KENEXIS

Network Reliability Monitoring

Measuring the Health & Security of ICS Networks

SAFETY INSTRUMENTED SYSTEMS

PHA LOPA QRA

FIRE & GAS MAPPING

SCADA SECURITY & RELIABILITY

Introduction

One of the main requirements that the engineers that design and operate industrial control systems (ICSs) and supervisory control and data acquisition (SCADA) systems consider is determinism. The ICS and SCADA systems need to respond to information being sent across the network within a certain amount of time or the system can destabilize and/or fail. Multiple factors can affect the determinism of the system, including individual device performance, network performance, intra- and inter-system interactions, and security settings to name a few. While some of these factors can be taken into account when designing a system, like individual device and network performance, factors like intra- and inter-system interactions and security settings may be harder to determine a priori. Measurements need to be taken and data collected during testing and operation to assess the system's determinism.

Many different methods for determining network performance have been used in the information technology (IT) environment. Some of these can provide valuable information about network performance and determinism for ICS/SCADA environments, while others are not as valuable. Network metrics, such as latency, throughput, and jitter, have been defined for quite some time. The basic concepts behind these metrics apply to the ICS/SCADA environment, however they take on different importance based upon the determinism requirements of different ICS/SCADA network architectures and communications protocols.

This paper will discuss some of the things the author has learned by conducting ICS network assessments. The paper starts out by discussing some of the factors that may affect device performance in the ICS environment. It then discusses some of the differences between current monitoring systems and the ones used by the author. Lastly, the paper discusses some of the tools and techniques used by the author to analyze ICS network performance at the lower layers of the network architecture.

Master/Slave vs. Periodic Reporting vs. Reporting by Exception

ICS network protocols have, for the most part, grown up from older serial-based or bus-based fieldbus networks. They may now run over Ethernet and transmission control protocol (TCP) / user datagram protocol (UDP) / Internet protocol (IP) networks, however much of the way the devices interact on the network is based upon this fieldbus history. When every device on a fieldbus was connected via some form of bus network, the communications streams had to be limited in number, length of message, and periodicity. This led to some different methods for limiting the extraneous traffic on the network to be developed.

The simplest method was for complex controllers to act as masters on the network and the I/O devices to act as slave devices. The master would either command a slave device to take some action or request some information from the slave device(s) on the network. The slave device would never communicate unless specifically requested by the master. This is the simplest form of communications and is utilized by the greatest number of ICS device networks. In this case, the I/O devices can be extremely simple and inexpensive, since they don't have to include much in the way of processing power internally.

Another method is periodic reporting, which allows the I/O devices and controllers to interact in a peer-to-peer way. The controller and I/O devices go through a handshaking routine when starting the

communication stream that establishes the parameters for that stream, including the information contained in the messages and the expected periodicity. The two devices would then communicate in the agreed upon way. The I/O device and controller act independently, monitoring the communication stream for errors. If some error occurs while communicating, most controllers have internal methods for reestablishing the communication stream. In this case, the I/O device needs to have extra processing power since it is required to have some internal real-time operating system to continuously monitor the communication stream and detect errors.

A third method is called reporting by exception, which is somewhat of a cross between master/slave and periodic reporting. In this case, the controller may communicate with the I/O device in some periodic way to ensure that both devices are still operating, called a heartbeat. The heartbeat generally has a relatively low periodicity. The I/O device then remains mostly dormant outside of these heartbeat messages waiting for some condition to be met. When that condition is met, such as a sensor value being outside of some range, the I/O device would then send a message to the controller indicating that an exception to the norm had occurred. These exception messages could represent binary information, such as a value exceeded a set point, or they could contain more meaningful information. But the important point is that they are only communicated during a situation that is out of the ordinary.

Positive vs. Negative Metrics

Metrics can either be positively or negatively reported. In this case, a positive metric is a metric that is reported as a particular value (Boolean, integer, floating point number, etc.) that represents some physical quantity. Positive metrics allow the value to be utilized by another device and/or system as an input to some sort of process, such as a control function.

Negative metrics are those that report a good status unless an error occurs. Negative metrics are often used as status indicators for devices, where a good value indicates that the device is operating properly, and a bad value indicates that an error has occurred. Often, the status indicator will give some error code to help determine the error.

For the process control cycle, positive metrics are often process variables that feed into the control loop and are utilized every cycle. Negative metrics, while checked every control cycle, are not actively utilized in the control loop except when some specific condition has been met. If that condition does occur, these negative metrics may cause the system to go into another state or take a different sub-loop within the main control loop, but they will generally not be actively utilized by the control loop during normal operations.

Security metrics are often reported as negative metrics. Antivirus software is constantly checking the system in the background, but it is designed to be unobtrusive until an error occurs. If it detects a known virus signature, it alerts the user that something has gone wrong. Unless the user specifically checks the status of the software themselves, the antivirus software remains quiet.

In order to maintain situational awareness, it is more important to gain visibility into how the entire system is performing at any one time, regardless of whether an error or special condition has occurred. Trends can be established before they reach the point of setting off warnings.

NSM and SIEM

Network Security Monitoring (NSM) is "the collection, analysis, and escalation of indications and warnings to detect and respond to intrusions." It represents "a way to find intruders on your network and do something about them before they damage your enterprise." [1]

NSM usually involves collecting all the available log and alert information from different sources in a network and analyzing them for indicators of compromise (IOCs). These may be network traffic streams going to known-bad actors. It may be signatures detected for known types of malware. It can even be indications of new types of attacks based upon heuristics and threat analysis.

NSM is extremely useful for detecting known and emerging threats, since it isn't a single product that can be defeated. It generally involves collecting information from multiple different sources and aggregating it together to form a better situational awareness. Due to the fact that it is more of a methodology on monitoring network IOCs, it is more likely to catch an attack as it occurs then by using a single type of product or technique.

Security Information and Event Management (SIEM) systems are similar. They are products and services that provide "real-time analysis of security alerts generated by network hardware and applications." [2] They perform data aggregation, correlation, alerting, dashboards, compliance, retention, and forensics analysis. These are all extremely valuable for detecting IOCs in a network.

NSM Pros

NSM is specifically well suited to large computing environments where sensors can be placed at key points to detect IOCs. In the industrial environment, NSM sensors can be extremely effective in the upper portions of the architecture, generally layers 2 and higher in the Purdue Reference Model. [3]

Figure 1 shows the upper layers of an example ICS network architecture. The business systems and servers are connected to the Internet through the corporate firewall and the ICS networks through another firewall. A demilitarized zone (DMZ) is used to prevent direct communication between the ICS plant networks and the business network. ICS plant servers communicate with servers in the DMZ, and those servers within the DMZ communicate with servers in the business network.

Most of the traffic on the network at these upper layers crosses network boundaries. The protocols used are more common information technology (IT) protocols. Generally, the systems running at these upper layers are more common hardware and software platforms.

For NSM, traffic that crosses zone boundaries is important because it can be monitored. It is easy to deploy NSM sensors at the zone boundaries to monitor the network communications. Since most of the traffic at these upper layers crosses the zone boundaries, it can be monitored and analyzed more easily for anything suspicious.

More common IT protocols are better understood by the security community. This means that less ICS-specific protocol knowledge is required to analyze the network traffic, more tools are available, and

anomalies are easier to identify. Less time and effort needs to go into analysis, allowing more automation to be applied to the analysis.

Lastly, NSM sensors can be deployed easier on these common hardware and software platforms. Agent software can be deployed on servers to monitor system parameters in real-time without requiring specialized tools or ICS-experienced staff. This allows NSM sensors to be incorporated at more places throughout the network, alerting on IOCs earlier.
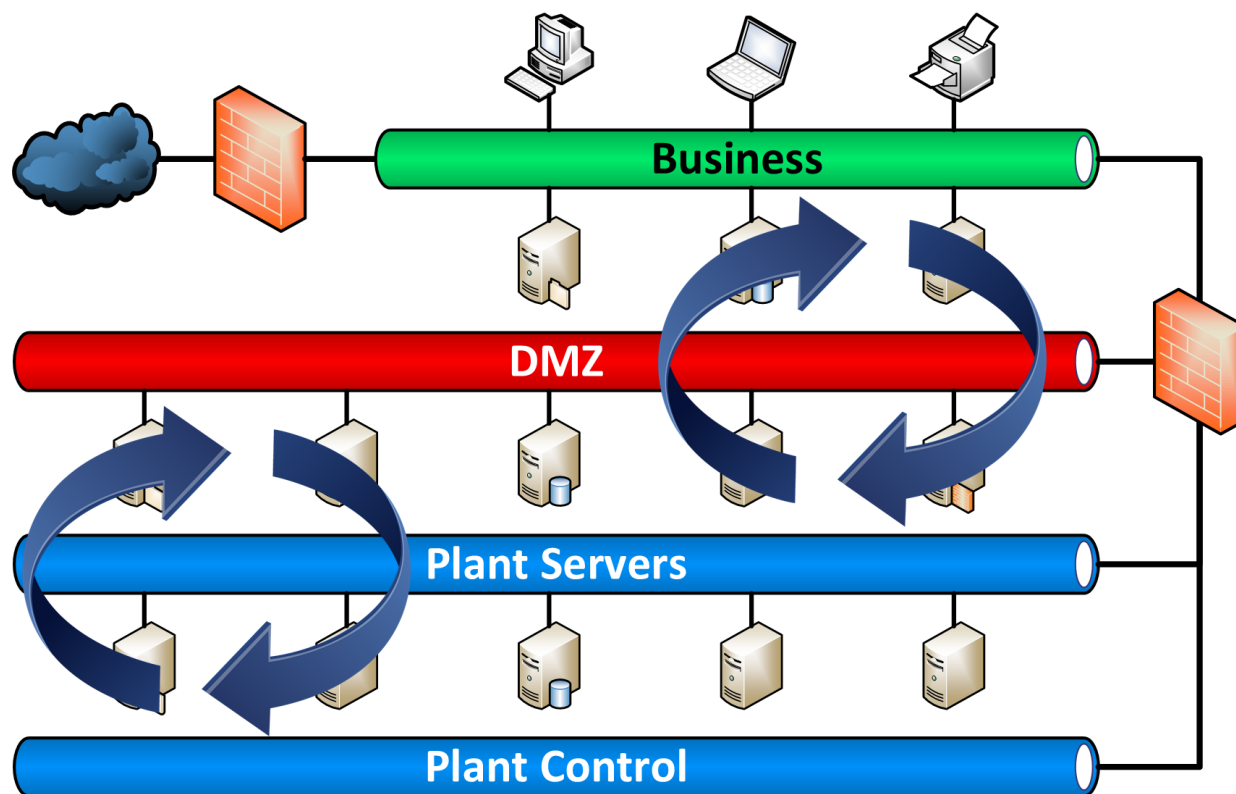

Figure 1 – Example ICS Network Architecture – Upper Level

NSM Cons

While NSM is very useful at the upper layers in the network architecture, it isn't as valuable at the lower layers, generally layers 0 and 1 in the Purdue Reference Model. Figure 2 shows the lower levels of an example ICS network architecture. In this example plant, there are multiple small workcells that communicate between the Ethernet-based plant control network and fieldbus networks. These workcells may represent robotic workcells in a manufacturing plant, individual process areas within a chemical plant, or remote control stations within a SCADA system. One of the workcells shown exists at a remote site that communicates across a radio link to the main plant control network. The plant control network also has operator human-machine interfaces (HMIs), engineering workstations, and specialized control equipment.

Most of the network traffic in these lower layers stays within each of the workcells. Only a small amount of traffic goes in or out of the workcells to the main plant control network when compared to

the amount that stay within the workcells. This leads to problems when applying normal NSM tools and techniques.
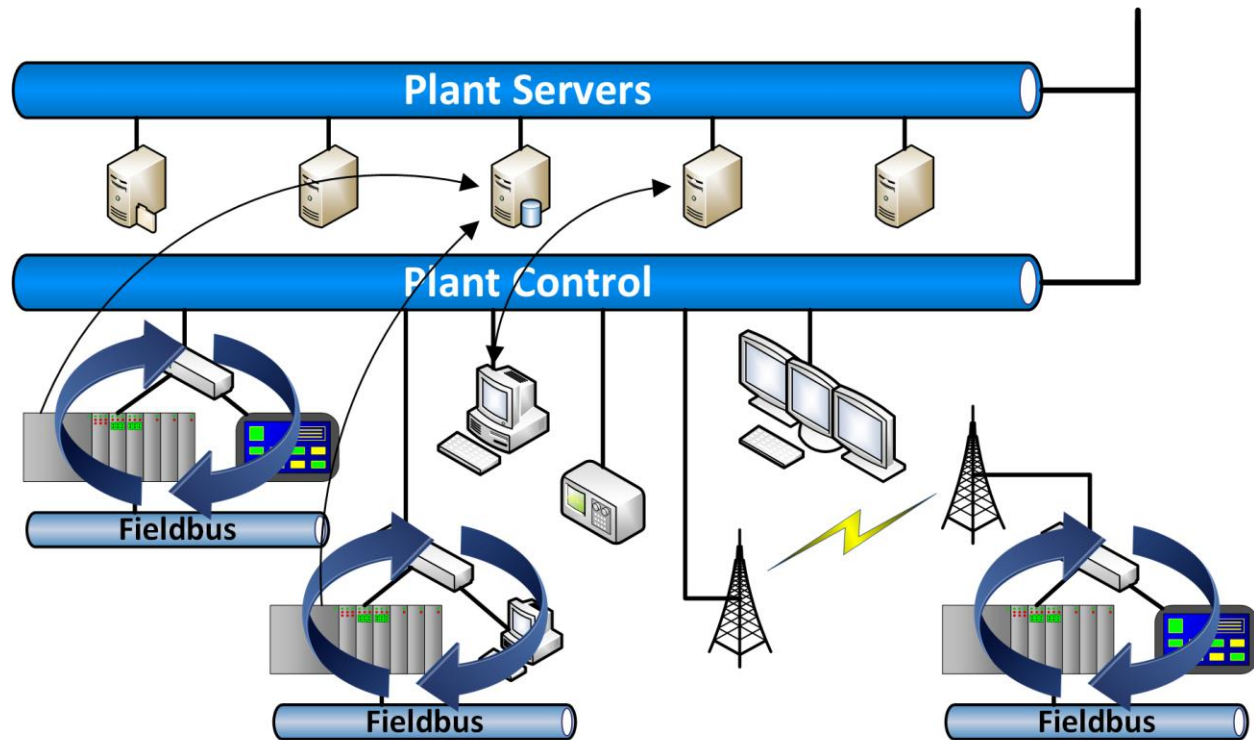

Figure 2 – Example ICS Network Architecture – Lower Layers

If "you can't observe the traffic that you care about, NSM will not work well." "Node-to-node activity, though, is largely unobserved at the network level." [1] These two statements describe how NSM is affected by conditions that normally occur in ICS networks. The bulk of the network traffic in the lower layers of ICS networks remains within the plant control network and never crosses zone boundaries. The protocols, hardware, and software used at these layers are much more ICS-specific.

The fact that network traffic generally stays within a zone, never crossing zone boundaries, means that the typical way to deploy NSM sensors in a network needs to be altered. More, dedicated NSM sensors need to be deployed around the network in order to gain situational awareness. Sensors would need to be added to each of the workcells to detect issues as they arise.

Since the hardware and software platforms at these lower layers are more ICS-specific, new systems would need to be deployed at each of these workcells. At a minimum of a couple thousand dollars apiece, the cost of deploying NSM sensors throughout a plant with a large number of workcells can add up very fast. In addition, the sensors need to each communicate back to a main NSM server that can do data aggregation around the entire network. This means that a large amount of network traffic will be added to the plant control network in order to maintain continuous monitoring.

Another problem is the fact that the protocols used at these lower layers are much more ICS-specific. NSM sensors would have to be ICS protocol aware in order to detect IOCs of any value. This adds cost and complexity to the system, and also limits the availability of tools and techniques.

Network Reliability Monitoring for The Lower Layers

The concept behind Network Reliability Monitoring (NRM) is very similar in principal to that of NSM. NRM uses data from multiple sources to establish a better picture of how a system is performing. The main differences are that NRM uses positive measurements of the network performance to establish that picture and doesn't require real-time monitoring.

One problem with NSM can be catching the IOCs in network traffic. If one of the NSM sensors isn't reporting anything, is that an indication that the system is acting properly or that the NSM sensor is down? The absence of a report doesn't provide enough information about the network to determine whether the system is operating properly.

By using positive metrics of the actual network traffic, NRM can establish the way that the network is performing. It takes positive measurements of different network performance characteristics and uses them to determine whether the ICS devices are performing as expected.

While it is possible to use NRM in a real-time manner, it is not the normal case. ICS do not change often, so testing is often conducted periodically or after significant changes are made. NRM can be used during factory acceptance testing (FAT), site acceptance testing (SAT), and commissioning to establish baseline signatures for the system. Then, during additional testing cycles, the new signatures can be compared to the baseline signatures. If changes are observed, or anomalies appear in the datasets, then a root-cause analysis will need to be conducted to determine why the change occurred.

Since NRM doesn't require real-time monitoring to look for IOCs, it is possible to reduce the number of sensors necessary to monitor large networks. A single capture device, or small number of devices, can be attached to the ICS network at various points for relatively short periods of time (minutes, hours, or days). These capture devices also aren't required to communicate over the network to a server, so the plant control network will be unaffected by the NRM system.

NRM Tools and Techniques

NRM can be both easy and complex at the same time. Many of the tools required to conduct NRM are either free or readily available. Wireshark [4] is an open-source packet capture and decoder software package that has become the de facto standard. Many ICS protocols have packet decoders developed for Wireshark, so there is no need to use proprietary software. Wireshark has the ability to produce simple spreadsheet files containing different network packet fields.

One of the more difficult tasks within Wireshark is to isolate the network traffic streams. Wireshark provides the capability to filter traffic based upon certain parameters within the network packets. This is important given that all of the packets are interleaved into a single file. All of the broadcast, multicast, and unicast traffic streams need to be separated into their individual streams to analyze their performance and reliability.

For ICS protocols, the process can be complicated, though. The process to generate traffic stream filters, while just tedious most of the time, can be made more complex by having proprietary or unknown protocols that are not previously built-in to Wireshark's normal set of packet decoders. It is possible to generate very complex Wireshark filters, however, any change in the protocol over time, and these may get broken, so care needs to be taken when analyzing these types of protocols.

After filtering the network traffic streams and generating data files that can be used to analyze each of the traffic streams, it is possible to analyze the traffic streams. Graphical analysis of these traffic streams can allow an engineer to identify anomalies in the data. Figure 3 shows an example of a network traffic stream being analyzed for cyclic jitter. The right graph shows a scatter-plot graph of the measured packet interval (MPI) vs. the test time, so the graph shows a delta-time vs. time. The MPI represents the measured time between subsequent packets in the same network traffic stream. With ICS networks and systems being very dependent on the deterministic communications between the devices, the network traffic streams need to be as close to the deterministic frequency as possible. Cyclic jitter is the variability in the actual network traffic stream packets with respect to that desired packet interval. In Figure 3, the desired packet interval was 10 ms. The mean of the MPI is 10 ms with a minimum to maximum cyclic jitter of ~500 µs.

The graph on the left is a histogram plot of the dataset. This gives some measure of frequency analysis, but is not as complete as a full Fourier analysis would be. This mostly helps the engineer understand bands that are visible in the data.

While the graph shown in Figure 3 is fairly stable with the majority of packets appearing at the center, a fairly tight distribution, and no long-term time events occurring, that is not always the case for all devices. Figure 4 shows another device during a separate test. In this case, the mean MPI was 1 ms with a cyclic jitter of ~210 µs with a fairly good distribution around the mean. The interesting thing to note in this device's analysis graph is the beat pattern that occurs approximately every 26 s.

Beat patterns of some sort are very common in ICS devices. The appearance, repeating frequency, duration, and such factors all vary depending on the device being tested, but the fact that the beat pattern appears is not unexpected.

One of the biggest problems with trying to understand the beat patterns or anomalies observed in the cyclic jitter analysis graphs is the root-cause for the beat or anomaly. Is it due to a device architecture issue, a network infrastructure issue, a security issue? Device architecture issues may be things like operating system garbage collection, antivirus software checks and updates, and on-screen operations. Many of the devices that are installed in the ICS environment are underpowered, so any condition that was not taken into account by the vendor can affect the network performance observed. There may also be network infrastructure issues as well, such as electromagnetic interference (EMI), signal degradation, or corrosion. The ICS environment is not always conducive to network infrastructure wiring and devices that were intended for the office environment. Sometimes, the large amount of machinery, chemicals, particulates, or other factors may affect the network traffic performance. There may also be cases where an actual security incident has occurred, and the beat pattern or anomalies

actually present an IOC. It is difficult to determine this without first having a baseline traffic stream to compare against.
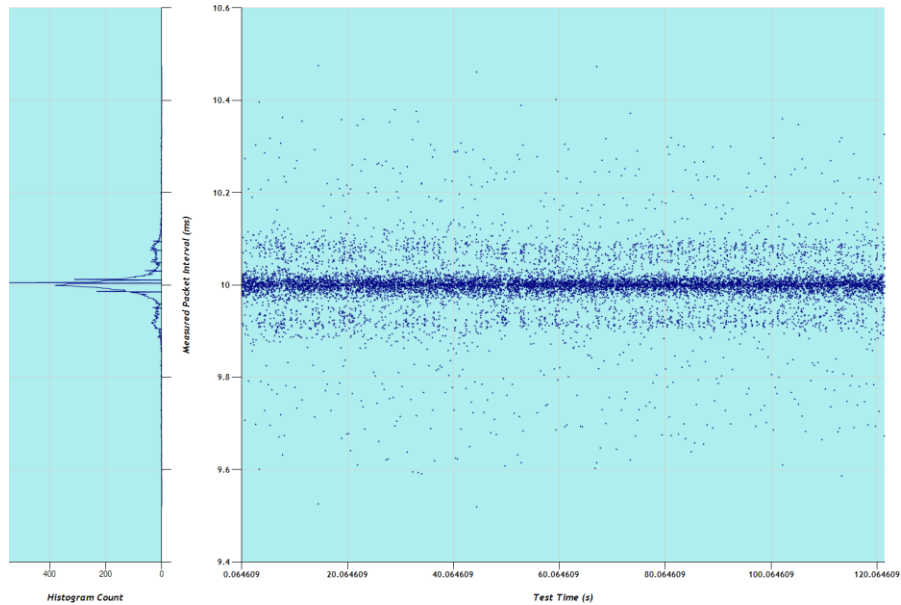

Figure 3 – Example Cyclic Jitter Analysis Graph #1
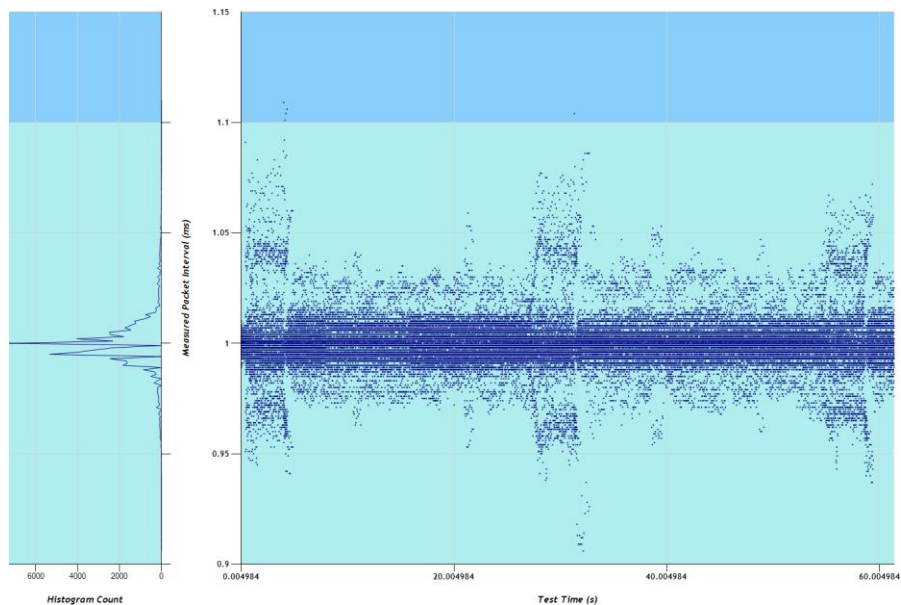

Figure 4 – Example Cyclic Jitter Analysis Graph #2

Baseline Testing

When looking for anomalies in network signatures, it is very important to develop a good baseline for comparison. While developing a good baseline may be difficult in the IT environment due to the constantly changing conditions, it is actually easy to do in the ICS environment.

Systems go through multiple tests before they begin operations, including factory acceptance testing (FAT), site acceptance testing (SAT), and commissioning. At each one of the tests, baseline signatures should be captured. These baseline captures can be used to when comparing future signatures.

In addition to the initial testing of the system, ICS generally don't change on a regular basis. The systems run as they were installed unless something in the system changes that predicates making a change to the ICS equipment or programs. This may be due to a process change, device replacement, program efficiencies, or other factors, but these are not the normal operating conditions. Systems can go years without having any change made to them at all.

In addition to the static environment, ICS environments are generally more consistent about using change management procedures. End users aren't perfect about recording every change, but when major changes are made to a system, they are generally documented. As part of this change management procedure, device signatures should be captured. This will allow the baseline signature to be updated as is appropriate for the newer configuration.

ICS equipment also goes through periodic testing during inspections and shutdowns. These may be yearly or multi-yearly, but they do generally occur at some regular period of time. Signatures should be collected for the systems during these as well.

Real-Time vs. Periodic Testing

It may seem that real-time analysis of these signatures should be done in order to maintain situational awareness. That may be true, but it isn't required.

Real-time testing requires that monitoring sensors be installed permanently throughout the network. These monitoring sensors need to be of sufficient power to be capable of capturing and analyzing network traffic in real-time and then reporting it to some sort of monitoring system. The infrastructure would need to be sized to handle the increased traffic generated by the monitoring system. An operator would have to be responsible for monitoring the results of the analysis as part of their normal control duties.

All of these factors make real-time monitoring of reliability data less operationally impactful. The amount of money saved via applying NRM in real-time wouldn't be justifiable in most cases.

The easier, and more impactful, way to use NRM would be to use a small number of temporarily deployable NRM sensors. These NRM sensors could collect and analyze network traffic at selected points throughout the system at different frequencies depending on the environment's criticality.

By using a smaller number of slightly more powerful sensors that are connected to the network for some duration of time, it is not necessary to modify the network infrastructure. The traffic would be captured locally at the desired network infrastructure device, so it wouldn't have to traverse the network in a way that it isn't already doing. It is also possible to collect data at various points throughout the network and post-process the data. An engineer could be responsible for doing the signature comparison at various times without having to view it in real-time as the data is captured.

Automating Testing & Analysis

While it may not be necessary to conduct continuous monitoring of the NRM system, it is preferable to automate the capturing and analysis process. The amount of data collected and the number of traffic streams in a normal ICS environment makes doing analysis by hand very difficult and time consuming. For a traffic capture of an hour, it can take multiple days to analyze all the traffic collected depending on the rigor applied to the analysis. Hence, automating the process is one of the first things to consider when choosing to use NRM.

Traffic analysis using tools like Wireshark in its graphic form is not desirable when looking at multiple capture files. It is possible to script the command-line version of Wireshark, called tshark. Small scripts can be generated that will take the desired capture files and performing the basic packet decoding repeatedly.

The bulk of the traffic analysis can be scripted as well. Many spreadsheet programs have some form of scripting language that can be used to import the files and conduct some calculations upon them automatically. An engineer would still be required to do some form of analysis on these, but the large portion of the analysis could be done by a computer without human interaction.

Data Analysis & Traffic Stream Statistics

When looking at graphs like the one shown in Figure 4, it is necessary to know whether that type of beat pattern is the normal operating procedure, or if that is something out of the ordinary. Because the datasets for traffic captures can be very large for even small capture times, it is necessary to convert the data into some form of mathematical statistics that can be monitored and compared over time.

One of the simplest and most common ways to look at the data is the average (mean) and the minimum and maximum values. These types of statistics are readily available from many devices themselves as part of their maintenance features. They do not tell the whole picture, though. Other statistical and mathematical factors need to be employed in order to get a better picture of what is happening.

Another simple statistic to calculate is the standard deviation. Even this simple metric can add a great deal of understanding of the traffic stream. If a traffic stream normally has a tight distribution, but starts to get wider over time, it may indicate a problem. The root cause of the problem will probably be difficult to determine, but an investigation into the root cause may show some new factor that affected the performance.

It can also be used as an IOC. Figure 5 shows the cyclic jitter analysis of a man-in-the-middle (MITM) attack conducted on a test system. [5] In this case, the mean remained the same, ~10 ms, however the distribution clearly changes. Side lobes at ~8 ms and ~12 ms appear. These are easily picked up during the standard deviation analysis.

Other types of analysis also may provide more information. Mathematical calculations, such as Fast Fourier Transforms (FFT), convolution, and correlation functions can be used to gain greater understanding of the traffic streams. These algorithms can be performed fairly quickly and with moderate hardware, so they should be considered. They do require some time to calculate, so they may

not be good for a first-pass on analyzing the data, but a more in-depth second- or third-pass at analyzing the data.
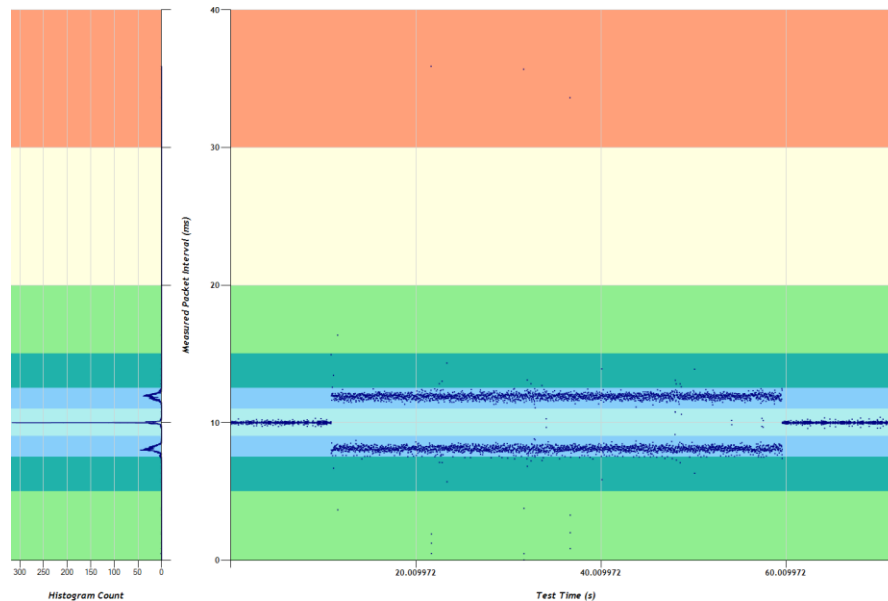

Figure 5 – Example Cyclic Jitter Analysis Showing MITM Attack

As a balanced, quick set of statistics to gain some basic understanding of each of the traffic streams contained within a dataset, the basic set of metrics that the author uses are:

- Mean
- Minimum
- Maximum
- Standard Deviation

These four statistics can be calculated in a short enough period of time that, for a fairly normal ICS network, the statistics can be calculated in approximate real-time. Traffic captures can be collected at a certain frequency, usually 30 or 60 s periods, and the statistics can be calculated for these traffic streams before the next traffic capture file is obtained. In this way, the basic data analysis can happen in near real-time. If more in-depth analysis needs to be conducted, the traffic captures can be re-analyzed using more complex mathematical and statistical methods, as described above.

In addition to being easy and fast to calculate, they are also very quickly and easily understood by engineers and operators. They can be shown on a dashboard and it is easy to compare those metrics as time goes by, building trends of how it changed. If the data goes significantly out of range, then a problem has occurred, and it requires investigation. In many cases, just being able to know that something is changing is good enough to alert personnel to a potential problem within the ICS environment. Not all indications need to lead to a major alarm condition, but they may indicate that some maintenance activity should be performed to bring the system back into line with its desired state.

While these four statistics seem to be very simplistic, they have proven to be invaluable to the author during multiple ICS network assessments. They provide a good enough set of information that the engineers can understand the data without being inundated with too much data. They also provide a good enough view of the traffic streams that certain streams can be targeted for more in-depth analysis using much more complex graphical and mathematical methods.

Summary

NSM is a good idea for many systems but it isn't really designed for the lower layers of the ICS environment. Many simple tools exist that can be used at all layers within the ICS environment to conduct some form of NRM. The tools and techniques discussed within this paper are generally better at the lower layers within the architecture, though.

It isn't necessary to perform continuous monitoring for NRM in the ICS environment. In fact, periodic, localized testing is probably a better idea given the numerous changes to the may be necessary to accommodate continuous NRM.

Automating the testing and data analysis is very important for NRM, since the number of traffic streams can easily require orders of magnitude more time to analyze as networks get larger. The tools and algorithms required to analyze the traffic need not be incredibly complex to gain a basic understanding of the ICS environment. More complex analysis can be performed on an as-needed basis when anomalies are detected, but are not needed in normal operations.

Bibliography

[1] Bejtlich, R., *The Practice of Network Security Monitoring*, No Starch Press, 2013

[2] Wikipedia website, https://www.wikipedia.org/

[3] ANSI/ISA-95.00.01-2010 (IEC 62264-1 Mod) – *Enterprise-Control System Integration – Part 1: Models and Terminology*

[4] Wireshark, https://wireshark.org/

[5] Gilsinn, J., *Mechanics of an ICS/SCADA Man-In-The-Middle Attack*, Presented at BSides Delaware 2015, November 13, 2015

KENEXIS

Kenexis Consulting Headquarters
3366 Riverside Drive, Suite 200
Columbus, OH 43221 USA
Phone: +1-614-451-7031
Fax: +1-614-451-2643
E-Mail: Info@Kenexis.com
Website: www.kenexis.com

UAE – Kenexis DMCC
Jumeirah Lake Towers
Almas Tower DMCC
Dubai, United Arab Emirates
P. O. BOX 31303

Kenexis Asia-Pacific Pte. Ltd.
9 Wilkie Road
03-01 Wilkie Edge
228095
Singapore

Kenexis Houston
Coming Soon!

ISO 9001:2008 Certified ✅

Kenexis is an independent engineering consulting firm providing Safety Instrumented System (SIS) services including the establishment of Safety Integrity Level (SIL), developing Safety Requirement Specifications (SRS), and quantitative design Verification (SIL Verification). Kenexis analyzes the risks associated with process plant operations using different techniques including PHA, LOPA and QRA. Likewise, we provide Fire & Gas Mapping services including performance target selection, detector placement and coverage assessments. Our cybersecurity team provides industrial control system services including design, vulnerability assessments and incident response.