Securing EtherNet/IP Control Systems using Deep Packet Inspection Firewall Technology

Eric Byres, Chief Technology Officer
Erik Schweigert, Lead Embedded Systems Developer
Michael Thomas, Lead Systems Developer
Tofino Security, a Belden Brand

Keywords: Deep Packet Inspection, EtherNet/IP, Common Industrial Protocol, CIP, CIP Object, CIP Service, Firewall, Access Control List, ACL, Security, GUI, User Interface, Usability, SCADA, Industrial Control Systems, ICS

ABSTRACT

Next Generation Firewalls with Deep Packet Inspection (DPI) capabilities are now mainstream products for IT protocols. Unfortunately, designers and operators of industrial control systems (ICS) have not had access to these advanced technologies to protect their critical communications that involved protocols such as EtherNet/IP™. This is a serious problem. Mission critical control systems need DPI technology even more than IT systems do. This paper discusses the creation of a DPI firewall for EtherNet/IP and Common Industrial Protocol (CIP™), and the lessons learned in the process. It explores why DPI is needed for control security, what is available today, and the challenges going forward. This paper looks at the technical issues in creating an EtherNet/IP DPI firewall that is useable and the solutions that are emerging. The paper closes with a case history of the use of an EtherNet/IP DPI firewall.

I. INTRODUCTION

The world's manufacturing, energy, and transportation infrastructures are currently facing a serious security crisis. These critical systems are largely based on legacy SCADA and Industrial Control System (ICS) products and protocols. Many of these products are decades old and were never designed with security in mind.

Yet industry has also embraced new network technologies like Ethernet and TCP/IP for ICS. This has enabled companies to operate cost effectively and implement more agile business practices through instant access to data throughout the organization, including the plant floor. While this interlinking improves efficiency, it also significantly increases the exposure of these control systems to external forces, such as worms, viruses, and hackers.

Given the 20 year life cycle that is common for industrial systems, it will be many years before more secure ICS and SCADA devices and protocols are in widespread use. This leaves millions of legacy control systems open to attack from even the most inexperienced hacker. If a hacker or worm can gain access to any control system, it can exploit the protocol to disable or destroy most industrial controllers.

There are a number of possible solutions to this issue, including encryption and authentication technologies and modifications to the ODVA standards. This paper looks at one specific security technology, namely Deep Packet Inspection (DPI). This advanced filtering technology promises fine-grained control of ICS network traffic, including EtherNet/IP, beyond what is typically found in the IT firewall.

This paper is structured in the following way: Chapter I provides a general introduction to the topic. Chapter II provides an overview of the EtherNet/IP and CIP protocols. Chapter III outlines some of the threats to systems exposed through the use of EtherNet/IP. Chapter IV discusses possible methods of securing EtherNet/IP-based systems. Chapter V provides more detail to one of the possible solutions discussed in Chapter IV, namely DPI technology. Chapter VI gives an overview of how EtherNet/IP DPI was implemented in an industrial firewall. Chapter VII discusses the details of creating a user interface for configuring an EtherNet/IP DPI firewall. Chapter VIII describes case history of the use of an EtherNet/IP firewall to ensure the safety of turbine systems in the oil and gas industry. Chapter IX concludes the paper.

II. THE COMMON INDUSTRIAL PROTOCOL

The Common Industrial Protocol (CIP) is an ever expanding and revised protocol to meet the needs of the dynamic industrial world. CIP can be used above varying network protocols including DeviceNet™, ControlNet™, EtherNet/IP, and CompoNet™. EtherNet/IP specifically uses the standard Ethernet layer of TCP/IP and is widely adopted in the industrial automation arena. ODVA is the caretaker of the protocol and diligently manages and distributes the specifications for CIP Networks in a common structure to help ensure consistency and accuracy in the management of these specifications. [1]

A. General Overview

CIP is based on abstract object modeling that allows it to represent real world entities. A CIP node is identified as a set of objects, where an object is the realization of a physical piece of a particular element of a product and is mapped to that logical object on a per product basis.

A class is a set of objects that all represent the same kind of system component. An object instance is the actual representation of a particular object within a class. Each instance of a class has the same set of attributes, but has its own particular set of attribute values. For example, the class may be Analog Input Point objects and an instance might be a specific analog input for a level transmitter. Example attributes for this instance include: a status attribute used to indicate the presence of a fault or alarm, a voltage input operating range of the input source, a 32-bit serial number of the channel's owner. [1]

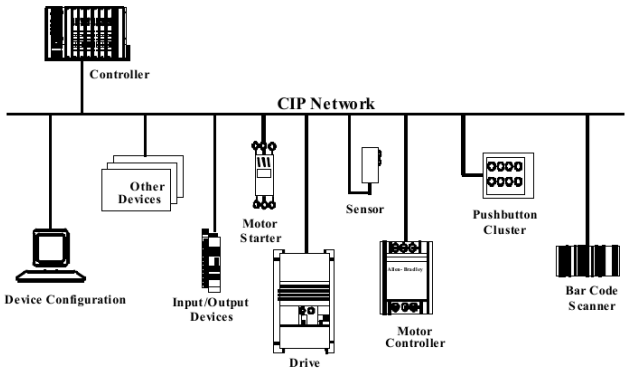


Figure 1: Various unrelated elements all using EtherNet/IP and CIP to communicate. [1]

Each object supports a set of default services and, in some cases, a set of custom services. A service is a function that is supported by the object. For instance, a Get Attribute All service allows a client to request the values of all the attributes for a specific instance of an object.

From a network perspective, CIP is a connection-based system in which a connection is made between multiple end-points. In general, two differing types of communication are used to meet latency requirements. Input/Output (I/O) communications commonly use Class 1 implicit messaging where low jitter and latency are needed. In contrast, Class 3explicit messaging provides a reliable connection based session. Typically, User Datagram Protocol (UDP) is used for the low jitter and latency communications and Transport Control Protocol (TCP) is used for multi-purpose explicit messaging, such as communications between controllers and HMIs.

Explicit messaging uses EtherNet/IP to encapsulate CIP explicit messages. EtherNet/IP's main purpose is to establish connections utilizing standard commands, particularly the Register Session command that must be executed prior to any CIP messaging. After the session is established, various fields in the EtherNet/IP header, such as packet lengths, and types of connection data that relate to the adjacent CIP header are updated.

The CIP layer is more complex, but follows a consistent format. It can map itself to various functions, physical entities, and products, all within a single application layer of an Ethernet frame. The strength of the protocol is its ability to group data points via the object representation. For example, if a PLC has an analog output point, then the HMI can use the CIP Analog Output Point object with its predefined set of actions and attributes to both extract data values and configure the analog output points in the PLC.

III. SECURITY THREATS TO ETHERNET/IP IMPLEMENTATIONS

While companies reap the benefits of new technologies such as EtherNet/IP, many are also discovering the inherent dangers that result from making control systems more accessible to a wider range of users. Linking

corporate systems together to provide access to managers, customers, and suppliers significantly increases the exposure of these systems to external forces such as worms, viruses, and hackers.

To make matters worse, network protocols like EtherNet/IP were never designed with security in mind. If they offer any capability to restrict what users can do over the network, it has been shown both in academic research[2] and in real world vulnerability disclosures[3][4] to be easy to subvert. If a moderately skilled individual is allowed to read data from a controller, then it is very likely they could also shut down or reprogram the controller.

This is not the fault of the standards groups creating the protocols. Like most other industrial protocols, much of the EtherNet/IP and CIP specifications were developed at a time when security of control systems was not a consideration. As a result, mechanisms to ensure the confidentiality, integrity, or availability of messages were not included in the specification. Nor were any features to allow authentication of the devices sending or receiving messages. Byres et al[2] discuss the ramifications of the lack of security mechanisms in ICS protocols in general.

The security impact of this lack of defined security services in EtherNet/IP was demonstrated by Digital Bond's public testing of a ControlLogix PLC in 2012.[5] In their example, an EtherNet/IP session was established between an attacker's computer and the PLC. Then using the CIP TCP/IP object, the attacker was able to modify critical values in the device, such as the IP address, network mask, and DNS name servers. Authentication was not required to do this: only the establishment of a new EtherNet/IP session. The CIP TCP/IP object is a commonly used object and is not malicious in its nature, but it can be exploited by an attacker.

Expanding on this attack technique, it is conceivable an attacker could listen to an active session and add a CIP packet within the sequence to modify a running PLC's IP address, thus severing the legitimate session. The lack of authentication leaves a glaring hole for attack.

Even more worrying is the ability of an attacker to initiate a firmware update using standard CIP messaging, also evidenced by attack #7 by Digital Bond [5] against the ControlLogix PLC. No fault can be found by the specification offering the ability for this to occur, much like Toyota could not be blamed for poor automobile drivers. It begs the question, though: what authentication mechanism could be added to the specification to prevent rogue CIP messages from being sent by any computer on the network? This is an area for future research and standards development.

Sometimes it is not the limitations of the specification but rather implementation-specific problems. These problems often manifest themselves when an EtherNet/IP stack does not properly bounds check for malformed frames. Again this is no fault in the protocol specification, but rather the lack of validation of the client or server application.

There is a perception that malformed frames only occur from a rogue vector, but this is not the only source. For example, a poorly designed HMI can provide an off-by-one error within the EtherNet/IP length field. This would inform the server process that another byte should exist, when in reality the size field is actually wrong. The improper use of length fields can also trick the PLC into thinking a massive data message is on its way when there is none (or vice versa). Without properly ensuring a request packet is of the correct form, vulnerabilities could exist.

IV. METHODS TO SECURE ETHERNET/IP

Unfortunately the security issues that are part of the current EtherNet/IP standards and implementations are likely to remain with us for at least the next decade. Discussions are underway to add security features into the specifications, but these are probably several years away. Even when these changes to the standards are completed, industrial controllers are replaced slowly; their useful lives may be 10, 20 or more years. And the security limitations of the existing SCADA and ICS protocols cannot be addressed through product patches, as their functionality is defined in established standards that take years to change.

Thus it likely will be years before newer, more secure ICS and SCADA protocols and devices are in widespread use. This leaves millions of legacy control devices open to attack from the average hacker. If a hacker or worm can get any control system access, it can exploit the protocol to disable or destroy most industrial controllers.

This means the industry must explore methods to secure these existing protocols and systems, independent of future improvements to the specifications. There are two primary technologies used in the IT world to secure on-the-wire messages. These are encryption and packet filtering. The following sections briefly explore the applicability of both to securing EtherNet/IP messages.

A. Encryption of Data and Virtual Private Networks

One common method of securing communications is to use cryptographic techniques, such as encryption-based tunneling. Usually referred to as Virtual Private Network (VPN) technologies, they are commonly used in the IT world, but are less prevalent in the ICS arena.

VPN technologies create a secure 'tunnel' between two end points over an untrusted network (such as the Internet) by electronically encrypting, authenticating, and validating every message sent between the end points. In other words, VPNs provide three key capabilities:

- Privacy: VPNs encrypt the data passing between the two end points, so that any unauthorized person or device listening to the conversation cannot understand what is being communicated.
- Authentication: VPNs authenticate each end point to the other, so each party in the conversation can be sure that the other party really is who they say they are and is not an imposter pretending to be an authorized party.
- Integrity: VPNs ensure that messages are not modified in transit between the sender and receiver.

While certainly promising in the long term, cryptographic techniques currently suffer from a number of limitations. EtherNet/IP is a time critical protocol and for every packet entering this tunnel there will be overhead incurred to encrypt and decrypt the packet. For devices like PLCs with limited CPU resources, this overhead can result is significant delays. In addition, VPN provides no data validation: if an authorized node on the VPN sends bad data into the tunnel, bad data will appear at the other end.

Probably the most promising use of cryptographic technology is for the authentication of devices, rather than encryption of the data stream. Unfortunately, the issues regarding reliable key or certificate management for embedded control devices like PLCs are still an open research topic.

B. Firewalls and Deep Packet Inspection

An alternative to cryptography-based solutions is to use firewall technologies to filter network traffic. A firewall is a device that monitors and controls traffic flowing in or between networks. It starts by intercepting the traffic passing through it and comparing each message to a predefined set of rules (called Access Control Lists or ACLs). Any messages that do not match the ACLs are prevented from passing through the firewall.

Traditional IT firewalls apply filters at the TCP and IP layers of a message, using ACLs to check three primary fields in a message:

1. The address of the computer sending the message (i.e., the Source IP Address)
2. The address of the computer receiving the message (i.e., the Destination IP Address)
3. The application layer protocol contained by the IP message, as indicated in the destination port number field (i.e., the Destination Port)

The problem with this simple scheme is that it is black and white. One can either allow a certain protocol or block it. Fine-grained control of the protocol is impossible.

This is an issue because the SCADA/ICS protocols themselves have no granularity. From the perspective of the TCP port number, a data read message looks EXACTLY like a firmware update message. If you allow data read messages from a human machine interface (HMI) to a PLC to pass through a traditional firewall, you are also allowing programming messages to pass through. This is a serious limitation.

Deep Packet Inspection (DPI) is an extension to traditional firewall technology that can provide the fine grained management of EtherNet/IP traffic needed. Basically, DPI allows the firewall to dig deep into the message to understand exactly what the protocol is being used for. It is designed to understand the specific ICS protocols and then apply filters on fields and values that matter to control systems.[6]

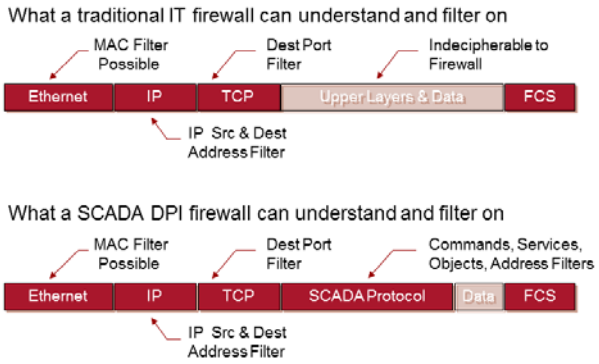


Figure 2: Comparing filtering options in a traditional firewall and a DPI Firewall. A traditional firewall cannot understand the ICS protocol so can only allow or deny the ICS messages as a group.

Can DPI technology analyze the message structure of EtherNet/IP packets and then make filtering decisions that would improve the security of a control system? The next chapter of this paper explores this question.

V. Analysis of EtherNet/IP for DPI

The main objective of the analysis of a protocol is to understand the structure of the protocol in its most basic form; that is as bits and bytes on an Ethernet network. While understanding the protocol from a high level is useful, it is important to understand the characteristics of every field in a message and how these fields interact. For example, consider the field "Length" in an EtherNet/IP message that is executing the Get Attribute All service on an Analog Input object. What exactly does "Length" indicate? Is there a minimum or maximum that, if violated, might indicate an attempted attack? Would these values change depending on the CIP object or service involved?

The second question is how can a given field (and all other fields that are interrelated with it) be evaluated at line rate while not impacting valid communications between client and server? For instance, imagine there is an inherent flaw in the TCP/IP CIP object used in a particular server. One cannot simply block all commands to that object, as the user may need to legitimately access it, for example to change the IP address in the server device. If you can identify the actual byte problem (perhaps it is an offset flaw) from a DPI perspective, it may be possible to block erroneous values yet also allow valid communication to continue. To do so, a detailed understanding of the protocol structure is required.

A. EtherNet/IP Header Structure

The EtherNet/IP header structure encapsulates CIP messages. The EtherNet/IP header is a 24 byte fixed length header with a trailing optional data portion. The maximum size of the packet, including header, is 65535 bytes and is ordered as little-endian as shown in Figure 3.

Table 2-3.1 Encapsulation Packet

Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	Encapsulation command
	Length	UINT	Length, in bytes, of the command specific data portion of the message, i.e., the number of bytes following the header
	Session handle	UDINT	Session identification (application dependent)
	Status	UDINT	Status code
	Sender Context	ARRAY of octet	Information pertinent only to the sender of an encapsulation command. Length of 8.
	Options	UDINT	Options flags
Command specific data	Encapsulated data	ARRAY of 0 to 65511 octet	The encapsulation data portion of the message is required only for certain commands

Figure 3: EtherNet/IP header layout. Note UINT is 16 bits and a UDINT is 32 bits.[1]

The command field represents actions that can occur via this layer. Session establishment occurs via the RegisterSession command (0x0065) and can be taken down via the UnregisterSession command. There are also various querying commands, such as ListInterfaces and ListIdentity. These commands provide client software with basic information about the device it is attempting to communicate to. The most important commands are SendRRData and SendUnitData, which have repercussions on the CIP encapsulated layer. All CIP messaging lays on top of EtherNet/IP SendRRData and SendUnitData commands. This field is an excellent candidate for DPI to limit communication pathways.

The length field is also important: it specifies total expected packet length minus the default EtherNet/IP header size of 24 bytes. Thus, if the actual packet payload size is less than or greater than the stated length, there is a problem with the packet.

Following the length field is the session handle. This field is unique to a session and is provided in a reply via the RegisterSession command generated from a PLC. This handle is used for the entirety of the communication session.

The status field is used by the replying entity to indicate whether the encapsulation message was executed successfully. Zero denotes a successful command; values above zero indicate the server had problems dealing with the request packet.

The sender context field is assigned by the sender and is echoed back by the receiver without modification in its reply. In many cases commands with no expected reply just ignore this field.

Finally, the options flag is not used and the packet should be discarded if its value is not zero. This field exists as a placeholder in case new data portions are required in the future.

The command specific data field as outlined in Figure 3 plays a large role in the format of the encapsulated CIP message. This section contains a fixed structure known as the common packet format (CPF). It contains a 16 bit field representing the amount of items to follow. An item is of two types: an address item or a data item. In practice an address item is initially listed, followed immediately by a data item. A connected address item is used when the encapsulated CIP message is connection-oriented and a sequenced address item is used for special CIP transport classes 0 and 1. These fields are then followed by the data item, either unconnected or connected. The unconnected data item is used for a number of CIP messages. One example is an Unconnected Send, seen embedded on top of a Message Router request. The connected data item is used when a connected CIP packet has been encapsulated.

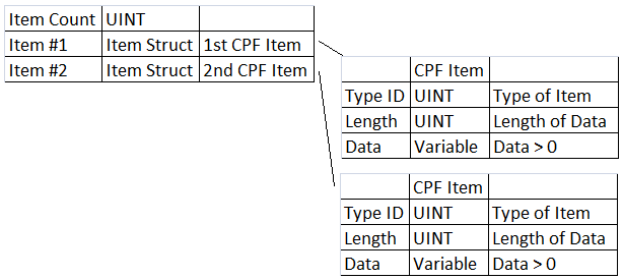


Figure 4: Common Packet Format outlining set of CPF items.

In most cases the item count is two, with the data item following the address item. One example exception is a special Large Forward Open type in which three items exist. The last item provides a special directive to the PLC.

B. CIP Messaging Structure

The CIP message structure has multiple static fields and a request path that contains a set of CIP segments including port segments, logical segments, network segments, symbolic segments, data segments, and possibly key segments, although not every packet contains every type of segment. In addition to this, there is always a CIP service executing some action upon the logical class segment. In some cases the CIP message may be a connection manager object using a Forward Open service for establishing communication. In other cases it could be a basic CIP service of a Get Attribute All on a particular object.

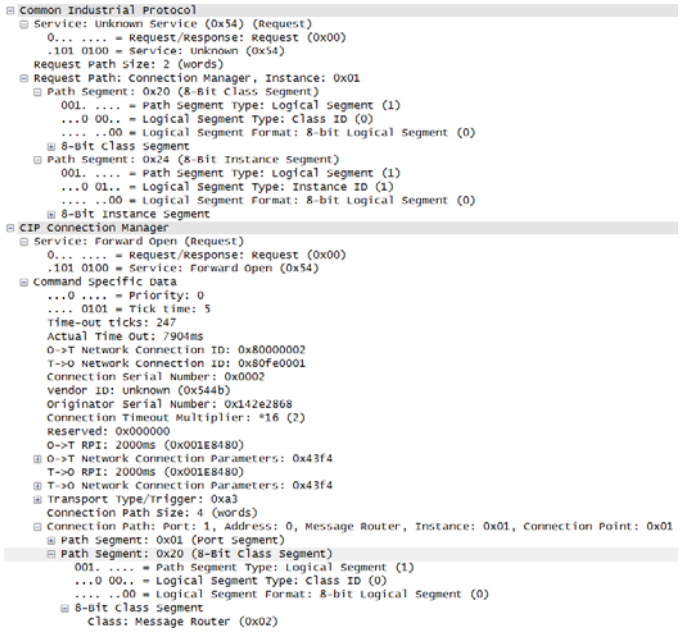


Figure 5: Connection establishment with Connection Manager class and message router class embedded.[7]

Figure 5 show the fields in a Forward Open, which initiates session establishment and also provides network connection parameters for the two talking devices. The CIP service field reserves the most significant bit to denote whether a packet is a request or a response. The next 7 bits are used to indicate the CIP service. As we will discuss later, the same service number can indicate completely different services for different objects. This re-use of service numbers complicates filtering decisions and is a challenge for implementing DPI for this protocol.

Following the service field is the request path size in words. In this example, the path size is 2, meaning 4 bytes (2x2). The request path contains the set of segments that are ultimately filtered upon.

For DPI design, the logical object class is a main field of interest, with the logical segment encoding being the most important. The logical segment is an 8 bit field with the 3 highest bits denoting the logical segment type, in this case 001b. The following 3 bits denote the logical type as pictured in Figure 6. Finally, the last 2 bits are the logical format and are used to identify the size of the segment value being one, two, or four bytes immediately following the segment type field.

	Logical Type		
Class ID	0	0	0
Instance ID	0	0	1
Member ID	0	1	0
Connection Point	0	1	1
Attribute ID	1	0	0
Special	1	0	1
Service ID	1	1	0
Extended Logical	1	1	1
	Logical Format		
8-bit logical value	0	0	
16-bit logical value	0	1	
32-bit logical value	1	0	
Reserved for future	1	1	

Figure 6: The accepted logical segment encoding values.

A special type of CIP message also exists; it is called a Multiple Service packet. This service is mainly used to decrease latency between requests, and is commonly seen on session startup or shutdown. It allows for an HMI to embed a number of CIP messages into a single EtherNet/IP header. The Multiple Service packet acts upon the message router object and contains an offset list of where the first CIP element starts, then the second, and so on. An interpreted example is provided in Figure 7. There is no stated maximum size of an offset list. A server device that receives one of these messages breaks out each encapsulated CIP message and replies in a one-to-many scheme. While quite useful for an HMI to transmit large sets of data, it deviates from a standard packet format and forces the DPI engines to develop a special case to handle these requests.

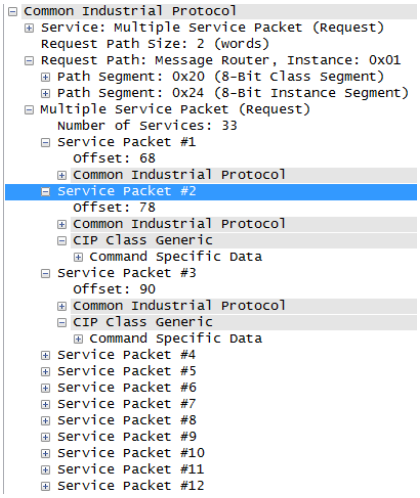


Figure 7: Wireshark outlining format of Multiple Service Packet.[7]

In contrast to a CIP request packet, as explained previously, the response frame does not contain the request path information; it merely contains the CIP service and a status as to the success of the response and optionally response data.

VI. IMPLEMENTING DPI FOR ETHERNET/IP

As noted in chapter V, the architecture of the EtherNet/IP and CIP layers are not trivial and contain many moving parts. Session establishment utilizes a set of commands with varying results and dynamic fields; encapsulation of CIP messaging is done in a tight coupling with the SendRRData and SendUnitData fields on the EtherNet/IP layer. CIP in itself carries complexity based on the type of connection, whether it is using a message router or Unconnected Send affects the packet structure. Following the connection type a request path embeds the various path segments in 8,16,32 bit forms which then contain logical segments, instance segments, attribute segments, and key segments.

How can you interpret this complexity and design a usable filter mechanism to ensure credible network traffic in an efficient manner? There are two main factors to account for. The first factor is message ‘sanity check’ actions, which are validation points against the protocol specification. These are used to ensure the structure and values are represented correctly in the packet. The second is the need to identify which fields or actions would make sense (from a user perspective) to be defined as filterable.

Let’s start with the sanity checking. There is no shortage of ways in which an attacker could make a PLC unresponsive if not protected correctly. To prevent this sort of Denial of Service (DoS) attack, validation must occur on both EtherNet/IP and CIP layers in tandem. This validation must also take into account directionality of request and response packets as they differ in their format. As noted earlier, there is a bit indicating request or response, so packets can be mapped against source and destination addresses and TCP ports. They can also be checked for correct packet lengths, valid request paths using data segments, valid CIP services and objects combinations, to name a few.

From the user’s perspective, sanity checking is executed ‘behind the scenes’ as it depends on integral knowledge of a protocol implementation. However, sanity checking is still required to be a selectable option because in some cases a vendor may fall short when attempting to adhere to the specification. Ideally, conformance testing should prevent this, but we have observed a number of tested products that vary from the specification in ways only seen on the wire. For example, vendors may interpret a field incorrectly or may implement a special case to meet some design need. Therefore, the ability for the user to turn off sanity checking is important and should be as simple as possible without removing all security.

The next area for filtering is those fields that could be user definable. Since CIP is an object based system with various CIP services acting upon these objects, it makes sense to filter on the fields that denote what object and service is being invoked. Then a knowledgeable user could specify the objects and services that are safe for the firewall to allow, and block all others. For example, to prevent an attacker from modifying the TCP/IP CIP object, the user could remove this object from their "white list" of allowed objects and services. Then the DPI firewall must identify that object in a packet and compare it against the allowed list. If the CIP service or object is not explicitly allowed, then the firewall will block this packet.

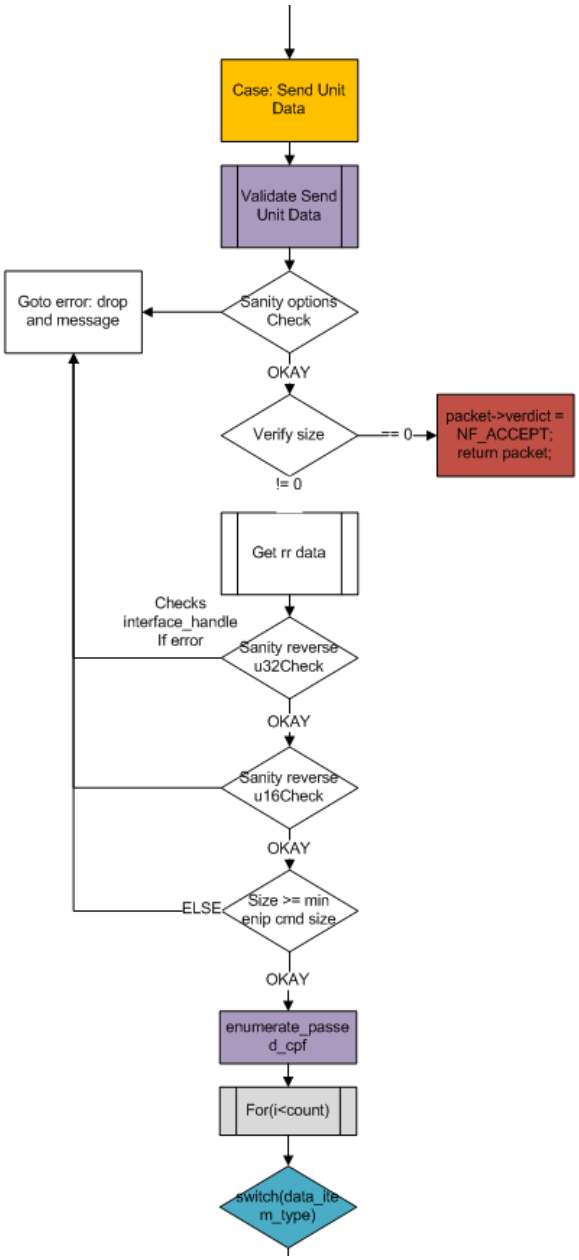


Figure 8: Initial processing flow of Send Unit Data.

The ODVA specification outlines a set of common services and optional object specific services that an object may adhere to. It also allows for vendor specific services. This grouping allows for the abstraction of a read-only filter list or a read-write filter list. As an example, one could group all Set Attribute {Single, All} as write commands, while the Get Attribute {Single, All} could be used as a grouping of read-only commands.

This becomes more complex when looking at the object specific services. If we look against the CIP service 0x54 of a Forward Open when tied to a specific CIP object, this exact same 0x54 on a different object could denote Write Set Value.

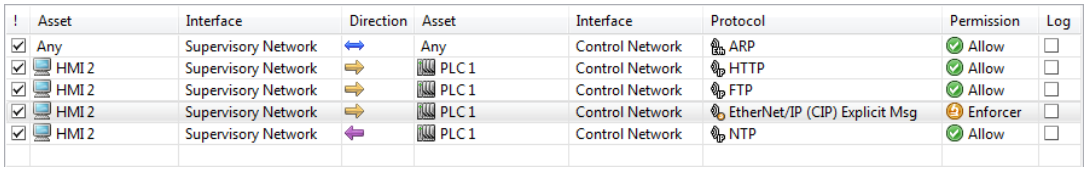
This means it is not enough to group CIP services as a whole; you must group CIP services and their partner objects together to develop an abstract grouping of functions. The benefit of creating this object/service pairing is that it provides a user a succinct way of protecting their EtherNet/IP communication stream.

Combine this intelligent filtering of object and service together with sanity checking and you have a powerful tool to not only validate on a per-packet basis but also permit only read functions. Expanding on the idea of combining CIP objects and services together, this filtering list can be dynamic and selectable; there is no limitation in the way the filter list set can be combined. For example, for a specific process it might make sense to allow a read-only function on the TCP/IP object, but permit read-write ability on Analog Output objects. This flexibility allows a user to customize the systems filtering on a per client/server pair basis.

VII. FIELD USABILITY OF DPI

The previous chapter describes some of the complexity in creating rules for filtering of EtherNet/IP and CIP headers. Unfortunately, the average control technician will have little understanding of the intricacies of the protocol. Many will not even know what particular objects or services their system requires to operate. Thus, there must be a means to present the rule creation in a simple, usable way for the technician in the field. This a graphical user interface (GUI) that guides the non-expert controls technician to create firewall configurations that make effective use of EtherNet/IP filters. And the GUI must do this in a way that is meaningful to the technician.

A possible starting point for a technician-friendly GUI is to use the concepts of the standard IT firewall, but extend them for deep packet inspection. This provides a platform for introducing DPI abilities to the technician. The technician would start by building a set of basic rules in the Firewall's ACL that matches the desired traffic at the TCP/IP layers. Then the user could specify those rules which he or she wishes the firewall to dig deeper into. This way, one can manage DPI rules alongside all of the other firewall rules that may not necessarily require DPI.



!	Asset	Interface	Direction	Asset	Interface	Protocol	Permission	Log
☑	Any	Supervisory Network	⟷	Any	Control Network	ARP	Allow	☐
☑	HMI 2	Supervisory Network	→	PLC 1	Control Network	HTTP	Allow	☐
☑	HMI 2	Supervisory Network	→	PLC 1	Control Network	FTP	Allow	☐
☑	HMI 2	Supervisory Network	→	PLC 1	Control Network	EtherNet/IP (CIP) Explicit Msg	Enforcer	☐
☑	HMI 2	Supervisory Network	←	PLC 1	Control Network	NTP	Allow	☐

Figure 9: Firewall ACL editor with EtherNet/IP rule highlighted for DPI configuration.

Figure 9 shows a set of rules in an example firewall ACL editor. In this example, the HMI physically exists on the Supervisory network (connected to one interface of the firewall), while the PLC exists on the control network (connected to a different firewall interface). As packets enter an interface on the firewall, the firewall checks the rules in the ACL to decide if the packets should be blocked or allowed to pass to the other interface. Notice that in addition to the EtherNet/IP rule, there are rules for traffic passing between the two devices to allow other protocols, such as FTP and HTTP.

One ACL field that may require additional explanation is the Direction field. In a stateful firewall, packets related to established connections are also allowed to pass through the device. This allows users to specify which entities are involved in connections and which direction the connections can be established without requiring them to specify additional rules for replies in the opposite direction.

This is most simply demonstrated with unidirectional HTTP (Web) firewall rules. IT firewalls will likely allow HTTP requests outbound to the internet from many of their workstations. This would allow workers to make requests to any website, but the website can only send data into the company network if the data has been explicitly requested by one of the workstations inside the company network. This functionality also applies to EtherNet/IP connections.

The highlighted rule in Figure 9 is one that matches EtherNet/IP traffic originating from the HMI and going to the PLC. Because the firewall is stateful it will automatically allow responses from the PLC to the HMI without an additional rule for replies from the PLC to the HMI.

By integrating DPI functionality with the existing ACL editor, the user gains several benefits. The main benefit is usability. The user can create their EtherNet/IP rules as they normally would in their ACL editor and simply add the DPI configuration to those rules, increasing the detail of their rule. This also improves the user's ability to track and maintain their firewall configurations. If the user was required to maintain a separate list for DPI rules, it could become difficult to visualize their complete firewall configuration. In real world control systems, firewall ACLs can become quite complicated. Maintaining multiple ACLs for a single firewall would compound the issue.

It is necessary to explore how to best add the DPI configuration information to an existing EtherNet/IP rule. Looking back to the example in Figure 9, the permission field in the firewall ACL table is expanded from the traditional Allow/Block options to include the “Enforcer” option. This is used to indicate that there are additional DPI conditions related to this rule. This reinforces that this rule is not just a simple allow rule. It also allows the DPI rules to visually stand out from the standard rules.

Keeping the ACL table fields common to all rules reduces the complexity of the table and improves overall readability and understanding by the user. For this reason it is best to opt for an “Advanced” or “Details” area that can be accessed for each individual rule. This provides a means to introduce fields and configuration related to one specific type of rule without complicating the configuration of simpler rules.

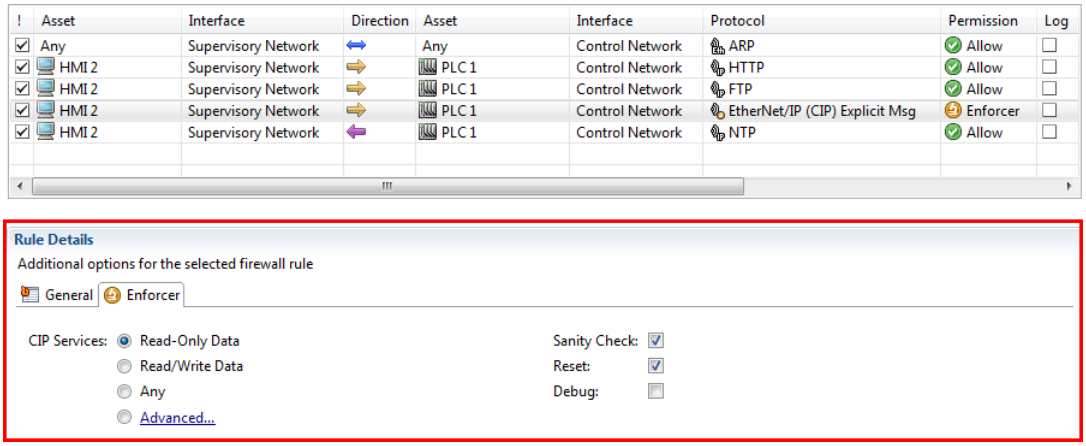


Figure 10: Editing the advanced details of an EtherNet/IP rule.

Figure 10 shows the advanced details editor for the EtherNet/IP rule selected in Figure 9. The editor automatically presents the user with all the DPI-related settings for the selected rule. Basic options related to the DPI functionality for this rule appear on the right-hand side of this editor. For example, the option to turn on or off sanity checking, as previously discussed, appears here. The left-hand side of this editor provides the user with the interface for configuring the rule conditions for CIP-related fields. This makes the most sense as it allows the users to create rules based on CIP services and objects. However, there is more involved than simply giving the user access to these fields.

Obviously there are many different types of users in the world who make use of devices that implement EtherNet/IP. These users have varying degrees of EtherNet/IP skills and knowledge. Some users know almost nothing about the protocol or the EtherNet/IP implementation details in the devices they are using. Others may be extremely knowledgeable regarding EtherNet/IP and have detailed knowledge about their devices, including the specific CIP objects and services the devices use. However, it is likely that many users fall somewhere on a spectrum between these two types of users.

The addition of DPI must not overcomplicate the firewall configuration for basic users, but should provide the advanced options desired by the more skilled users. If the process is too complicated, many users will simply opt to skip the DPI options. They will assume they are unable to configure DPI or won’t take the time to learn how to do it properly. On the other hand, high skill users will want the option to fine tune their ACLs for their use case.

In order to address this problem, the concept of CIP Service groups was developed. For the basic user, these groups allow the user to select a general description of the operations their device should be performing using EtherNet/IP. The firewall will then make the determination for the user as to which CIP object and service combinations should be allowed.

For example, if a company has a view-only console situated in an insecure physical location, the technician can select the Data Read-Only option to indicate that any data editing, commissioning, or programming actions should be blocked. Alternatively, for an HMI located in a secured control room, the Data Read/Write option might be appropriate, allowing full operator control but preventing PLC programming from the HMI. Utilizing CIP service groups allows the firewall to perform DPI-level analysis on EtherNet/IP packets without requiring the user to know the exact services being used by their devices.

Grouping CIP services will be limiting for more advanced users. For these users, providing the ability to select the specific CIP objects and their associated services is necessary. To accomplish this, you need an interface that is intuitive to technicians. If the GUI was to follow the CIP structure as it appears in the packet (see Figure 5), it

would be likely that the user would be required to select a CIP service and then identify the object. This is because the service field appears before the object field in the CIP header of the EtherNet/IP packet. However, conceptually it makes the most sense to users to identify the CIP objects that they use and then select the CIP services that they want allowed on those objects. In addition, identifying CIP Objects by class ID allows users to specify the CIP services permitted for classes of Objects, instead of just a single instance. Figure 11 shows an example of an advanced CIP service and object pair table for a single EtherNet/IP rule. The user has the ability to group by CIP Object or CIP Service.

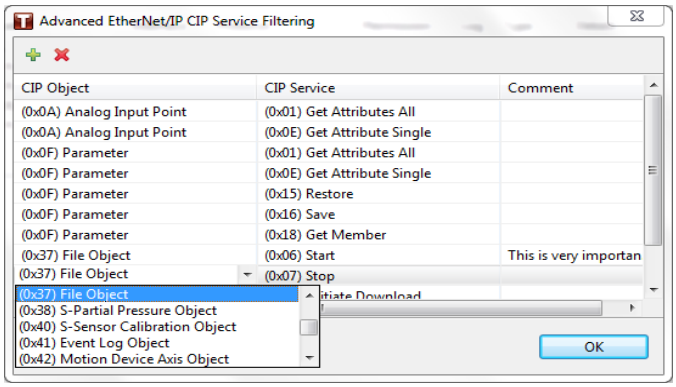


Figure 11: EtherNet/IP DPI CIP Object and Service pair editor.

When the user is selecting CIP object and service pairs, they are required to select a single object, followed by one or more services that are valid for that object. This is shown in Figure 12.

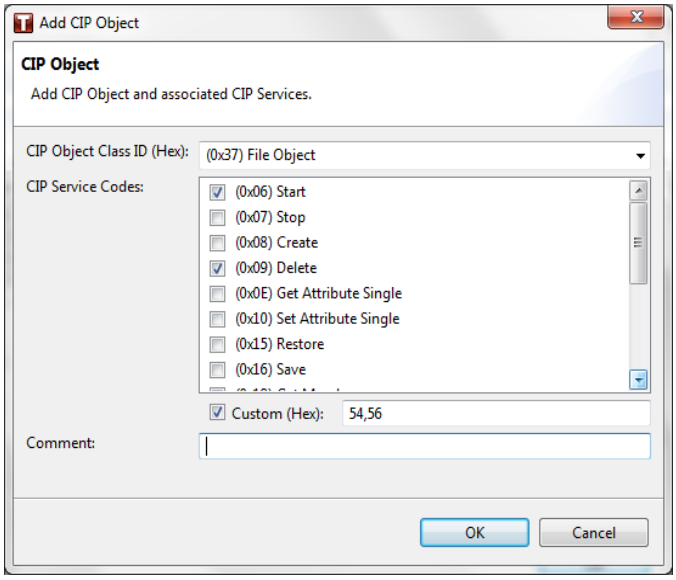


Figure 12: EtherNet/IP DPI CIP Object and Service pair selection dialog.

Selecting the CIP objects and their associated CIP Services should also be assisted by a dictionary of standard, well known, and common CIP objects and services. Without this, the user will have to refer back to documentation many times during the rule creation and, more importantly, when reviewing existing rules.

Of course, it is impossible to maintain a complete list of all used CIP objects and services, including vendor specific ones, or to predict those used in future implementations. Thus it is important that the user not be limited to the predefined object/service list. They should have a means of selecting an arbitrary class ID as well as service codes not found in the configuration library.

While creating the GUI for the EtherNet/IP DPI module a number of lessons were learned. First it may seem trivial to develop a means to allow users to implement DPI-based firewall rules, but without careful design, it was easy to overwhelm the user with information. It is also important not to overload those users with limited EtherNet/IP knowledge by presenting a complex interface. At the same time, the GUI must not limit more knowledgeable users who wish to make the most of the DPI possibilities. Care must be taken to find the perfect balance between complexity and usability.

VIII. CASE HISTORY

The effectiveness of DPI firewalls for EtherNet/IP-based control systems was illustrated in a project initiated in 2011 to secure the remote monitoring of compressor/turbine systems used on offshore oil and gas platforms. These inherently complex, high risk systems can be seriously impacted if rogue set points are sent to the PLCs controlling the turbines. Thus, it is important to prevent any accidental or malicious changes to either the logic or data values within the PLCs via the network. Yet at the same time, remote monitoring of these systems is critical, as their isolated locations mean that highly experienced staff are often thousands of miles away.

The initial security strategy was based on VPN tunnels to prevent third parties from intercepting and modifying communications between the onshore monitoring centers and the platform PLCs. However, a threat analysis indicated that a major risk remained. If either rogue insiders or malware compromised a computer at the monitoring office, it would be authorized to use the VPN. With this authorization, attackers could easily send EtherNet/IP commands to the PLCs capable of destroying the platform equipment.

The solution was the creation of a read-only firewall designed specifically for protecting turbine PLCs. Placed between the remote access control computer (basically a remotely controlled HMI) and turbine PLCs (Figure 13), a stateful firewall process in the firewall blocks all non-EtherNet/IP traffic on both the Turbine PLC and HMI interfaces. At the same time, it provides rate limit controls on all incoming traffic to prevent Denial of Service (DoS) attacks.

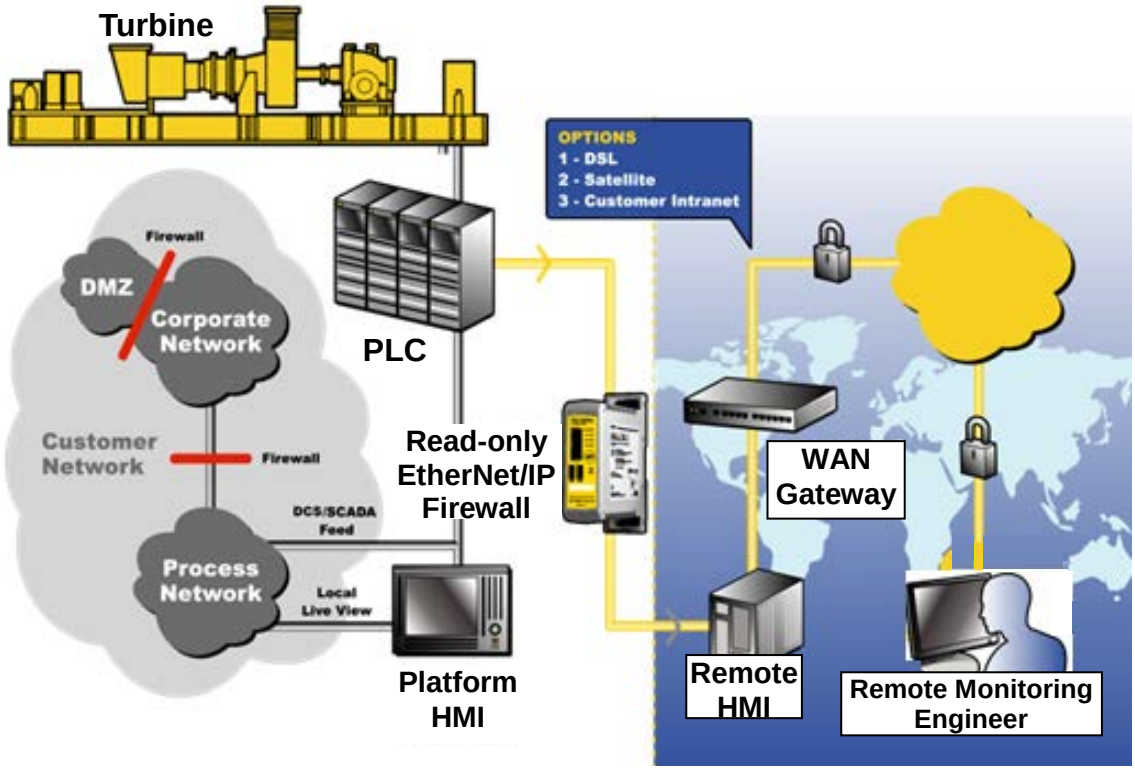


Figure 13: Deployment of EtherNet/IP read-only controller firewall in natural gas turbine protection system.

Once the stateful firewall application has provided this basic sanitization of the network traffic, the DPI application steps in to inspect each EtherNet/IP message to ensure that only certain CIP data-read operations are permitted through to the PLCs. In effect, any attempt to use CIP services other than 0x4c (Read Data Table), 0x01 (Get Attribute All), 0x03 (Get Attribute List), and 0x0e (Get Attribute Single) are blocked. The DPI engine also checks to see that all messages conform to certain key requirements in the EtherNet/IP and CIP specifications. Any network packets that are either malformed or contain inappropriate commands are blocked by the read-only firewall.

These EtherNet/IP read-only controller firewalls have now been operating for over two years, securing turbine systems on platforms around the world. During that time there has not been a single security incident. It has also had no impact on platform reliability or uptime.

IX. CONCLUSION

With the advent of advanced networking technologies like EtherNet/IP, every control system is effectively connected to a larger and hostile world. This is both a blessing and a curse. It is easier to manage and maintain systems, the network infrastructure is less costly, and improved productivity is possible. At the same time, critical systems are now more accessible to malicious acts and accidental events.

The good news is that Deep Packet Inspection technology offers a solution to the security challenges of EtherNet/IP-based control systems. DPI technology has been shown to be easy to deploy. It doesn't require the complete replacement of millions of dollars of existing SCADA and ICS equipment and it is effective in securing critical control systems.

REFERENCES

[1] The CIP Networks Library Volume 1: Common Industrial Protocol, November 2012

[2] E.J. Byres, M. Franz and D. Miller ; "**The Use of Attack Trees in Assessing Vulnerabilities in SCADA Systems**", *International Infrastructure Survivability Workshop (IISW'04)*, Institute of Electrical and Electronics Engineers, Lisbon, December 4, 2004

[3] ICS-Alert -11-186-01, Password Protection Vulnerability in Siemens SIMATIC Controllers

[4] ICS-ALERT-12-116-01A RuggedCom Weak Cryptography for Password Vulnerability

[5] Basecamp Digital Bond, Attacking ControlLogix: ControlLogix Vulnerability Report, 2012

[6] Eric Byres, Understanding Deep Packet Inspection for SCADA Security, http://web.tofinosecurity.com/download-kit-understanding-deep-packet-inspection, 20 December 2012

[7] Wireshark output, http://www.wireshark.org/, 20 December 2013.

[8] iptables - Linux kernel firewall, http://www.netfilter.org/projects/iptables/, 12 March 2013

[9] Linux netfilter, http://www.netfilter.org/, 12 March 2013

[10] Buffer overflow, http://en.wikipedia.org/wiki/Buffer_overflow, 12 March 2013

****