Leveraging Ethernet Card Vulnerabilities in Field Devices

Daniel Peck, Dale Peterson
Digital Bond, Inc.
Fort Lauderdale, Florida
{peck, peterson}@digitalbond.com

Abstract: Field devices essential for the monitoring and control in DCS and SCADA systems are increasingly being deployed with Ethernet cards to connect these devices to local and wide area IP networks. Many of the Ethernet cards have their own CPU, memory, operating system and applications. Field device vendors are also providing the capability to upgrade or replace the firmware in these Ethernet cards. Unfortunately in most cases there is no effective security on the firmware upload to the field device Ethernet cards.

In this paper the authors demonstrate how using commonly available tools an attacker can learn how firmware is loaded into two different field device Ethernet cards, write his own malicious firmware, and load that malicious firmware into the field device Ethernet cards. After this proof of concept malicious firmware load, the authors discuss different classes of attacks that could be launched against the process being controlled, other field devices and other systems on the control system network.

The paper concludes with a brief discussion and examples of vulnerabilities in common management services, such as HTTP, FTP and SNMP, found on many field device Ethernet cards.

Keywords: Unauthenticated Firmware Upload, PAC, PLC, RTU, Controller, Field Device, Field Device Worm, Field Device Ethernet Card

1 Introduction

Field devices are critical components of control systems. They communicate with sensors and actuators to monitor and control a process. The increasing computing power and capabilities in field devices allows asset owners to push more intelligence to these field devices, and they become much more than communication devices between an HMI or control center and the sensors and actuators on the plant floor or field site. Common terms for field devices include Programmable Logic Controllers (PLC), Programmable Automation Controllers (PAC), Remote Terminal Units (RTU) and Intelligent Electronic Devices (IED).

As with most control system components, field devices were designed for a closed, trusted network with little thought for security. If an attacker can ping the typical field device, and has a client or HMI to communicate with the field device, he can send any

read, write, diagnostic or configuration command to the field device. This is due to a lack of source or data authentication in most control system protocols. Some examples of control system protocol based attacks include:

- Writing new values to set points, thereby changing the process
- Rebooting the field device continuously to affect availability
- Setting the field device to listen only mode so it will not respond to requests

Digital Bond has developed network Intrusion Detection System (IDS) signatures that have been integrated into most commercial IDS sensors to identify these attacks for Modbus TCP, DNP3, and ICCP [1].

A similar lack of security features is often found in non-control system protocols such as Telnet, FTP, HTTP and SNMP that are used to manage a field device. Some field devices support one or more user accounts for login, but even this feature is rarely used.

Beyond demonstrating the ease in exploiting an insecure control system or management protocol, most field device security analysis and testing to date has involved fuzz testing the field device's protocol stack [2]. Unfortunately many field devices have failed even minor fuzz testing of Ethernet, IP and TCP/UDP protocols. Some have even failed when legitimate, properly formatted broadcast and multi-cast packets are sent to the field device. Failure most often is a crash of the Ethernet interface or the entire field device and requires a reboot to resume normal operations.

Given this extremely vulnerable state of most field devices, why is further testing or analysis of exploits warranted? For at least two reasons:

1. A knowledgeable attacker could stage an attack for a time and purpose with maximum impact, and this could be done well after the compromise. This paper will discuss some potentially devastating scenarios.
2. A compromised field device can be used as an attack platform for other elements in the network. There are tens or hundreds of field devices in control systems that could be used for individual or coordinated attacks. Imagine a distributed denial of service (DDOS) from the field devices on all servers and workstations in the control center. A field device is also less likely to be analyzed for its security posture as compared to an HMI running Windows or a server running Unix. This is analogous to the use of compromised printers as an attack platform in a corporate network.

The easiest point of field device access for a cyber attacker is the Ethernet card. Many of these cards have their own CPU, RAM and operating system. The computers on these cards can be attacked and often exploited. Once exploited, they can be used to attack the CPU and other cards on the field device as well as other devices on the control system LAN and WAN.

This paper describes different classes of attacks on field device Ethernet cards, sample vulnerabilities found in numerous field devices and corresponding exploits, and the potential consequences of these exploits assuming a knowledgeable attacker.

2 Rogue Firmware Load

Many field devices allow the firmware to be updated over the network. This is also often true of field device Ethernet cards with their own computing resources. A remote firmware update can be very useful, especially in a geographically dispersed SCADA network, for recovery from system corruption, patching newly discovered problems or updating the field device software with new features. It also can be very useful for an attacker if it is not properly authenticating the source and data being uploaded prior to accepting and loading this new firmware.

In the multiple field devices that Digital Bond has in its lab, not one of these field devices authenticates the source or data prior to accepting and loading the firmware. If it is possible for a legitimate user to upload new firmware over the network, it is possible for an attacker, with some work at the assembly code level and logical access to the field device, to upload his own firmware.

In Section 2, the authors will describe the steps required to load arbitrary firmware on two different systems and some possible ways an attacker might use this capability beyond simply attacking the selected field device. It is important to note that this problem is widespread, and goes much beyond these two systems. Any vendor or asset owner that has field devices that allow unauthenticated or weak authentication firmware uploads should be concerned and contact their vendor for guidance and a solution.

2.1 Systems Described

Our test devices for this research were the Rockwell 1756 ENBT Ethernet module and the Koyo H4-ECOM100 Ethernet module, see Figure 1. While similar in functionality these devices were built on very different platforms.

The 1756 ENBT module is a PowerPC device running a VxWorks 5.5 operating system, intended to be managed primarily by the RSLogix family of products. Services available on the device include EtherNet/IP, HTTP, and SNMP. FTP is also available with a little more work, as we'll discuss later.

The H4-ECOM100 module is a device based on a Nios Altera Field Programmable Gate Array (FPGA) chipset. The services available on this device include HTTP and MODBUS/TCP.

While the web interface on the 1756 ENBT is intended almost exclusively for displaying diagnostic information, the H4-ECOM100 web server is used extensively for configuration and maintenance activities.

Figure 1 – Koyo H4-ECOM100 and Rockwell Automation 1756 ENBT Modules

2.2 Learning and Exploiting the Load

We began the process by obtaining several sets of firmware images from the Rockwell vendor site. Several versions were downloaded to allow for comparison, to help in identifying static fields, and various "special" locations in the firmware. Our first step in the analysis was to examine the binaries themselves. In this analysis we find several different segments, some with readable text, that appear to be some of the web pages that the device serves. These appear to exist in a CISF FTL partition, which is similar enough to FAT16. These will be investigated later, but for now we will move on and continue looking for the actual code that gets executed, and being that we are at a bit of a dead end here we'll start to rely of assumptions and experience.

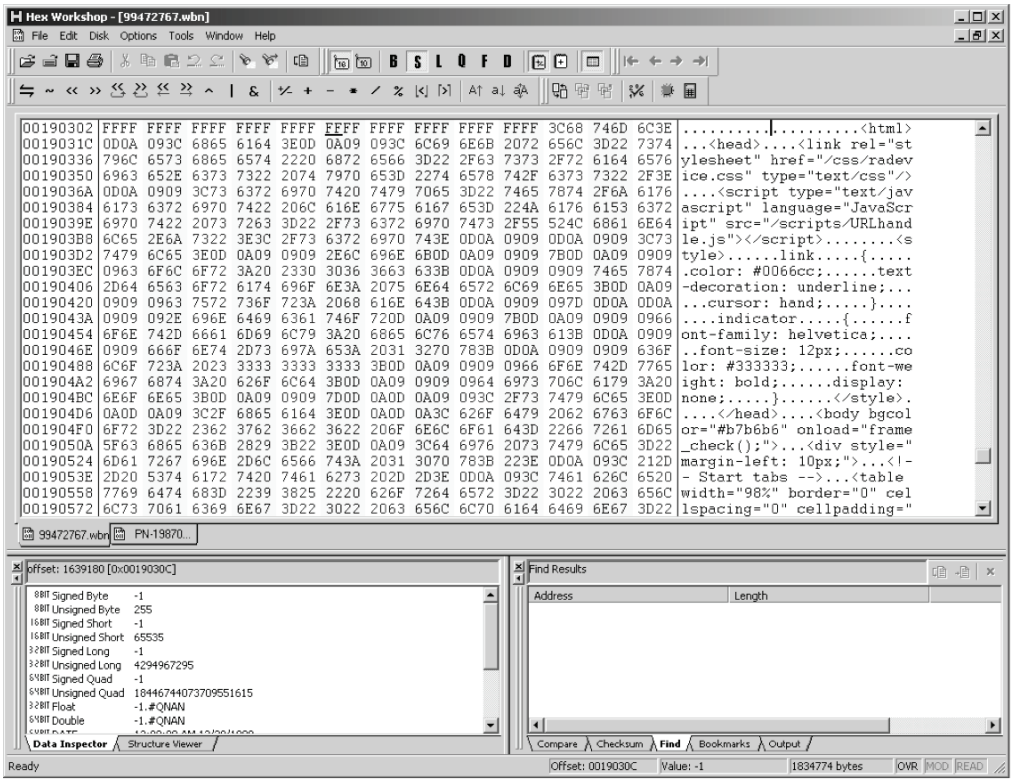

Figure 2 - HTML Visible in Firmware

Using a tool developed by Matasano named deezee, we scan the firmware image looking for zlib compressed sections, and we get a hit. Pulling this into our hex editor and examine some strings, we find what appears to be some symbol names, such as AB_Exit at 0x001ef0b4 and AB_Init at 0x001ef0ac. These seem interesting but we have to find the offset table to be able to do anything very useful. Looking through the file we notice a very regular ten-byte pattern beginning at 0x001fbdf8.

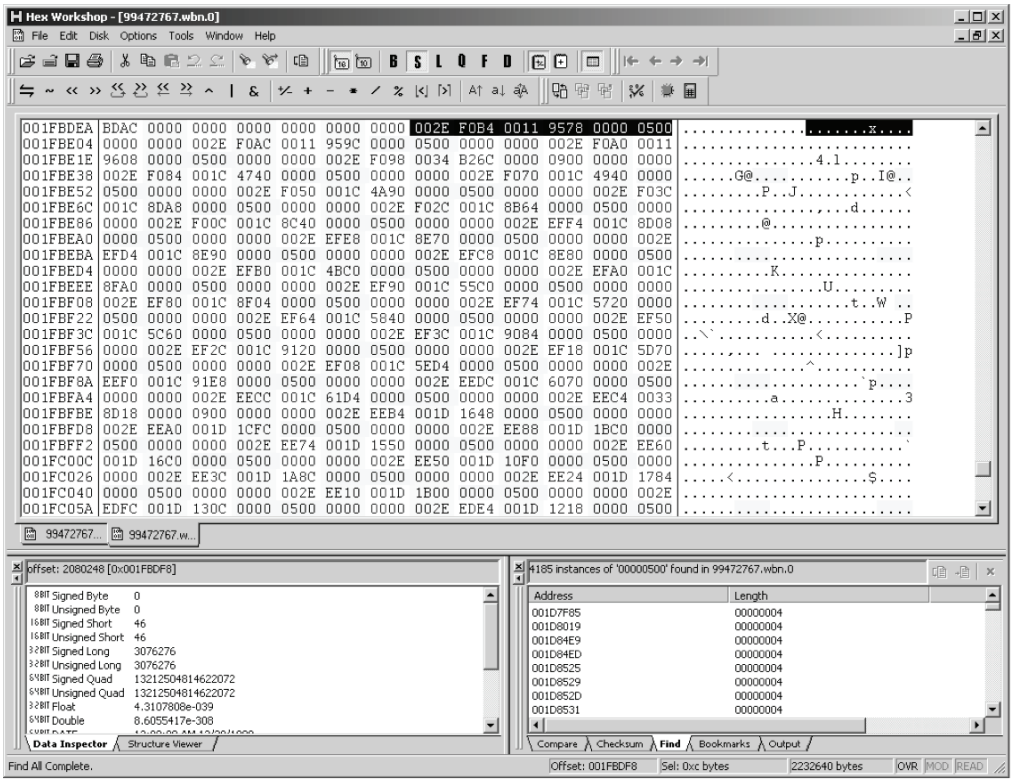

Figure 3 – Ten Byte Pattern: 0x 0000 0500 0000 0000 002E

The first address looks a lot like the address of AB_Exit, and the second one looks a lot like the address of AB_Init. Some quick subtraction and we find that our load address is 0x00100000, and it looks like the next entry in that repeating pattern is the address of the code itself. We're almost ready to begin some real analysis, but first we have to know what kind of processor this code was meant to run on, our first guess was that it was on an ARM chip, but looking at the firmware dump we see r0, sp, r2, and quite a few others, we have our answer, a Power PC processor.

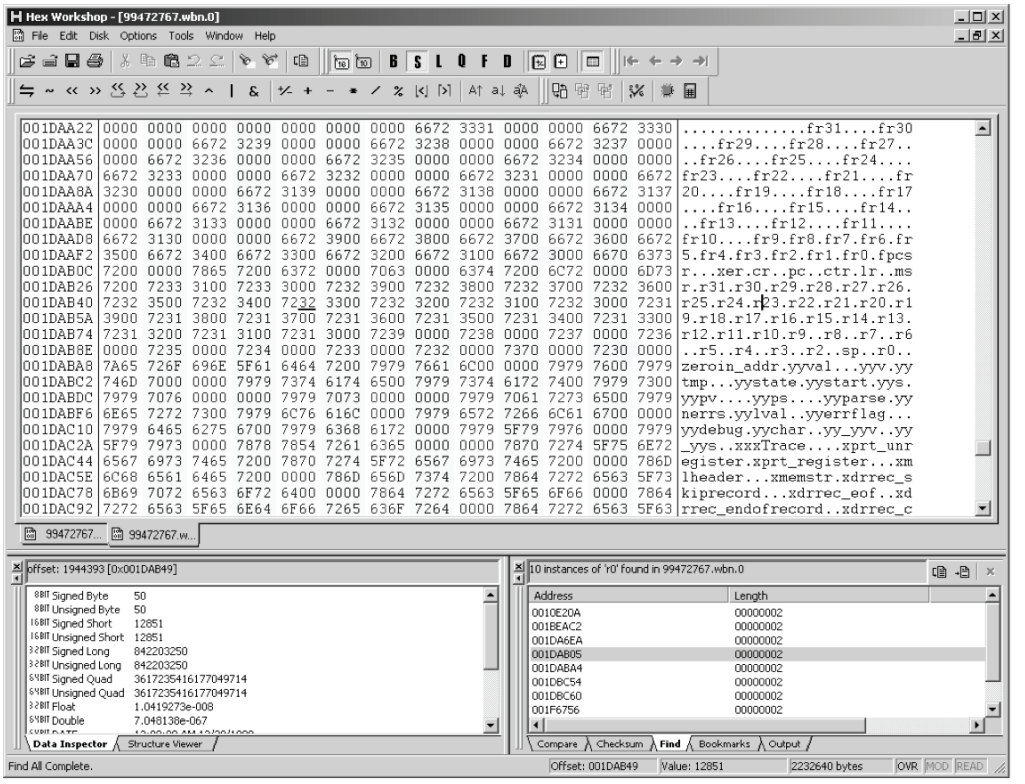

Figure 4 – Identifying the Power PC Processor

We have loaded the file into the IDA PRO disassembler and debugger program and rebased the image. Now after a little scripting to add the rest of the symbols into our firmware image we can get down to business on understanding the rest of the firmware file structure and any validation routines it might have in place. The function nv_RamValidateChecksumsWriteFlash that makes two calls to ffs_CalcChecksum looks interesting, and here is where we begin to really understand the inner workings of the device.

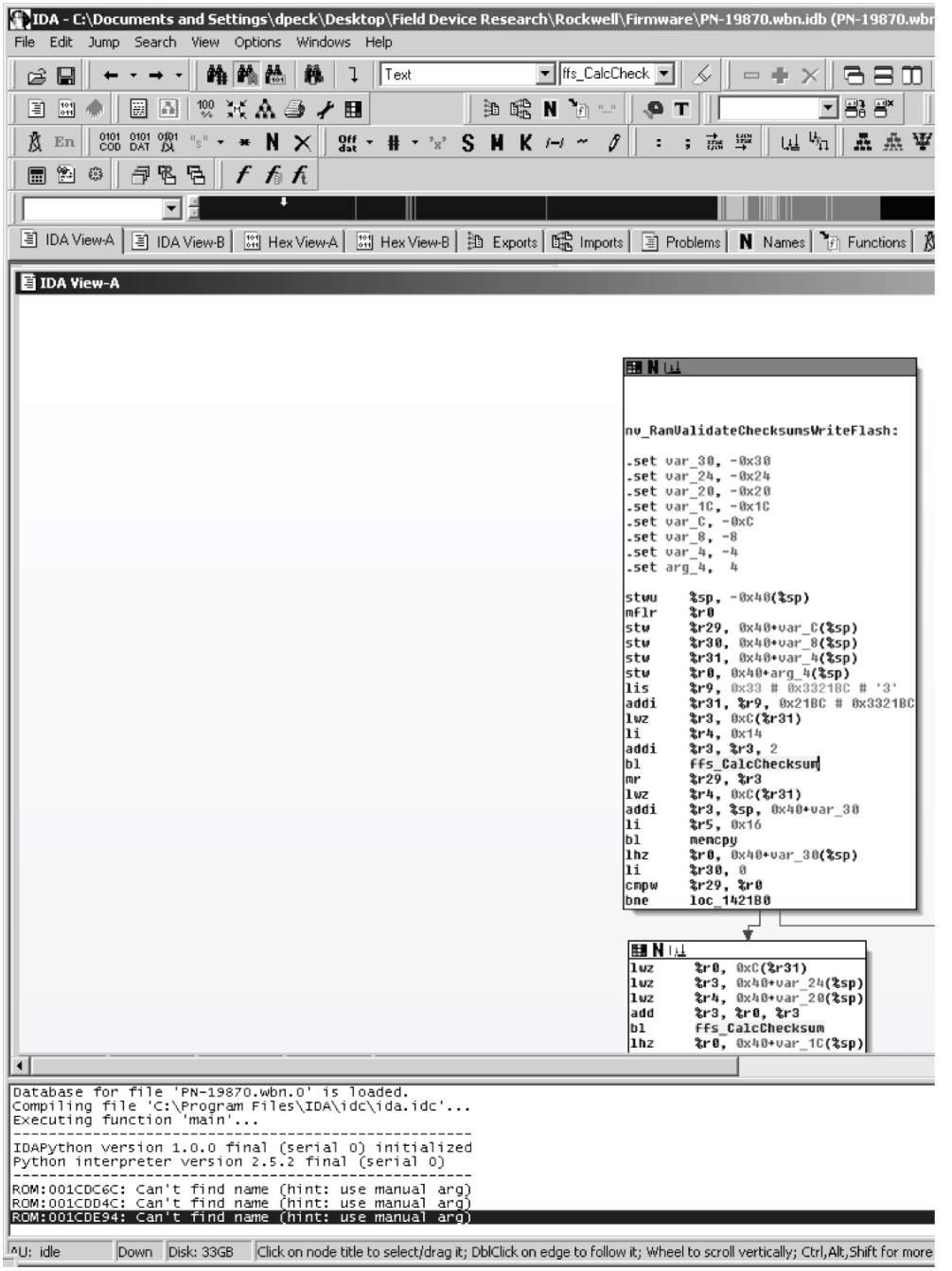

Figure 5 – nv_RamValidateChecksumsWriteFlash Function

Reverse engineering the disassembly we find 2 checksums. A 2 byte sum or 2 byte halfwords of the header, and a 2 byte sum of 2 byte halfwords of the remainder of the

file. So we have the header of the file, 2 byte header checksum at 0x0, 4 bytes data offset at 0xc, 4 byte data size at 0x10, and 2 byte data checksum at 0x14. With this knowledge we are able to change the external "wrapper" portion of the firmware as we see fit, this include the web pages stored there, and upload it with the ControlFlash program provided by the vendor so customers can load new firmware.

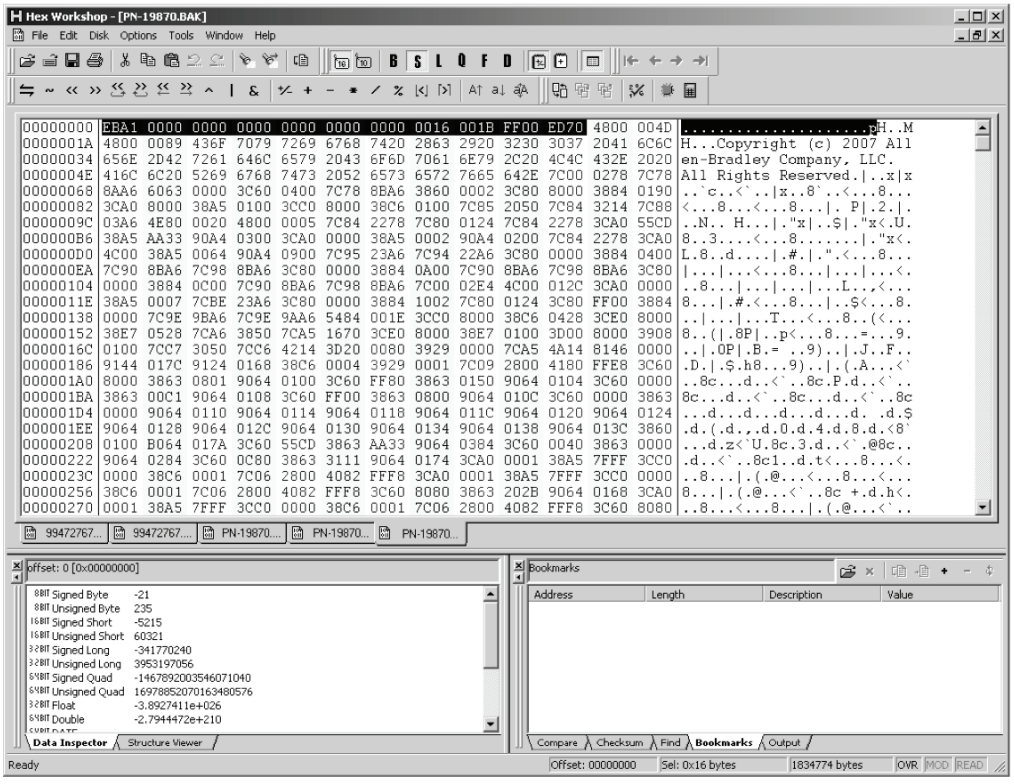

Figure 6 – Firmware Header and Checksum

The authors developed proof of concept shellcode that makes use of the "PING" function already in the 1756 ENBT module. This is much like making use of a system call on a Windows operating system. Being careful not to disrupt the current functions of the device we have embedded this code into a portion of the FTP server code. The 1756 ENBT module now pings a specified IP address while continuing to perform its designed purpose for the Logix PAC.

This ping is meant only as a demonstration that an attacker with logical access could load and run arbitrary firmware on the 1756 ENBT module due to lack of any source or data authentication. Since the card has significant unused memory and processing power, an attacker could load and run much more substantial and dangerous code on the module within a reasonable amount of development time. The vendor's

ControlFlash program was used to make the upload easier, but we could have developed our own flash upload utility if necessary.

The process of learning and exploiting the load with Koyo/AutomationDirect H4-ECOM100 module was similar. In fact, the process was similar for all of the field devices in our lab that had a firmware upload lacking in source and data authentication.

With firmware images and utilities found on AutomationDirect.com website and the commonly used packet capture utility, Wireshark, we find that the firmware is upgraded using a protocol that Wireshark doesn't know about. We need more information on this upload protocol. One thing that is common in all the messages is they begin with "HAP".

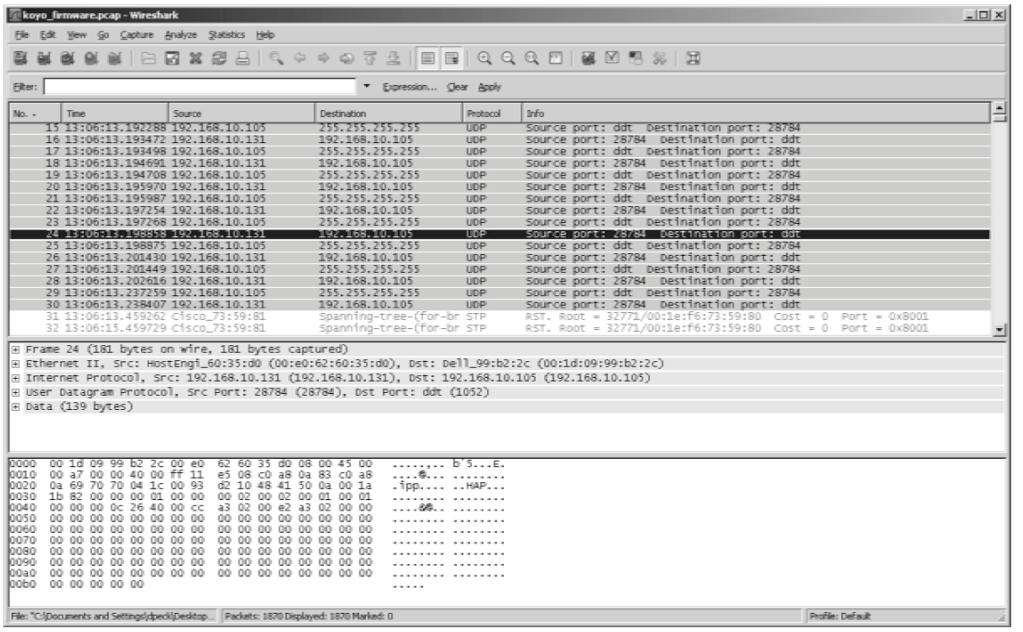

Figure 7 – Identifying "HAP" in Messages

We could not find information on this protocol on the Koyo or Automation Direct websites. However, an Internet search revealed a document that described the protocol and ECOM interface in some detail [3], see Figure 8.

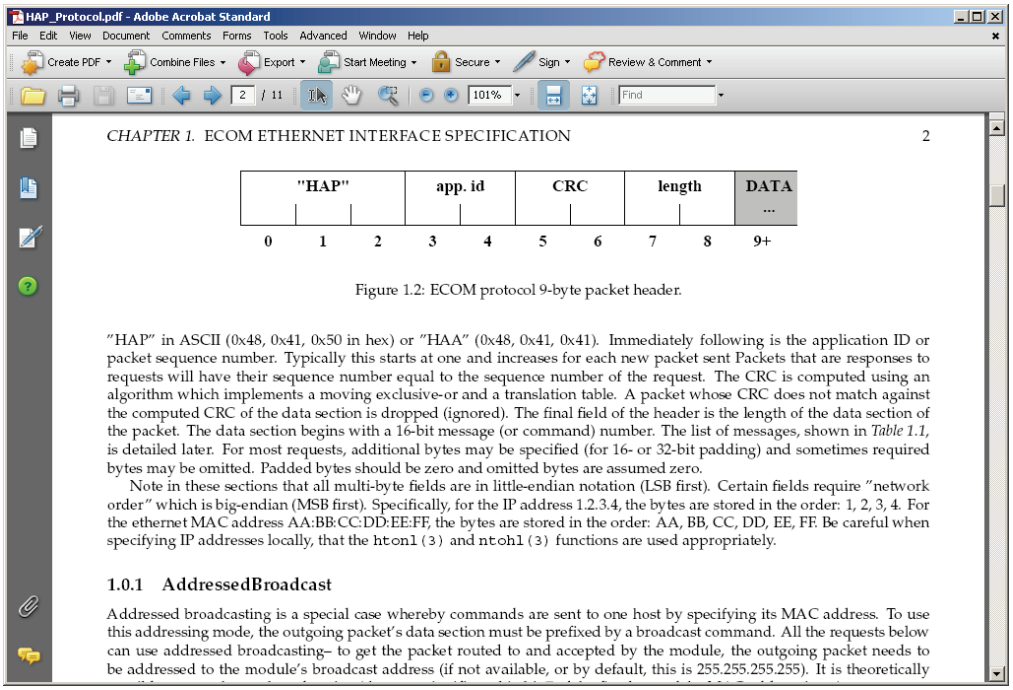

Figure 8 – ECOM Protocol Packet Header

Examining the firmware images we get some clues about the internals of the module. "Nios OS" is a good bit of information that its run on the NIOS embedded FPGA processor from Altera. Looking further into the firmware upload procedure we find that the upgrade program, netedit, seems to work over UDP broadcast protocol. However with further investigation we find that this is a weakness of the program and not a weakness of the protocol itself.

We find the same lack of source and data authentication on firmware uploads in the Koyo system as found in the Rockwell Automation PAC. In fact the Koyo PLC firmware upload appears to even lack a checksum. As a proof of concept for malicious firmware upload, we uploaded modified web pages that are available from the ECOM module.

While it is bad enough that these systems can be so easily compromised, an even greater concern is as long as the compromised field device is still functioning as expected the asset owner could be completely oblivious to the fact that many or even all of the field devices in the control system have been compromised.

Extrapolating out from this and working with other devices we have found similar results. Over the network firmware uploads are, in general, not authenticated or otherwise protected in any meaningful way. Typically a checksum is used to authenticate

the integrity from a non-malicious, load integrity standpoint. This checksum offers little protection from a motivated and moderatedly skilled attacker.

Many field devices use different and more obscure processors than what an average attacker would see in the PC-dominated corporate world. While the obscurity of a processor can make the potential attackers work harder to investigate, develop and implement a malicious firmware upload to a field device, it is not a safeguard. Much work has been done on similar platforms by the gray and black hat community, such as home switches and routers, and that knowledge will transfer over easily for future field device hackers.

Even this security by obscurity is becoming less of a deterrent. Many field devices are moving to Intel x86 chips as well as more common operating systems such as embedded XP or embedded Linux. While these are not necessarily easier to develop firmware uploads for, there are many more attackers and tools available to help an attacker exploit these systems. It will take less specialized knowledge and less expensive tools to attack a field device on an x86 platform as compared to a field device with an ARM.

2.3 Attacking the PLC

As mentioned in the Introduction section, logical access to most field devices allows an attacker to read and write to the device. So what benefit is there in uploading new firmware over directly monitoring the sensors and controlling the actuators and other instruments in real time? The answer to this question is limited only by imagination. Here are a few scenarios to illustrate the benefit to an attacker.

2.3.1 Delayed Attack

An attacker may want to stage an attack in advance and have it ready to launch at a time that is particularly damaging. Imagine an attack on a region's power system that caused an outage at 11:58 on New Years Eve, on election day, or at the start of a big event like the Olympics. The malicious firmware upload could allow normal operations until a certain time and date and then launch a scheduled attack.

A delayed attack scenario could be coordinated between multiple field devices that had been exploited with a malicious firmware upload that set up a cron job to run at a specific time. These could be field devices across a single control system or even field devices in multiple control systems.

Another benefit of the delayed attack is it would be substantially harder for an asset owner to determine what caused the outage or incident. A review of recent traffic to and from the field device, if this were even available, would not show the firmware upload. Depending on the nature of the attack, there is a good chance that the asset owner would not even realize a cyber attack took place. It could be made to look like a hardware failure, communications error or some other non-malicious cause.

Finally, a delayed attack can address the situation of an attacker losing access to the field device. Imagine a malicious firmware upload that required a message from the attacker on a monthly basis or an attack would be launched. This attacker may want to have the capability to bring down part of the critical infrastructure or a particular organization he

holds a grudge against, but chooses to reserve that capability for an unspecified future need. If the asset owner has improved perimeter security and the attacker no longer has access to field devices he owned. His capability to schedule an attack is lost, but the attack would launch within a month.

2.3.2 Brick the Ethernet Card

The malicious firmware could be programmed for an Ethernet card failure that required a return to factory by removing the ability to reload new firmware and removing the proper firmware capabilities - - turning the card into essential dead weight or a brick.

A single Ethernet card failure is unlikely to cause much of a problem for an organization mindful of redundancy and recovery requirements in a control system. Spares would be available. However it will take time to replace the card, especially in a geographically dispersed SCADA system. If this attack was made on Ethernet cards in a large number of controllers this could exhaust the spares and cause availability problems to travel and replace numerous simultaneous Ethernet card failures.

2.4 "Random" Attack or Failures

An attacker's goal may not be to cause an outage that affects a large number of people. The attacker may not want to shut down power to a region or affect the ability of a large city to get gasoline or water. This level of impact may be more than the attacker desires and the consequences of getting caught performing such an attack is likely to prevent many potential attackers as discussed in the Langner and Singer paper [4].

For an attacker, perhaps a disgruntled insider, who just wants to make difficulties for the operators and engineers a "random" attack or failure mode may be more interesting. The attacker could load malicious firmware into multiple field devices and have a pseudo random number generator select the attack and the attack time. Imagine the case where the malicious software could temporarily make the device unavailable, brick the card, send false data back to the control center, make process changes or simply reboot the system multiple times. The random number could select the attack from the list, a second number could select the attack start time and date, and a third random number could select the attack duration.

Troubleshooting these problems could be made very difficult if the field device returned the expected information, such as version number and checksum, during diagnostic tests. The malicious software could also prevent the loading of new firmware onto the Ethernet card without a special code or authentication mechanism. The malicious firmware could be programmed to issue a response that indicated the asset owner had successfully loaded new firmware, perhaps even modifying the firmware version number returned when requested, while in fact the malicious firmware remained intact.

2.5 A Field Device Worm

Each field device has its own set of software and methods to program, configure and manage the system. Once an asset owner has purchased and learned to configure and program a field device, they are likely to purchase additional field devices of the same

model. While it is not unusual for an asset owner to have a variety of field devices, it is typical that they will have tens or hundreds of each vendor model.

Since the unauthenticated firmware upload allows an attacker to load malicious code on the field device, the attacker may be interested in identifying all field devices of the same model and compromise all of them. This is achievable with a field device worm designed to take advantage of the lack of source and data authentication of a field device firmware upload.

As an example, imagine an attacker has identified an Allen Bradley field device on the network. Since these devices typically use the EtherNet/IP protocol on TCP/44818 they are easy identify on a network with a port scan of that port. A worm can be written that would:

1. Port scan to identify other systems with EtherNet/IP ports that are open
2. Attempt to load the attacker's firmware on all identified EtherNet/IP systems
3. Report back to the master after the system is compromised
4. Standby for future instructions or take preprogrammed actions

An attacker would be in a similar situation to attackers that have control of an army of bots except they would be on a control system network.

2.6 Attack Other Cards on the Field Device

While the Ethernet card is the most accessible to an attacker or malware, it is not the most important card in a field device. In fact many control systems today use the field device Ethernet card for maintenance and still rely on serial communication cards for SCADA system monitoring and control. This is likely to be less common over time as more asset owners move to an IP WAN for their SCADA systems.

The most important card in field devices is typically the CPU card. If this card is not functioning properly, the entire field device will not function. Some CPU cards also support a remote firmware update through the Ethernet card. The CPU card may have a different computing architecture, different load utility and other differences from the Ethernet card, but it also may be identical.

Perhaps the simplest attack on the CPU card is a denial of service attack that could brick this CPU card. Imagine a case where the Ethernet card is compromised with rogue firmware that periodically, or perhaps pseudo randomly, uploaded to firmware to the CPU card that bricked the CPU card. The asset owner would replace the CPU card, but the underlying problem would remain in the Ethernet card.

2.7 Attack Other Systems

The attacks discussed in Section 2 to this point have focused on attacking the field devices to affect the monitoring and control of the process. However, one or more compromised systems on a control center LAN or WAN provide an attack platform to go after other systems on the SCADA or DCS, such as workstations or servers.

Field devices are usually not considered as potential attack platforms. They are not patched or upgraded frequently; typical asset owners do not worry about security patches, anti-virus and security configuration issues with field devices nearly as much as they do for a Windows HMI; they are deployed and often not touched for years. In many ways, these field devices are similar to printers on a corporate network. They are computing platforms spread throughout the network that are treated like appliances rather than computing devices.

One of the common findings in control system security assessments is missing security patches. It is not unusual to find Windows systems missing patches for many years. An asset owner who is diligently patching the Windows OS every month may be missing patches in the Oracle database, Java Runtime Environment, Apache web server, or other non-Windows software. An attacker could upload malicious firmware on a field device to identify and exploit missing security patches on workstations and servers on the SCADA and DCS.

An example of this malicious software could be in the form of a worm that would seek out other similar devices to exploit and load itself onto, but also have a set of attacks for HMI's, real time servers, and historians. This behavior could be uncontrolled as many early worms were, and simply exist for destructiveness, or it could be controlled by an authority or "botmaster" directing the activities of the compromised devices and systems.

2.8 Solutions

The solutions to the malicious firmware upload over the network are straightforward. The low-tech solution is to not allow over the network firmware upload. Many controllers have physical keys that can restrict the ability to load new firmware over the network or locally. This is an effective control, but it can have a significant impact on the ability to manage a system of field devices, especially in a geographically dispersed SCADA system.

The high-tech solution is to add source and data authentication to the firmware upload process. The most straightforward way to do this is with a public key certificate infrastructure and a digital signature on the firmware upload. The field device could verify the certificate and the digital signature to confirm the firmware was from an authorized entity and had not been changed in transit. This would require public key implementations on the Ethernet cards, but efficient and secure algorithms are available using elliptic curve technology [5]. Since firmware updates are rare events, the processing time required to verify the certificate and digital signature is not an important factor.

3 Management Application Vulnerabilities

Field device vendors have been trying to find ways to make field device management easier and provide more status information to network management and monitoring systems. It is common to find field devices with web servers, FTP servers, the SNMP

service, and the Telnet service. Each of these servers or services are well known and commonly attacked by the hacking community.

Sometimes the operating system on the Ethernet card provides these servers and services; in other cases the field device vendor develops their own server and service; and in many cases the field device vendor buys a low cost or obtains a free management server or service. The commonality in all these cases is these management servers and services are rarely designed to be full featured, maintained and secured. They often fall into the category of "quick and dirty".

An alert asset owner will know they should patch a Microsoft IIS or Apache web server by monitoring one or more cyber security bug fix lists. But that same asset owner is unlikely to even monitor the vendor of the web server on a field device for vulnerabilities and security patches.

In a review of the field devices installed in Digital Bond's lab, we found that many of the common vulnerabilities in similar IT applications in the applications integrated into the field device. This section provides examples of common vulnerabilities in these management services found in a small sample of field devices.

3.1 Web Server

Several different forms of cross site scripting (XSS) were found on the sample field devices. These vulnerabilities can be leveraged to exploit vulnerabilities in client side software, such as Internet Explorer. We found both persistent and non persistent XSS, some that would require social engineering to encourage a user to follow a link, and others that could wait for a user to visit the management interface before the attack was launched, weeks or even months later.

Flaws were also found which enabled unauthenticated users to view the ASP source code of a page and access configuration files without a password by appending a trailing "/" to the URL.

3.1.1 Example

The Koyo H4-ECOM100 module provides a good demonstration of these types of vulnerabilities. The management interface requires no authentication to access the device and the user editable fields are not properly sanitized before being served back to the user.

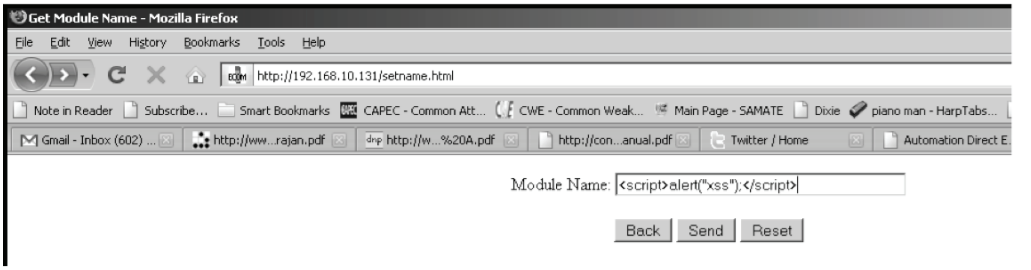

Figure 9 – Part 1 of XSS Example on Koyo H4-ECOM100 Module

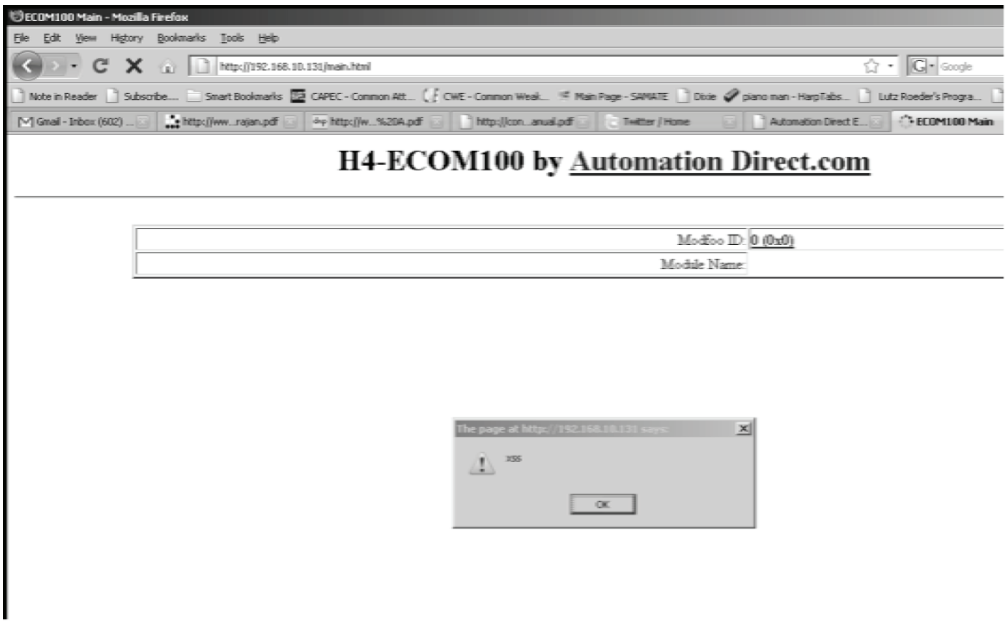

Figure 10 – Part 2 of XSS Example on Koyo H4-ECOM100 Module

3.2 FTP Server

During the disassembly of the Rockwell Automation device code a FTP server was discovered, however the device does not have this FTP server functionality enabled. By manipulating the firmware and simply replacing a 0 with a 1 in an XML configuration file that was clearly visible in the firmware we were able to activate and interact with the FTP service. So we have not only expanded the functionality of the device, but also expanded the attack surface. This functionality was provided by the underlying operating system and the vendor may not have been aware that the FTP server code was even on the Ethernet card.

Initially we are simply presented with a login when connecting to the FTP service, but with the Administrator username and blank password, discovered through the assessment of web server, we are able to login to the device. Not only did this open up many other FTP commands to attack, commands in which vulnerabilities were found, but also now provides a location to store information for later use in exploitation or to store damaging or illicit data.

3.3 SNMP

The Rockwell Automation 1756 ENBT module also runs a SNMP service that allows large amounts of information to unauthenticated sources that could be leveraged in further attacks against the system, such as services and types of device. The lack of authentication is just further evidence that these field devices were designed with the

expectation they would be in a secure environment and only accessible by authorized users.

4 Operating System Vulnerabilities

While beyond the scope of this paper, the authors feel it is important to recognize that the underlying field device operating system is a large attack surface that has received little scrutiny by security researchers or the hacking community to date. Some real time and embedded operating systems are used in numerous field devices that monitor and control critical infrastructures. These are likely to garner more attention by directed attackers in the future and are worthy of analysis by security researchers.

5 Conclusion

The Ethernet cards on field devices present an attractive target for attackers because of their distribution throughout a modern control system, lack of security features and lack of security attention paid to these cards. In this paper, the authors have demonstrated how a lack of authentication in the firmware upload allows an attacker to load malicious code into two field device Ethernet cards. The impact of this malicious code on the process, field device and other components on the control system is limited only by the creativity and imagination of an attacker.

Control system asset owners should be determining if firmware can be uploaded into their field device Ethernet cards. And if the answer is yes, is this source and integrity of this firmware upload authenticated prior to loading on the card? If firmware authentication is not required, the vendor should be contacted and strongly encouraged to address this vulnerability. Asset owners should also consider adding authentication of firmware uploads to any future buying criteria.

Finally, control system asset owners and vendors should not ignore the security of the management services commonly found on Ethernet cards such as HTTP, FTP, Telnet and SNMP.

About the Authors – Daniel Peck is an Offensive Security Researcher at Digital Bond, and a key member of the U.S. Government funded Portaledge and Quickdraw research projects. Prior to joining Digital Bond, Mr. Peck was a security researcher at SecureWorks and a frequent presenter at leading security events such as BlackHat, Defcon and the Pentagon Security Forum.

Dale Peterson is a founder of Digital Bond, Inc. and has led their Control System Security Consulting and Research Practice since 2000. He began his career as an award winning Cryptanalyst at the National Security Agency and has more than twenty years experience in information security.

References

[1] Digital Bond SCADApedia, http://www.digitalbond.com/wiki/index.php/SCADA_IDS_Signatures.

[2] E. Byres, D. Hoffman and N. Kube, "On Shaky Ground – A Study of Security Vulnerabilities in Control Protocols", American Nuclear Society, November 2006.

[3] ECOM Ethernet Interface Specification, Host Engineering.

[4] R. Lagner and B. Singer, "SCADA Threat Modeling Using Attack Scenarios", 2008 S4 Proceedings, Digital Bond Press, 2008.

[5] R. Lambert, "ECC and SCADA Key Management", 2007 S4 Proceedings, Digital Bond Press, 2007.