Keywords: smart grid, smart meter, power meter, electricity grid, infrastructure, stuxnet, flash, aes, elliptic curve, industrial, cyber warfare, cyber attack

TUTORIAL 5445

Stuxnet and Other Things that Go Bump in the Night

By: Kris Ardis, Business Director
Nov 27, 2012

Abstract: Stuxnet, a sophisticated virus that damaged Iran's nuclear capability, should be an eye opener for the world. We can choose to learn something very narrow (how to combat the Stuxnet virus) or we can choose to focus on the larger goal of thwarting the next type of creative cyber attack. Unfortunately, critical industrial infrastructure is not currently designed with security as a key goal, leaving open multiple avenues for an educated and funded attacker to create massive problems. This tutorial outlines some basic concepts that engineers and product definers should consider to make sure their new projects stay ahead of future threats.

A similar version of this article appeared on *EDN*, October 11, 2012.

What Is Stuxnet?

Stuxnet is at its root a computer virus. But it is a very sophisticated one. Most viruses are passed by email, file attachments, or USB sticks, and cause mayhem by exploiting weaknesses and flaws in modern PC operating systems. They can spread quickly, but many commonsense approaches to handling email, internet downloads, and antivirus software can help mitigate their threat. Normal computer viruses are indiscriminate: they attack every PC they touch. Stuxnet is different—while it spread in a similar manner as other viruses, in most systems it had no effect. Stuxnet was designed to:

1. Infiltrate a PC through the typical virus pathways. USB sticks were highly effective.
2. Confirm whether the host PC's location was Iran.
3. Establish whether there was a certain type of programmable logic controller (PLC) connected to the PC.
4. Check if there were a specific number of those PLCs attached.
5. Confirm whether those PLCs were connected in a very specific arrangement and controlling a particular piece of equipment.
6. Reprogram the PLCs to alter their behavior, but report diagnostics that everything was fine.

This sounds pretty complex. But it is only part of the story; the number of PLCs and the configuration

that Stuxnet targeted prove that the virus was clearly defined to attack a specific nuclear facility in Iran, and to slowly and permanently damage the centrifuges there to set back the Iranian uranium enrichment program. The damage was intended to be done over time to confuse operators—they would not think of a virus or even any kind of IT problem until they were deep into diagnosing the issue.

Evidence suggests that Stuxnet was successful at permanently damaging 1,000 centrifuges in the Iranian nuclear facility. There is speculation that the virus was designed by the United States, Israel, or both. Note that Stuxnet did not actually damage most systems it infected—it was a highly targeted attack. This allowed it to spread to its target before it was detected and antivirus companies were alerted to its presence.

Why Should We Worry?

We should worry for many reasons. This is a model for effective cyber warfare. It has likely inspired the IT efforts of nation states all over the world. While the United States/Israel may have achieved their targets in this opening strike, has a Pandora's box been opened where we will now see retaliatory attacks that target U.S. or other allied interests? Will attacks in the West model Stuxnet, but broaden the focus to cause maximum damage to critical infrastructure?

There is another reason to worry: Stuxnet shows that critical infrastructure is not protected. Security is often not a design consideration in many applications, including industrial control. There is a basic rule about designing security into an application: it never happens because it is a good idea. It only happens because (1) it is dictated by the situation (e.g., a certification that requires a level of security); or (2) in reaction to public or damaging exposure of a security flaw. Note that the first reason is pre-emptive, and the second is reactive. What situation are we in now?

In the post-Stuxnet era, are we asking ourselves the right questions to try to solve this problem? Some are asking, "How do we combat Stuxnet?" and "How do we secure a USB stick?" Solutions resulting from these questions are bandages, similar to airport security measures that require us take our shoes off to check for bombs. Yes, inspecting our shoes will detect the attack that was nearly effective, but it doesn't check for the next type of creative attack. Even if we believe USB memory sticks are a weakness and a path for spreading viruses, a secured USB stick won't help—it is easy enough to create a USB that looks like the secure one, and load it with the virus.

The Root Problem

In our shoe bomber example, the root problem is easy to identify: How do we ensure that everyone who has access to the plane (i.e., passengers, maintenance, crew) has good intentions to travel on, repair, or operate the aircraft? The question is easy to identify, but not to answer.

It is similar with Stuxnet. The question needs to be "How do we protect critical infrastructure so that it operates in the way it was intended?" The answer is accordingly complex, but at its root is a need for embedded systems to consider security in all aspects of design. Engineers typically design a system first for nominal functionality, and if they consider security at all it is generally an afterthought, applied over an existing and designed system. Everyone involved in the security industry is familiar with the notion that "security will be designed in later"…which actually means that the solution will never be secured. Security added at such a late step can only be superficial, only protecting against obvious threats.

Let's use another piece of equipment as an example: a smart meter is first designed to meet metrology accuracy requirements and implement a communication stack correctly. In that communication stack, the Advanced Encryption Standard (AES) is usually added to protect data that the meter generates and sends back to the utility, and to protect the commands sent from the utility to the meter. The answer from many utilities, communication providers, and meter manufacturers is then that the meter is secured because it uses AES to encrypt the data going in and out of the meter. But this is a bandage—the threat is identified as only the "tampering of data coming from the meter and commands going to the meter." This is the equivalent of taking off our shoes in the security line at the airport—we have a single threat identified, and a single bandage to address that single threat. There is no consideration for the next new threat. In the case of the meter, the next clever threat might be against the physical meter itself—can an attacker access the physical meter and load a new application program into the firmware, allowing the attacker to take control of the meter? We now have a new threat—a malicious firmware programmer. But do we develop another bandage or instead think about a broader approach to security?

A Broader Approach

Bandages will not work. They only prevent the past attacks. Our adversaries are smart enough to move on to a new trick, but our security approaches are not. Is a terrorist today going to approach a plane with a bomb in their shoe? Probably not. They will develop a more creative attack—perhaps they will penetrate the security at a much smaller airport, then fly a private plane to a larger airport where they will have access to larger targets.

A broader approach to security requires some fundamental changes in how embedded systems are currently designed. Some industries may have specifications or security requirements to be met. However, in the smart grid and most industrial applications, we find a lack of security guidelines, leaving engineers directionless. In these cases, engineers and product definers should take a more fundamental approach to security:

1. Security needs to be a key goal before the product is defined. Threats must be assessed as part of the conceptual phase of a product, and components should be selected with security in mind.
2. A secure product must consider cradle-to-grave security. This means controlling and validating the manufacturing process such that an infiltrating attacker cannot provide their own firmware or manufacture their own board design instead of the intended product.
3. Firmware/software and communications need to be trusted. In order to trust something, you need to validate it. So before operating systems and applications are loaded to run, they must be cryptographically verified as trustable. Before a command is executed (like "disconnect this user from the electrical grid"), the command needs to be cryptographically validated. This leads to another set of questions on authority—who has the authority to load the device with new application code or send it a command?
4. The process of designing secure equipment should not be a closed process. This may seem

contradictory, but working with industry or academic security consultants can help you identify threats in your system and problems with your implementation. Even in areas where standards are lacking, a security consultant or third-party security evaluation can help with overall threat assessments. Security designed in a vacuum has a problem—it is secure enough to stop only the people who did the design!

5. Avoid security through obscurity. Secure techniques and algorithms should not rely on the fact that documentation on an algorithm or scheme is not publicly available. The best security techniques in fact do not rely on secrecy of the algorithm or protection scheme. As an example, many engineers believe that communication schemes that employ frequency hopping are secure because the "hopping" makes it difficult for an attacker to latch on to the current active frequency and decipher the communication (i.e., the frequency carrying the data is theoretically hidden). The truth is that the frequency hopping must be deterministic somehow (or the other side could not predict the hops), and could therefore be determined by an attacker.

One common misconception is that the technology to truly secure an embedded system does not exist. This is absolutely untrue—the financial terminal industry and many government applications have built equipment against security standards for years. Processors can embed security technology ranging from simple (only an AES engine) to complex (real entropy generation, modular math acceleration for asymmetric cryptography, tamper detection, environmental monitoring, code scrambling or encryption, and more). The National Institute of Standards and Technology (NIST) states in NISTIR 7816 that technologies such as real entropy generation and asymmetric crypto are difficult in embedded systems.1 Open any financial terminal and you can see that the technology exists today.

Achieving Perfect Security

If we follow the approaches above (committing to security as a goal, securing the life cycle, validating inputs, consulting with outside experts, and avoiding security through obscurity), can we achieve perfect security? No. In fact, you can never achieve perfect security. The closer you get to perfect security, the more exponentially expensive it gets to close the gap. Security design is very much a game of trade-offs between the threats, the risks, and the cost to secure against those threats. Vigorous adoption of the approaches listed above will definitely help close the security gap so you are closer to the ideal of "perfect security." If the PLC devices attacked by Stuxnet had cryptographically validated their new application code, the attack path would have not been possible. The Stuxnet perpetrator would have needed to consider another path—perhaps attempting to get the root keys through some social engineering path (more threats to be identified in the early phases of design!).

Embedded security is only a piece of the equation as well. A security mindset must be applied to the entire system. However, embedded security is often the most lacking in industrial control systems today. Implementing the above approaches to improve the security of embedded systems will only help improve the state of endpoint security—the most critical gap in today's critical infrastructure.

References

1. National Institute of Standards and Technology, "Computer Security Division 2011 Annual Report." http://csrc.nist.gov/publications/nistir/ir7816/nistir_7816.pdf.

Related Parts		
71M6541D	Energy Meter ICs	Free Samples
71M6541D	Energy Meter ICs	Free Samples
71M6541F	Energy Meter ICs	Free Samples
71M6541F	Energy Meter ICs	Free Samples
71M6541G	Energy Meter ICs	Free Samples
71M6541G	Energy Meter ICs	Free Samples
71M6542F	Energy Meter ICs	Free Samples
71M6542F	Energy Meter ICs	Free Samples
71M6542G	Energy Meter ICs	Free Samples
71M6542G	Energy Meter ICs	Free Samples
71M6543F	Energy Meter ICs	Free Samples
71M6543G	Energy Meter ICs	Free Samples
71M6543GH	Energy Meter ICs	
71M6543H	Energy Meter ICs	
DS3231	Extremely Accurate I²C-Integrated RTC/TCXO/Crystal	Free Samples
DS3231M	±5ppm, I²C Real-Time Clock	Free Samples
DS3232	Extremely Accurate I²C RTC with Integrated Crystal and SRAM	Free Samples
DS3232M	±5ppm, I²C Real-Time Clock with SRAM	Free Samples
MAX13256	36V H-Bridge Transformer Driver for Isolated Supplies	
MAX2991	Power-Line Communications (PLC) Integrated Analog Front-End Transceiver	Free Samples
MAX2992	G3-PLC MAC/PHY Powerline Transceiver	Free Samples
MAX71020	Single-Chip Electricity Meter AFE	Free Samples

More Information
For Technical Support: http://www.maximintegrated.com/support
For Samples: http://www.maximintegrated.com/samples
Other Questions and Comments: http://www.maximintegrated.com/contact

Application Note 5445: http://www.maximintegrated.com/an5445
TUTORIAL 5445, AN5445, AN 5445, APP5445, Appnote5445, Appnote 5445