Uncovering Cyber Flaws

To ensure the safety and security of the process, company, and staff, find the vulnerabilities and break a negative chain of events

By Eric Byres and Matthew Franz

FAST FORWARD

- Security vulnerabilities in software have enabled hackers, viruses, and worms to run rampant, but flaws also are prevalent in computer hardware, including industrial controllers.
- Many vulnerabilities in the industrial world are the result of administrative security failings rather than software failings.
- Classify flaws by how or where they enter into a product's life cycle—whether at the product design, implementation, configuration, or other stage.

Ten years ago, the IT department at a major food products manufacturer hired a consultant to scan its corporate computer systems for possible security vulnerabilities. The scan would likely identify a number of these potential vulnerabilities before they became a serious security issue.

Included among the systems to be scanned was a new Ethernet network on the production floor that connected all the programmable logic controllers (PLCs). Unfortunately, when the scanner software sent out a certain class of ICMP redirect messages, all the PLCs crashed, resulting in reported production losses exceeding $1 million. These PLCs had serious, but undocumented, vulnerabilities in their communications firmware, enabling a single malformed packet to crash them.

Today, security vulnerabilities in commercial operating system and office software have enabled hackers, viruses, and worms to run rampant in the IT world, often making headlines. But vulnerabilities can be just as prevalent in computer hardware—printers, routers, cell phones, and, most importantly for automation professionals, industrial controllers.

Whether it's a PLC and distributed control system (DCS), a remote terminal unit (RTU), or an intelligent electronic device (IED), the facts are human beings design and implement these systems and make mistakes, and those mistakes translate into vulnerabilities attackers are eager to exploit.

Procedural failures, external threats

Most of hardware flaws get little publicity, yet even in major brands of controllers, one can find flaws even an unskilled hacker could exploit. For example, one PLC failed during scanning with a standard security port scanning tool, indicating a serious transport control protocol (TCP) implementation issue. Further investigation of this device showed its behavior completely violated the TCP specifications. Major vulnerabilities like these can seriously impact production, even without the help of a hacker.

Vulnerability is "a flaw or weakness in a system's design, implementation, operation, or management that could be exploited," according to the Internet Engineering Task Force's (IETF), *Internet Security Glossary.* Results are loss of confidentiality, integrity, or availability of an application or resource, which leads to

an undesirable consequence for system owners.

In January 2003, the Slammer Worm took advantage of a known software flaw in Microsoft SQL Server, causing numerous SCADA and control systems worldwide to lose critical network communications. This loss of network availability caused the loss of view or control of the system in the best cases and loss of production in other cases. Fortunately, Slammer was a relatively benign worm, which didn't directly target control system hardware or software in its payload.

Today, because malware has become increasingly sophisticated targeting specific applications on the infected host, it is conceivable that a piece of malware could alter or delete a human machine interface (HMI) application, OPC server, or data historian. Similarly, just as botnets (networks of computers controlled by a hacker) have the capability to flood Web sites with malicious connection requests, the target of attack could just as easily be a PLC or protocol gateway on a critical control system.

Many, if not most, vulnerabilities in the industrial world are the result of procedural or administrative security failings rather than software failings. When combined with technical software vulnerability, the result is a probable exploit. For example, permitting unnecessary applications, like game software, to run on plant floor HMI computers is an example of a procedural vulnerability. A poorly configured firewall (or no firewall at all) is another. Add a technical vulnerability in the game software, and the result is a chain of vulnerabilities that can be exploited by a disgruntled employee, commercial hacker, or automated threat. Finding the vulnerabilities and breaking this chain is critical for the safety and security of the process, company, and staff.

Problem gaining attention

Vulnerabilities are a fact of life in virtually all microprocessor-based systems, such as routers, printers, PLCs, servers, and workstations.

While not every flaw is exploitable by an attacker, research indicates about 8% of bugs are exploitable, translating into five vulnerabilities per small device. Unfortunately, this estimate is likely conservative in the industrial control world. Modern controllers are designed on embedded systems platforms with CPU and memory limitations, resulting in design compromises. And because the primary focus in any controller is control functionality, security design is likely to take a back seat.

So far, the hacking community simply hasn't focused its attention on industrial devices, and the vendors have not been forthcoming with announcements when they discover faults. This lack of attention on control systems is not likely to continue for much longer; since late 2003, talks on hacking SCADA systems have been showing up at hacker conferences worldwide.

Classifying technical weaknesses

To stop hackers from exploiting the vulnerabilities, we need to find them first. Start by classifying them by how or where they enter into a product's life cycle. Here are four general classes:

- **Inherent protocol vulnerabilities:** TCP/IP protocols such as Telnet and file transfer protocol (FTP) specify weak authentication credentials, making hacking them a breeze. In the industrial world, serial and Ethernet-based control system protocols lack even the weak authentication. To implement these protocols, the vendor can modify the implementation of the protocol so it no longer follows the specification or use it knowing it is vulnerable. Most vendors choose the second option.
- **Product design vulnerabilities:** These typically result from an inadequate understanding (or improper assumptions) of the operating environment. Technical or market factors may also lead to known or unknown design vulnerabilities. For example, hardware limitations, whether CPU, file system, or memory limitations, may make implementation of authentication or encryption impossible. Even if threats are known, the rush to bring a product to market or the need for user convenience features may take priority over proper security design.

To stop hackers from exploiting the vulnerabilities, we need to find them first. Start by classifying them by how or where they enter into a product's life cycle.

RESOURCES

Securing the power grid
www.isa.org/intech/050034

Cyber security meets plant politics
www.isa.org/intech/050025

Homeland Security test bed
www.isa.org/intech/050041

ISA-SP99 Manufacturing and Control Systems Security Standards Committee
www. isa.org/isasp99

***Industrial Network Security* by David J. Teumim**
www.isa.org/networksecurity

***Electronic Security for Manufacturing and Control Systems – An Overview* by Robert Webb**
www. isa.org/secguide

Process Control Security Requirements Forum (PCSRF)
www.isd.mel.nist.gov/projects/processcontrol/

- **Implementation vulnerabilities:** These flaws are the result of errors in the product development process and are essentially quality control issues, such as programming errors, buffer overflows, format string errors, and failure to handle "specially crafted" packets or traffic floods that result in device reboots. For example, one PLC communication module crashes when it receives a certain malformed Web request. In the IT world, these vulnerabilities are most often discussed in the vendor advisories and are the primary focus of hacker or worm exploits.
- **Misconfiguration vulnerabilities:** Not applying security patches or disabling unnecessary software features is probably the most common example. Another cause is user confusion as to the correct settings for security. The majority of wireless systems are insecure when deployed because users do not understand how to configure them correctly.

Sharing blame

It's easy to blame vendors for the vulnerabilities, but end users, consultants, integrators, vendors, and the standards bodies all share in the problem. Is misconfiguration vulnerability the fault of the user or integrator who deploys the product or the vendor who makes secure configuration difficult?

The various standards bodies responsible for industrial protocols must begin to define security as part of any standard, including defining appropriate security features into new or revised protocol specifications (based on formal threat analysis) and defining appropriate external countermeasures when robust security features are not available in the older specifications. These bodies also can help by defining implementation guidelines and best practices based on likely threats to their protocols. One example is *IEC TC57 WG15 (Power system control and associated communications—Data and communication security)*, currently working on securi-

Flaw-Finding Tools

Numerous studies have demonstrated the cost savings when vulnerabilities are discovered early in a product's life cycle. Consider some of the general types of tools and techniques available to vendors, consultants, integrators, and end-users. The best choice depends on the class of vulnerability you're seeking and the system or device being tested. Create a proper test plan that outlines the objectives of the test, the exact methodology and procedures of the test, and what the test will and won't discover (test coverage). Just deploying a tool without a plan may find a few "lucky" vulnerabilities, but it can't provide a systematic analysis of a system's flaws.

- **Application and device profiling:** Scan devices to determine possible vulnerable services operating on them and fingerprint to uncover underlying operating systems. Tools like nmap, SuperScan, and Xprobe are freely available on the Internet.
- **Known flaw checks:** Scan for known vulnerabilities in traditional IT systems. (Products such as Nessus, FoundScan, and Internet Security Scanner have been popular with IT administrators seeking unpatched computers on their networks.)
- **Resource starvation testing:** Check what happens when a device receives an overwhelming number of traffic or requests that exhaust its resources. While some free tools are available on the Internet, use purpose-build traffic generators like SmartBits.
- **Protocol robustness testing:** Use grammar and fuzz techniques to create directed pseudo-random sets of network messages, which then transmit to target devices to create and detect unexpected behavior. (Try PROTOS suite and its commercial counterpart Codenomicon, or Mike Frantzen's ISIC, which is free.)
- **Source code analysis:** Automated tools will search source code for constructs that might indicate software vulnerabilities. Since vendors typically are the only groups to have access to a system's source code, this is a vendor tool.
- **Binary analysis:** If product source code is unavailable, analysis of binary executables and libraries alone can work to discover software vulnerabilities. These tools profile and analyze the execution environment to identify application and control data paths and can uncover issues with third-party libraries.
- **Threat modeling:** Early in the product's life, conduct a thorough analysis of attacks a product will face to decide appropriate security features and to ground vulnerability testing. A threat model should capture the security relevant attributes of a system, including hardware and software components, data stores, information flows, and critical processes, as well as assumptions about the attacker (access, techniques, objectives) and trust boundaries. One well-known method of conducting threat modeling is attack trees. This area needs more research.
- **Formal methods:** These are mathematical techniques and languages for defining and then analyzing a system or protocol for inconsistencies or flaws. Examples are the Specification and Description Language (SDL) and the Testing and Test Control Notation (TTCN). Some experts advocate these as a means of increasing the reliability of many safety or business critical systems, including communications systems. Unfortunately, the industrial uptake of such methods has been slow, most likely due to the perceived difficulty of the mathematics involved and the lack of tool support.

ty for protocols such as DNP3, ICCP, and IEC 61850.

For vendors, product security needs to become a part of the product development cycle, including establishing security engineering groups that conduct vulnerability testing of all hardware and software and define internal security standards. One example in the IT world is Microsoft's system Development Lift Cycle (SDLC) program.

Once the product is ready to ship, the vendor needs to help the end user deploy it correctly. The vendor can develop configuration hardening guidelines to define features that can be disabled and provide example firewall rule sets (and other network security countermeasures) to implement multiple layers of protection. It is also useful to establish *"/security"* Web sites and provide *security@* e-mail addresses to allow customers and security researchers to communicate new ideas, discover vulnerabilities, and ask for emergency security assistance. When vulnerabilities are discovered in released products, vendors need to respond to the controls community with clear policies, including timelines for disclosing internally and externally discovered vulnerabilities to end users and to government coordination centers.

It's the end users' responsibility to consider product security when making purchasing decisions, encouraging vendors to make investments in personal, processes, and tools to secure their control products. Don't assume a product is secure. Perform periodic vulnerability assessments to check for known vulnerabilities as a regular part of maintenance shutdowns. If you find something, report it to the vendor. And don't forget all the procedural vulnerabilities—follow the industry standards and best practices for control system security.

Ultimately, security vulnerabilities are a quality assurance issue. We have come to expect standard testing of equipment with respect to other measures of quality such as radio frequency interference (RFI) immunity, flame safety, and reliability. There is no reason why industry can't develop similar test standards to establish minimum levels of security.

As industry gets more experience with proper security testing, it can move toward more in-depth security testing throughout the engineering, design, and factory acceptance stages.

ABOUT THE AUTHORS

Eric Byres (Eric_Byres@bcit.ca) is a professional engineer and research leader at the Critical Infrastructure Security Centre at the British Columbia Institute of Technology. **Matthew Franz** (mdfranz@threatmind.net) is a security researcher with more than 10 years of experience identifying and testing network vulnerabilities in a wide variety of products and applications.

Share the online version at www.isa.org/intech/060002