of what has changed - sometimes you'll resort to measures such as generating text printouts, editing them to remove headers and footers, then using diff to compare the two.

Inventory of components and hardware/firmware versions
(This may require disassembly of stacked components to find labels, and/or connection with programming tools to find firmware info.)

Complete network map
This should include all controls components, switches, NAT routers, cabling, IP addresses, network types (remember, there are many possible ICS networks), other network addresses. What networks are supported by in-house IT Department, what are controls-system-only? Again, don't overlook serial data streams like Modbus, CANbus, HART, and so on, also remember wireless systems - point-to-point wifi, Zigbee, 3G/4G Data, etc.

Firewall rules
Obtain a complete listings of rules which relate to the controls system address spaces - look for open doors like PLC being placed in DMZ, or "Allow All" in ruleset.

Vulnerabilities
Once system discovery is complete, we can look in vulnerability databases to find out what potential threats there are against the hardware and networking systems we're using - but honestly, you may wind up just assuming the worst and working from there. If you assume that nothing is passworded, that any passwords which might be used are stored in plain-text or have hard-coded alternatives or are brute-forceable, that attached PC systems have programming tools sitting on them with the passwords readable, that operator interfaces
are accessible via VNC or web pages, that any network connection can be used to reprogram, modify data values, or crash a controller, then you'll be more prepared to look at ways to really secure these very vulnerable and important systems.

Attack entry points
Networks, physical access, connectivity, accidental operation or shutdown, power issues

Possible results of attack
Target value to attacker
Target value to owner

Budget for remediation
First year budget may all be for discovery

Level of provider support
Drawings, software available? Upgrades?
If the Systems Integrator or Manufacturer are still available, they may be the best avenue for remediation - however, it's highly unlikely that they've given any thought whatsoever to ICS cybersecurity concerns, especially regarding older systems.

Level of manufacturer support
Hardware and Firmware upgrades?)
Many manufacturers have simply ignored all notifications of security issues with their hardware, some have provided firmware upgrades, some have released hardware with updated versioning numbers which contain updated system ROMs that fix vulnerabilities.

At this point, we know What Systems we have, and we know What Order we are going to address the systems in, and we know roughly what level of work we intend to do to mitigate the various vulnerabilities we see.

Phase 3 - Backups

Backing up ICS systems is not as simple as plugging a USB Hard Drive in and running some software. There are often many physical components required to really have a "bare metal" backup/restore solution - the good thing is, due to the nature of control systems, generally once a full backup is made, it doesn't need to be repeated often since the programs (and often the data) remains unchanged for long periods of time.

Batteries - not just CPU & OIT, some comms and motion cards as well.

Hardware required for programming
- Needs native serial port?
- Needs special cables?
- Needs PCMCIA Card?

Programming Integrated Development Environment
- Activation of software?
- Hardware drivers?
- Dongles (Parallel, USB)?

OS to support IDE
- Could be MS-DOS, Windows XP, Windows 7
- Can be quite specific as to patch level

Firmware version that's currently running on controllers

PLC / OIT program to load

DATA For PLC program if that's stored separately

Comms drivers for OIT - verify driver versions

What media is everything stored on? What media does the hardware need?

Consider obtaining/making a VirtualBox VDI file with everything needed.

Consider obtaining a laptop with everything configured, then label the heck out of it so it doesn't disappear when somebody is tidying up or getting rid of old, unused stuff. Remember that solid state hard drives can lose data within a year of sitting unpowered - it's a good idea to insist on a conventional hard drive in the laptop rather than a solid state drive.
(Special cabinets exist to store laptops permanently powered and sitting in suspend mode.)

Programs that get loaded into special function cards
- Communications Modules
- Co-Processor Modules
- Motion Controller Modules

Cabling for these special function modules

IDE software for these special function modules

Verify anything that can be verified - some things can't, which puts them higher on the "Replace" list.

Bring in component-level spares if you don’t have them

- Power Supplies
- Racks
- CPU + any memory cards or add-ons
- I/O Cards
- Special Function Modules

It may well be worth building up a complete spare controller system which will let you test the restore process - even to the point of making a restore, then (at a propitious time) swapping out the CPU and other components for the backup units to make sure that they properly function.

Phase 4 - Take Action

For each system, decide what level of response it will receive:

Ignore
What the vast majority of vulnerable system owners are doing already - the difference being, we actually know what the system is, and we are making an informed decision to ignore it (for now.) Depending on the system, its function and benefit to the organization, and how it's connected to the outside world (if at all), it may have such a small risk profile or low vulnerability that it's not worth doing anything to it - but at least you'll _know_ that and have acted accordingly.

Monitor/Protect
(Not modify the core system, but add features to prevent or detect tampering.)

This includes physical access control to panels and operator devices, tamper switches on doors and on manual controls like valve handles, Intrusion Detection System rules to observe network traffic and alarm on changes from a preset standard (ICS traffic tends to be extremely predictable and very easy to set useful alarms on.)

Isolate
(Cut the cord - possibly lose functionality)
Some steps can be as simple as turning a key from "Remote Program" to "Run". Now the controller will need manual intervention before the PLC program can be modified - if it's a remote or unmanned station, this may not be a practical answer.
If someone says "But we need to be able to reprogram remotely!" feel free to ask "Why? What haven't we considered in the programming that requires it to be modified on a regular basis?" Remember that the overwhelming majority of ICS systems are programmed at installation, then rarely if ever modified.

Could wind up losing remote access for vendor service - is that function being used? Often it's in place, but dormant. You may well have been paying for service that never occurs.

Could wind up losing remote access for monitoring or other important data collection tasks. Note that not all data collection is continuous, especially in older systems - often a system will save days' or weeks' worth of log information, and a desktop PC somewhere will remote connect weekly or monthly to pull data from the PLC. Other systems will send an email on a timed basis, or even dial out on a modem to connect to a remote data collection device.

Enable and/or Upgrade passwords wherever possible - PLC, OIT, Smart Switches/NAT Routers/ Gateways, SCADA PCs. This is far from a panacea, but it's all part of a program of Defense in Depth - everything that slows down an attacker increases the chance they will be detected before doing damage.

Remove configuration/programming tools from local PCs which can reach the PLC/OIT unless there's a **really** good reason. Leaving those tools available on a laptop that's normally not connected and lives locked in a drawer is often more than enough.

Manually powered-up ethernet switch or NAT/VPN Gateway with timeout - only allow access after manual intervention, and only for a limited time. This is useful for situations where access is needed only for remote maintenance work.

Timer-powered ethernet switch or NAT/VPN Gateway - only allow access during pre-programmed times, like daily between midnight and 00:15. This is useful for batch data collection, cuts amount of time available to intruders to try their attacks.

While air gapping of any sort is not a perfect solution, it's a huge decrease in potential attack surface, and every little bit helps.

Firewall
(Maintain or even gain functionality)
Many industrial control protocols are wide open and have no means for authentication. Many implementations allow for full read/write access to the entire PLC's data memory, some also allow programming and even firmware access without forcing authentication.

Dedicated protocol firewall devices exist for some PLC protocols, primarily for Modbus since that's the "least common denominator" of PLC protocols. Newer units can automatically build firewall rules based on observing the (rarely changing) data patterns of ICS communications.

Stand-alone protocol converters can be used as firewalls, limiting the amount of accessible data and making most or all of the data be read-only - sort of like a “Data Diode”, but without the need to invent special protocols for one-way physical data transfer.

Many operator displays have protocol conversion features, which allows the addition of data monitoring and control displays while improving system security - possibly also adding remote monitoring in a more secure way while maintaining existing controls connections.

Interposing a small conventional hardware firewall between a control system and the house network will allow you to limit the IP addresses which can query the controllers - control systems generally have a limited number of IPs which will attempt to access them.

Add a "communications interface" PLC to the controls system network, which is unable to write to the other PLCs and is only written to and read by them. It becomes a sort of DMZ for controls system data, any external devices read from and write to only this new PLC.

Intrusion prevention systems can be programmed to monitor traffic between house network and ICS network, that traffic tends to be limited and predictable - bit-bucket any traffic not from proper IPs or which is attempting to access the wrong ports.

Update hardware and/or firmware
(Form/fit/function replacements)

Many legacy PLC systems don't have updateable firmware. Some security issues can be addressed with an updated module release, which is almost always backwards compatible with existing systems. While fixing known issues with device firmware is helpful, it is a rare system where this would be the only action indicated.

Upgrade system
(Replace PLC, replace OIT, but keep panels in place)

The PLC is just one part of a control panel. Often it's worth leaving the other components in place, and changing just the controller.
Look at issues of form and fit, whether wiring will reach the new PLC terminals. If fit is an issue, it may be possible to add a small panel containing just the PLC, and cable over to existing connection points.
Research to ensure compatibility with other systems which communicate with the device being upgraded.
Logistics of porting software from one platform to another, even when the new PLC is the same brand but a newer model.
Addition of security and monitoring features for vulnerable physical devices

- Cabinet locks
- Door tamper switches
- Valve tamper switches
- Power and air supply pressure monitoring

Replace entire system with a more modern system.
(Can be a very expensive option, but may be less costly overall than spending lots of time digging into poorly-documented existing systems.)

This option is often less painful because operation of the new system can be bench tested to some degree, and transition can be scheduled for an acceptable down-time (or some dual-operation method devised so that the change-over is done without losing complete functionality.) This can also be justified by increases in function, improvements in overall maintainability, even power efficiency.

Recognize that installing new equipment without paying proper attention to security during the specification and design phase can result in a system which is much more vulnerable than the one it has replaced.

If “Replace” is the chosen option, it’s likely that the research you’ve done into finding out as much as possible about the existing system will make the process of specifying, obtaining, installing, and commissioning the new system much easier and provide a much better result.

Sample Specification Language for Replacing Existing Control Systems

NOTE: Not all these ideas will apply to your systems, some are contradictory and represent different levels on the tradeoff of security versus ease of expansion.

General Good Ideas

Vendor shall produce a detailed description of system operation, specifying the function and behavior of each physical input and output point, describing all internal and external data pathways, and enumerating all data tables used for external control and monitoring of process logic, system status, and warning and alarm conditions. This description shall be used by the vendor to produce a system test plan which will be used to verify correct operation of the system at commissioning.
{Language like this is because if you don't ask for it, they won't provide it.}

Panels shall be built and labeled to the UL 508A standard.
{Just a good idea. UL makes sure that devices are designed not to catch fire. We like that.}

Panel Longevity/Maintenance Language

Controls hardware shall be PLC-based. HMI systems shall be dedicated panels..
No commercial general-purpose computing hardware shall be used.
PLC and HMI hardware shall have a minimum manufacturer support window of ten years.
{Trying to avoid having an entirely unsupportable computing platform six years from now.}

Panels shall be labeled internally with the integrator's name and contact information, and (if different) the manufacturer's name and contact information.
{This will matter after the panels have been in service for a decade.}

If any equipment in a panel uses a replaceable battery, that panel shall be labeled externally with the battery function, part number, installation date, and recommended replacement date.
{This makes the whole issue of lithium battery lifespan much easier to deal with.}

A laptop computer with a magnetic hard drive (not SSD) shall be provided which contains all development software required to program the PLCs and HMIs. All required software licenses shall be installed and registered to [Insert Facility Name Here]. All PLC and HMI programs, as well as all CAD drawings, manuals, and ancillary information shall be stored on this laptop. The laptop shall be physically labeled showing that it is the programming tool for [Insert Project Name Here].
{This may become very, very important after several years. Alternatively, you can ask for Virtual Machine instances to be created, assuming the licensing can be arranged to allow moving the VM from one hardware platform to a newer one as time progresses.)

Data Security Language

Programmable Controllers shall have a physical switch to select between Run, Remote Program/ Run, and Program modes.
Controllers can not be programmed or remotely stopped when switch is in “Run” mode.
{I like this concept, but it pretty much locks you into Allen Bradley controllers.}

No PLC or HMI systems shall use wireless communications for any reason.
{Most locked-down, probably won’t affect your system’s design or cost unless you’re trying to get data from some remote device.}

PLC or HMI systems which use wireless communications must use only dedicated industrially-hardened point-to-point systems (i.e. Zigbee, WirelessHART), NOT systems which use 3rd party external gateways (i.e. 3G, 4G, LTE) or consumer-grade general-purpose networking (WiFi, Bluetooth)
{Use this if you need to get data from a remote tank or facility - if you can possibly avoid sending data over commercial airwaves, you’re better off from a security standpoint.}

No ethernet cable may be run between panels. Ethernet may be used inside of panels (i.e. for communications between PLC and HMI) but must be point-to-point - no switches, routers, or NAT routers may be installed in the panel. Ethernet to serial bridges (protocol converters) may be used to convert internal ethernet to an external serial protocol.
{Most restrictive. This is similar to language I’ve seen used by the Federal Reserve Banks.}

Communications between panels must not be TCP/IP based. Acceptable protocols include (but are not limited to) Modbus RTU, DeviceNet, ProfiBus, DL485, CC-Link, CANBus, and BACNet. {Again, Federal Reserve.}

If TCP/IP communications is used between panels, all IP switches shall be DIN rail mounted in the panels. IP Addresses used shall be from a pool defined by the IT department which does not intersect with the house IP pool. Any connection between house networks and controls networks shall be via a separate, dedicated Ethernet port on a controls device or via a NAT translating gateway located in one of the panels. Vendor shall provide a complete network map of the system, showing all ports in use, all data paths, and all protocols used.
{This maintains a separate network which has extremely limited points of contact with your house network. You may want to specify IP switch brand and part number, ensuring your IT deparment’s familiarity with the hardware and its capabilities. Note that not all controls network protocols behave nicely on all ethernet switches, especially smart switches. You may want to monitor spanning ports on these switches with your IDS.}

Controller data networking shall be segmented into different networks such that only the data required for a given function will be available for that function. For example, SCADA monitoring would be accomplished through a limited Modbus connection which permits reading only the data that is required by the SCADA system’s displays, and only permits writing of any specific items which can be changed or controlled by the SCADA system.
Attempts to read or write any other data areas will be blocked.

Side note: ICS/SCADA as your very own "Shadow IT" department...

Controls system networks versus house ethernet:

IT department thinks controls people are sloppy, don't document anything, don't know anything about networking or the requirements of the hardware they're bringing, don't know what operating systems or software versions they're running, never patch or update anything, insist their data can't co-exist with any other data on the network, use weird IP addressing schemes, don't use any sort of IDS systems, have problems with advanced networking technologies, bring in thousands of dollars of hardware and software which doesn't get tracked, has no asset tags, no upgrade plans or maintenance agreements, provide no technical support, and generally are total control freaks.

Controls vendors think house IT people can't deal with anything that doesn't run Windows (and don't ever mention Windows CE, because they'll insist on trying to patch it every other Tuesday), take forever to do simple things like assign an IP address, insist on putting controls-related switches in data center racks clean across the building, use networking hardware which buffers packets to the point that PLC comms stops being reliable, will randomly reboot network switches (causing controls calamities) or just disconnect cabling to check connectors or dress wires, and generally insist on decreasing the stability and reliability of a controls system to the point that it only works on Thursday afternoons.

(And they're both correct, to a point.)

Problems of integrating the knowledge bases of controls system networks and house IT networks:

The large majority of ICS systems with communications capabilities are not integrated into house IT networks. They have isolated networks which use a variety of protocols (see the Not-So-Fun Facts above) hardware implementations (Ethernet, Twisted Pair, 3G/4G, direct microwave, POTS, leased line, parallel bus, digital and analog I/O)

Unusual/stealth communications - you will sometimes find that systems transfer information in unexpected ways, sending each other an email, placing data on a web page, or using the physical systems under their control as a signaling method.

Many IT departments consider ICS the purview of the facilities engineers. Many facilities engineers consider ICS hardware to be a “Black Box” which will basically continue doing its thing until cockroaches rise up and take over the planet. Backups are rare, may be on old media (5-1/4“ floppies are common) and may be incomplete, or out of date, or impossible to verify.

As more and more controllers and displays sprout ethernet ports, it’s becoming critical to get IT involved with ICS. There are panels out there with $5000 CPUs and without asset tags. There are unmapped networks, unprotected data paths, and unknown communications methods. Critical infrastructure is depending on systems with no backups, no recovery plan, no scheduled maintenance, and no spare parts. IT is capable of dealing with all the issues which securing ICS brings, but IT is rarely invited to weigh in, and rarely is given authority to catalog, back up, and secure these systems.

Links

Dedicated ICS Protocol-Aware Firewalls:

https://www.belden.com/product/industrial/networking/security/tofino-xenon
http://www.moxa.com/product/Industrial_Secure_Routers.htm
http://www.waterfall-security.com/solutions/industrial-protocols
https://www.fortinet.com/content/dam/fortinet/assets/alliances/sb-fortinet-nozomi.pdf
http://www.sequi.com/portbloque-e.htm

Protocol Translators which can act as isolating firewalls:

http://www.redlion.net/products/industrial-networking/communication-converters/protocol-converters
http://www.sierramonitor.com/connect/all-protocol-gateway-products

Operator Interfaces which can do protocol translation as well as add real-time visual monitoring:

http://www.redlion.net/products/industrial-automation/hmis-and-panel-meters/hmi-operator-panels/
http://www.beijerelectronics.com/

Other Interesting Things:

History of PLCs:
http://www.control.lth.se/media/Education/DoctorateProgram/2012/HistoryOfControl/Vanessa_Albert-PLCDCS.pdf

Rockwell (Allen-Bradley) presentation on ICS CyberSecurity:
https://www.rockwellautomation.com/resources/downloads/rockwellautomation/pdf/events/packexpo/PE-07_Securing-Manufacturing-Control-Networks.pdf

Forums for PLCs and Controls:
http://control.com
http://www.mrplc.com/

DHS ICS CyberSec Training - Idaho Falls, Idaho:
https://ics-cert.us-cert.gov/Training-Available-Through-ICS-CERT#workshop

Sources for older PLC and OIT hardware:

http://radwell.com
http://ebay.com
http://labx.com
http://amazon.com
http://alibaba.com