

Session 6: Network Defense, Detection, and Analysis

Welcome to the last of the six sessions of the Industrial Control Cybersecurity (301) Training. The sessions you have completed thus far have provided important building blocks for this part. Before jumping into the specifics of defense, we need to touch on the importance of communication in the process.

- Are you part of the 'IT' side of the company?
- Are you part of the 'ICS'/engineering side of the company?
- Do you talk to one another?

Generally as we visit companies, we see a battle between the two organizations. Each side blaming the other for the state of affairs.

From the ICS side of the house we hear:

- IT wants to come in and change everything.
- They don't understand and don't want to learn about our environment.
- They insist we do things their way or they'll tell management.
- Our manager doesn't like the IT manager and forbids us to talk to the other group.

From the IT side of the house we hear:

- There systems are so old we don't support them anymore, why don't they just upgrade to what we support.
- We have company policies to live by and enforce, but they won't follow any of them.
- Our manager doesn't like the ICS manager and forbids us to talk to the other group.

Best advice: Take out a pencil and tie a white tissue to it. Then, find out what the other side likes to eat/drink and solve this situation!



Here are some benefits to solving the issues:

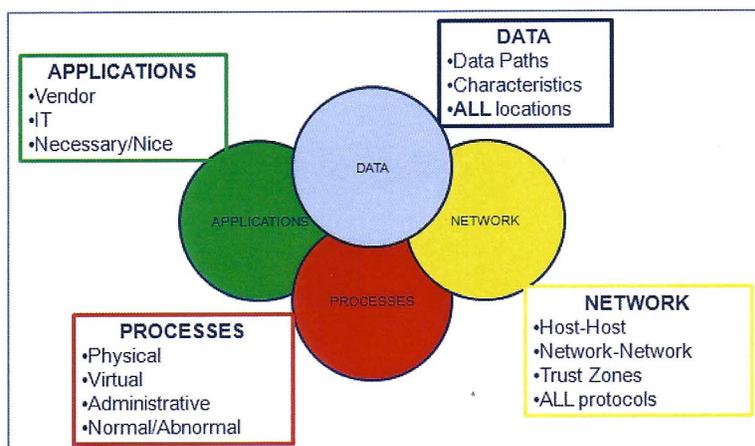
1. Better understanding of issues.
2. Shared man-power.
3. Shared funding.
4. Shared resources (equipment, software costs, licensing, etc.).
5. Together you can meet company security requirements.

Once at a conference a student asked about manpower (full time employees) loading for IDS and firewalls. The answer was at least two employees for each to allow for vacations, sick leave, meetings, etc. Immediately panic struck the poor student thought he had to come up with funding for four employees. We told him, "No. Go tell the other side that you'll buy two if they'll buy two because you both have the same equipment and the same problem." You would have thought he had just won the lottery!

Learning Objective 13

LO13: Describe the components for defense in depth.

The first step of a good cyber defense implementation is to **Know Your Environment**. Your entire environment, not just things like the ICS protocols. When protecting your environment, focus on the sensitive/critical data first.



First we need to define what we mean by "our environment." It is:

- Data – paths, characteristics, and location(s)
 - What are its characteristics?
 - Is protection of integrity required?
 - What data must be available 24/7?
 - Are backups in place?
 - What is the categorization of the date (public, proprietary, etc.)?
 - What is the location of the data?

- Applications – any software tool that touches the data
 - Which applications are critical to business?
 - Where do these applications reside?
 - How do we protect the servers and applications?
- Processes – physical, virtual, and administrative
 - What processes are in place for cybersecurity? For risk management?
- Network – how the data moves and which systems create or use it
 - Do you have a documented network design?
 - Do you know where all you network equipment is and how it's configured?

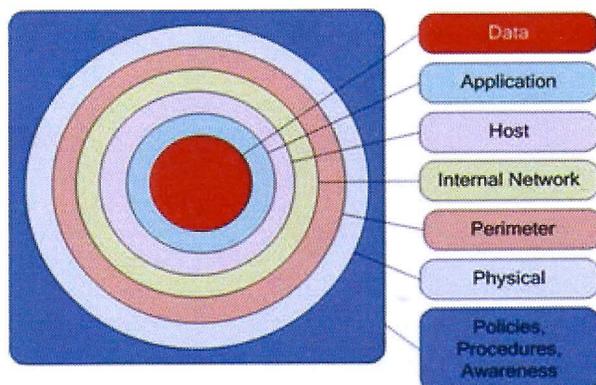
Once this is done, we can define what our baseline or NORMAL environment should look like.

Defense-in-Depth Security

Intrusions today start with the client side attacks for a reason – the firewalls, patching, etc., will stop the easy stuff. Egress filtering is a norm in most corporate environments. If it isn't at your location, you need to make it a priority. Focus on your operational security and develop incident handling capabilities.

Defense in depth is a layered approach to defending any environment, especially an ICS environment. It requires developing defenses for all systems and subsystems in the environment including:

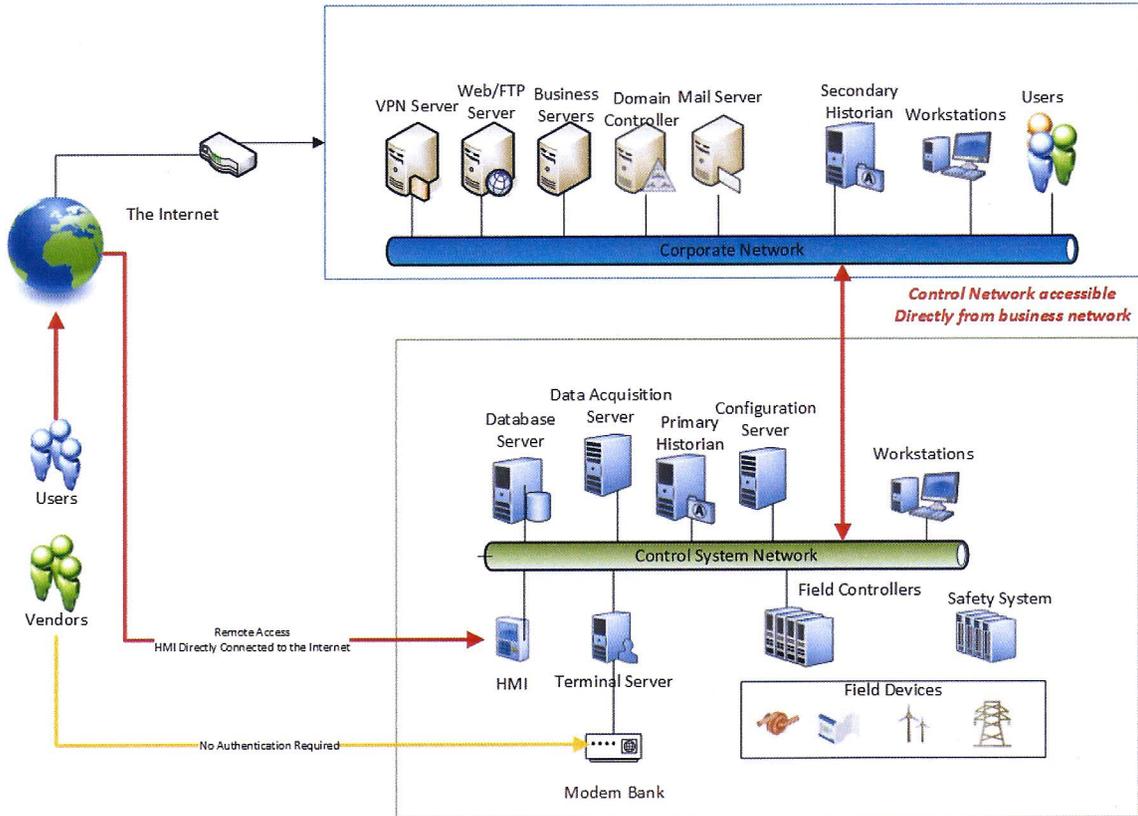
1. Data
2. Application
3. Host
4. Internal Network
5. Perimeter
6. Physical
7. Policies, Procedures, Awareness (People)



Network Architecture

The following diagrams represent a generic facility and shows the electronic perimeter and a representation for the external communication links.

Modern Connectivity – Wrong Way



The first diagram shows a typical facility that has not considered security into its design. Notice how the communication paths bypass any type of security for the control system network. Firewalls are not implemented. Intrusion detection systems are not implemented. There is direct access to the control system network from the internet with no boundary checks. The sad fact is this configuration is typical of small municipalities and companies.

The second diagram on the next page has the infrastructure divided into three network segments or Demilitarized Zones (DMZ) or enclaves:

- Control system LAN
- Control System DMZ (historians, web servers, etc.)
- Corporate environment with separate DMZ's for various public functions, e.g., web, vpn, wireless, etc.

The colored pipes are the data conduits that allow data to flow between the various zones in the environment. Any of these conduits could be an entry point for malicious traffic and need to be protected as a front line.

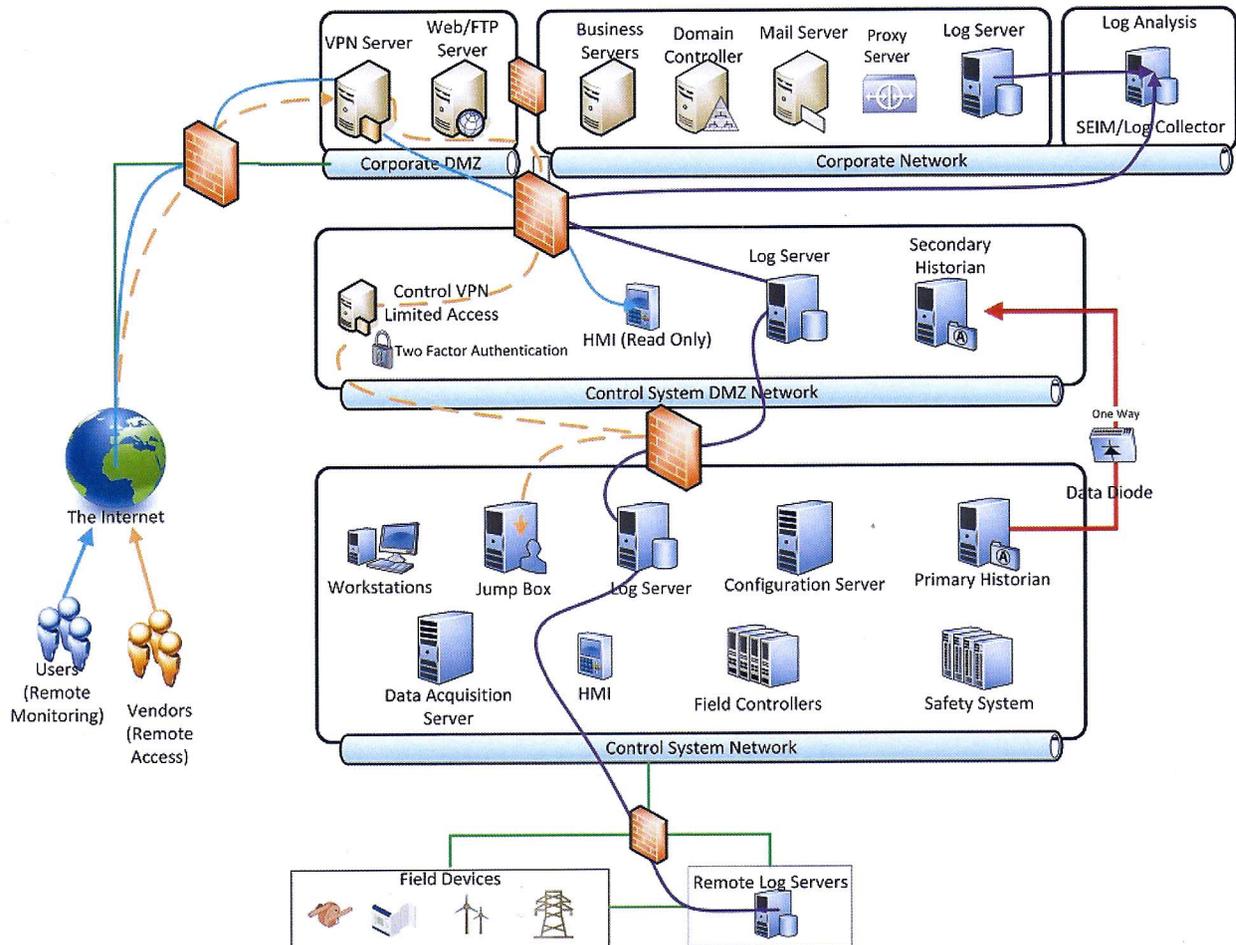
The firewalls are placed at the front line of defense for each of the various zones. These firewalls provide the trusted path for users and applications to communicate with and between all of the various pieces of computer equipment in the facility.

There are two complimentary principles for segmenting networks. The first principle includes the general functions of a system:

- Serve external customers
- Handle facility environmental controls
- Support IT
- Process HR data
- Store/serve ICS process data
- Run/Supervise ICS.

The second principle is trust level. What is the sensitivity of the data/system/data path? Segmentation should be implemented using firewalls or at least routers with access control lists (ACLs).

Modern Connectivity – Best Practice



The **Maximum Transmission Unit (MTU)** is a parameter of layer 2 (Ethernet). The maximum size for an Ethernet frame is 1500 bytes. Thus, the MTU is generally 1526 bytes. This actual number is negotiated and depends of the parameters of the network.

Firewall Implementation

Firewalls are your front line of defense. This is where you want to stop the bad guy right at the front door. The most important aspect of installing any kind of firewall is **Know your environment!**

- How does data flow?
- How is data used?
- Who uses the data?
- What is the security rating (CIA/AIC)?

For network and host firewalls, there are two general classes:

1. STATELESS Firewall

Stateless firewalls watch network traffic, and restrict or block packets based on source and destination addresses or other static values. They are not aware of traffic patterns or data flows. A stateless firewall uses simpler rule-sets that do not account for the TCP session state or that a packet might be received by the firewall pretending to be something you asked for.

2. STATEFUL Firewalls

Stateful firewalls can watch traffic streams from end to end. This means that stateful firewalls are totally aware of what stage a TCP connection is in (open, open sent, synchronized, synchronization-acknowledge, or established), it can tell if the MTU has changed, and whether packets have fragmented, etc. They are fully aware of communication paths and can implement various IP Security (IPsec) functions, such as tunnels and encryption. Neither type of firewall is really superior over the other. There are good arguments for both types of firewalls. Stateless firewalls are typically faster and perform better under heavier traffic loads. Stateful firewalls are better at identifying unauthorized and forged communications.

- Know your environment! How does data flow? What does data mean? What is data usage?
- Firewall vendors are now creating firewalls which support multiple ICS protocols and standards.
- Trade off speed/throughput vs. security vs. cost.
- Erroneously deployed as a cornerstone of architecture.
- May introduce massive architecture rebuilds.

There are three categories of firewalls:

- **Network:** These firewalls act as the gate keepers of the network. Determining what traffic is allowed to pass into [INGRESS] the next (trusted) area AND what traffic is allowed out [EGRESS] of the trusted area.
- **Host-based:** Host-based firewalls include utilities that watch the

activities and data on the actual host. These include antivirus, configuration monitors, log watchers, etc.

- **Application:** Application firewalls monitor and check the traffic to and from a particular application. The most common are those that monitor web transactions to/from a web server/application.

The following is a list of features offered in most network and host-based firewalls:

1. Policy-Based Access Control
2. Packet Filtering
3. Network Address Translation
4. Proxy
5. Encryption
6. Protocol Tunneling
7. Virtual Private Networking (VPN)

Writing Firewall Policy (Rule Sets)

Once the security policy AND the architecture have been planned in detail, it is time to construct the firewall rule set. Rule sets are critical. A misconfiguration somewhere could expose your environment to unwanted guests or could cause a disruption in high value data streams.

Most firewall rule sets work in the same manner. Rules are read and processed in *sequential* order! When a packet enters the firewall it is compared to the very first rule. Each rule is then tested until it finds a rule that matches the characteristics of the incoming packet. If no packet matches, **MOST** firewalls will drop the packet by default.

Order is very critical to the structure of the rule set!

The same basic rules apply regardless of which type of firewall you are implementing:

Rule 1: Disable default settings. Ensure you have any default setting disabled. You can turn them on one at a time later.

Rule 2: Keep the rule set simple.

Rule 3: Keep the rule set easy to understand. Document the rules. What does it do? Why did you write it? There had to be a reason.

Rule 4: Put specific rules before general rules:

- Lock down to specific user.
- Put time limits on the connections.
- Use all the capabilities available to you.

Rule 5: Put common rules at start for better performance.

SUGGESTION: Write the firewall rules out as though you were explaining what the rules do. Once you have this documented, then create the actual rules in the required syntax. This document then becomes the baseline documentation for your firewall. It is easier and less mistake prone to do it this way than to just sit down and start coding rules.

Rule 6: That which is not explicitly allowed is denied.

Rule 7: Log the actions of the firewall and route (if possible).

Example Firewall Rules

Rule	Src IP	Dst IP	Protocol	Dst Port	Action	Description/Comments
1	Any	192.168.10.10	TCP	80	Allow	Allow Internet to access DMZ Websites
2	192.168.0.10	192.168.10.20	TCP	1433	Allow	Allow PLC to send data to Historian on DMZ
3	204.134.25.201	192.168.10.30	TCP	22	Allow	Allow user to access DMZ system via SSH for management
4	Any	Any	Any	Any	DENY	Cleanup Rule. Deny all that does not match above rules

Remember the **KISS** principal: “**KEEP IT SIMPLE STUPID!**” You can buy the most expensive elaborate firewall available, but if there is a mistake in your rule set, you **WILL** have uninvited “guests” in your environment.

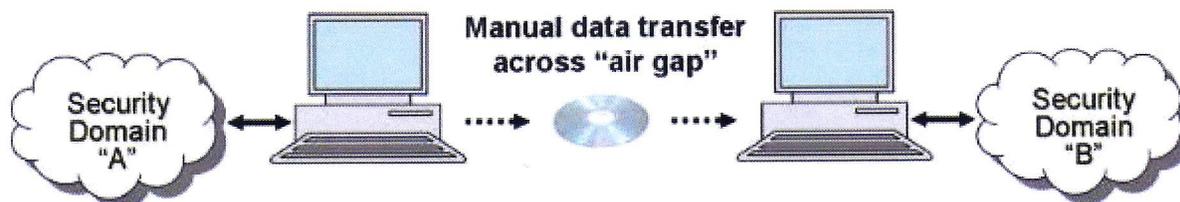
The most common mistake is having a general rule at the first that overrides a more specific rule later on in the rule set.

When searching for a firewall there is always a trade-off between speed/throughput vs. security vs. cost. You have to make the choice based on your knowledge of your environment and budget.

Be aware that the introduction of a firewall into your environment may introduce massive architecture rebuilds. Plan accordingly. A quick install is a recipe for disaster.

Data Diodes

The idea of unidirectional networks has been around since 1960. The Australians further developed the idea and the concept of a data-diode was born around 1990. Before that, the usual method of unidirectional data transfer was done via physical media and sneaker-net.



Data-diodes can be implemented in hardware, software or a combination of both. The hardware implementation is the most secure because it is physically impossible to send any messages in the reverse direction. Physical one-way links cannot be hacked with software tools. NIST provides a specific security controls, NIST 800-53, AC-4.7 that describes using hardware enforced one-way communications as a threat-mitigation method.

Data-diodes are often compared to firewalls with rules to pass data in only one direction. The fallacy in this logic is that in a firewall, even though the **DATA** is passed in only one direction, many protocols both network and/or application, are two-way conversations. In the data-diode case the TCP conversations are between the external devices and the data-diode, **only** the actual data is transmitted within the hardware confines of the data-diode.

Keep in mind that:

- It can be easy to build a simple hardware data-diode.
- There is nothing wrong with implementing several of these types to reinforce border security.
- Most hardware solutions focus on passing the data and not the actual network packets.
- It is difficult to engineer the implement the hardware solution both easily **and** with provide high quality of service, low error-rate and high transfer speeds.

Traditional one-way transfer configurations are discussed below. Each method has its advantages and disadvantages.

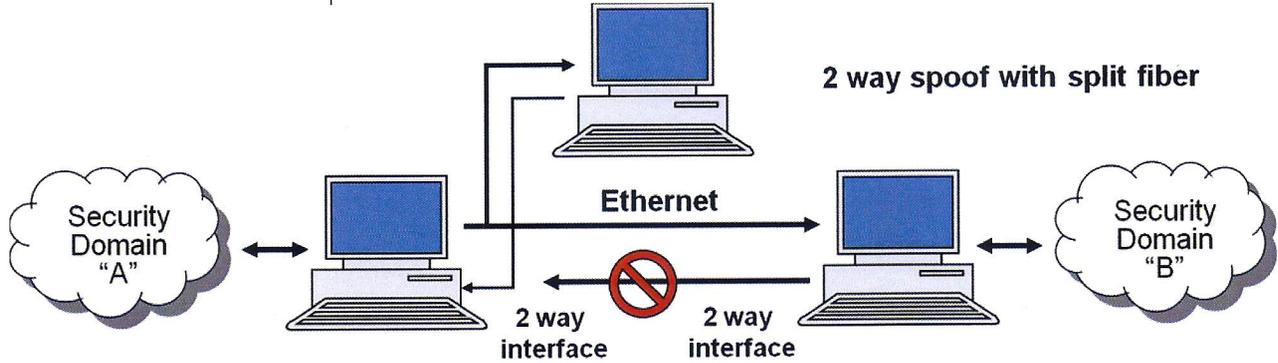
Clipped RS-232 (or Fiber) Data-diode

A very simple hardware data-diode can be built by modifying an RS-232 serial cable by simply clipping the return wire. A similar device can be created using a media converter to go from RS-232 to fiber. Sandia Labs received a US patent in 1997 for just such a device.



Ethernet Fake Out

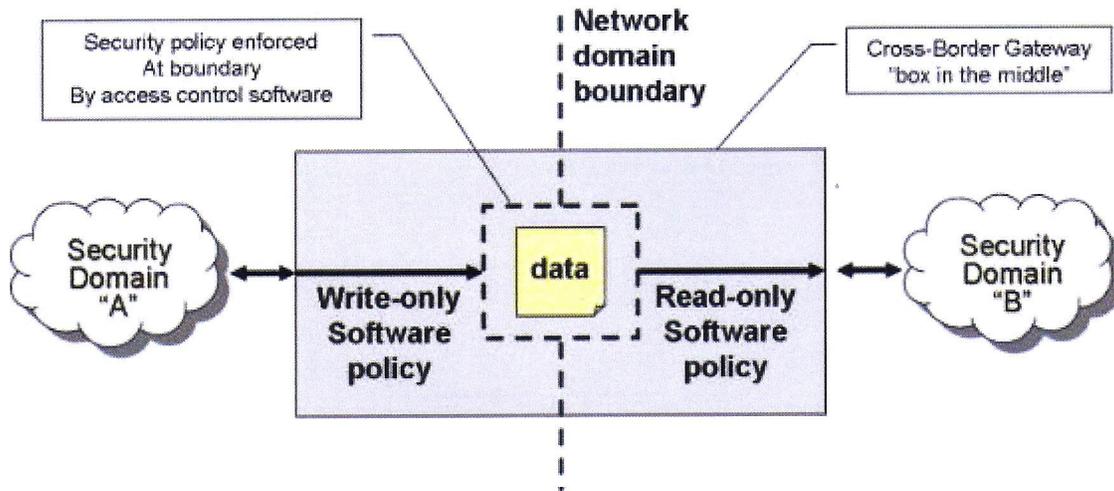
By employing a network splitter, one can fake out a TCP. It is similar to using a man-in-the-middle scenario.



Complex Software Security Policies

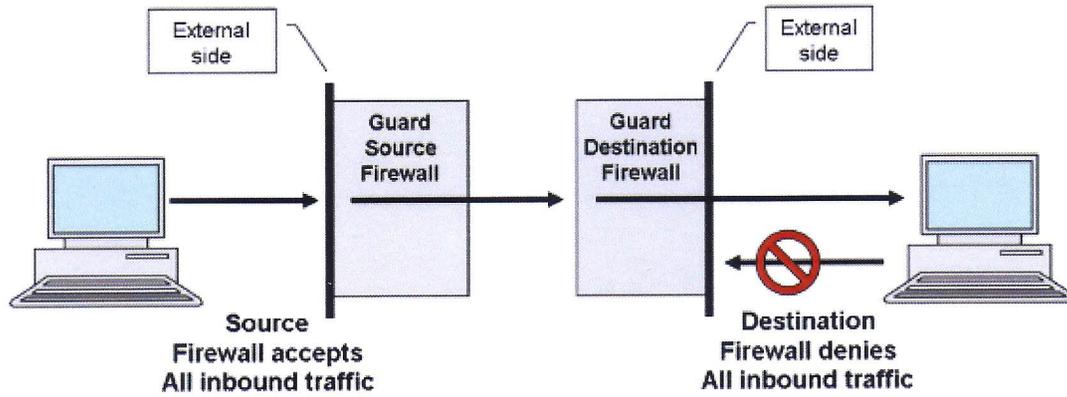
The software implementation is usually implemented using operating system policy tools, like SELINUX, that allow you to control data flow/access within a single hardened host. There are four requirements necessary to make software security policies work:

1. Base Operating System security configuration.
2. Testing of policies.
3. Configuration Control of policy files.
4. Logging, Logging, Logging.



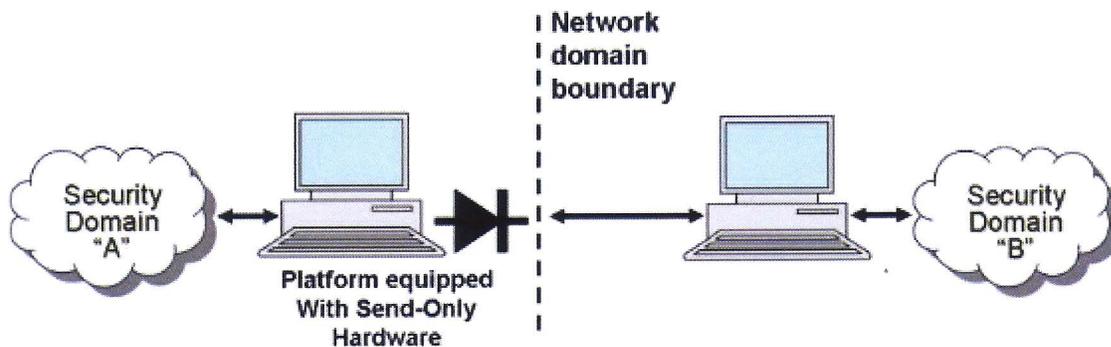
Double Firewall One-Way Transfer

In this method the connection between the two firewalls is a trusted connection, probably physically configured side-by-side. This environment requires tight configuration control of the firewall rules.



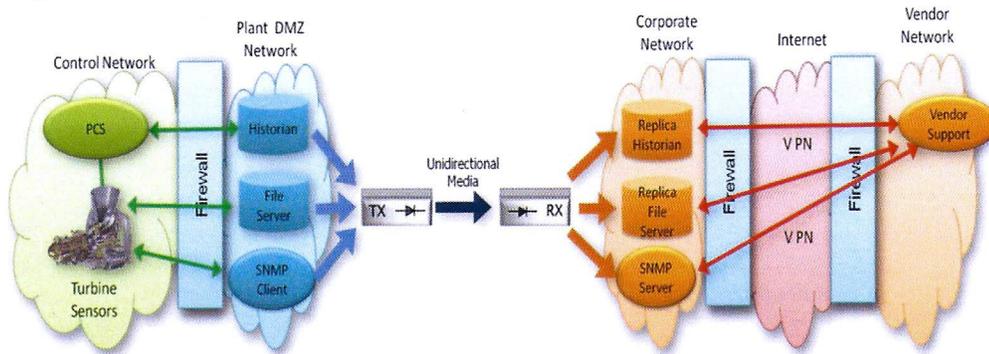
Hardware Data Diode

In March 2001, the UniDirectional Link Routing Working Group of the IETF (UDLR) issued RFC 3077 (<https://tools.ietf.org/html/draft-ietf-udlr-lltunnel-05>) describing a mechanism to emulate full bidirectional connectivity between nodes that are directly connected by a unidirectional link. The following figure illustrates the modern concept of a data-diode.



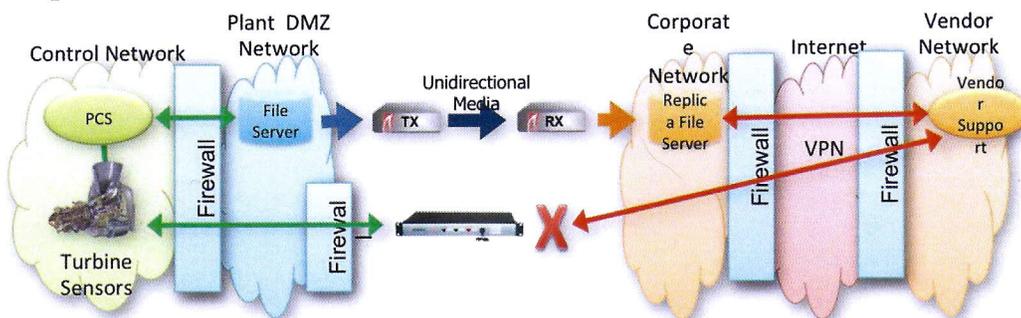
Example 1 below illustrates how a data-diode could be used in an ICS environment using OPC-UA protocol. The data-diode transmit side could emulate an OPC-UA client talking to the server in the ICS network and the receive side could emulate an OPC-UA server talking to clients in the corporate environment.

Example 1



Example 2 illustrates using a data-diode along with a firewall to allow external (i.e., vendor) access to the corporate database replica while protecting the ICS environment.

Example 2



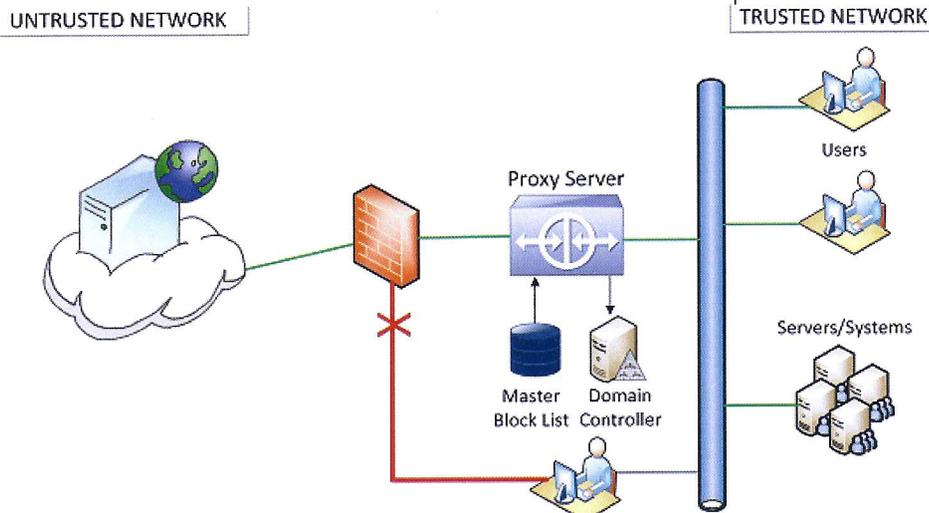
Proxy Servers

Proxies are used to control the data flow between trusted and untrusted networks. They greatly simplify the rules used in a firewall because the traffic is narrowed down to one single IP address that is allowed to access to systems on the other side of the firewall. Proxies can filter access down to the user level, not just the IP level.

There are two basic implementation designs for proxies, forward, and reverse. The most common is the **forward proxy** server. A forward proxy server is commonly used for controlling access to the internet for web browsing. When used in conjunction with caching and content filtering, proxies can provide a useful service to any sized organization.

It can also be used to control SSH, FTP, and other protocols. In this case, the system containing the resources only sees the IP

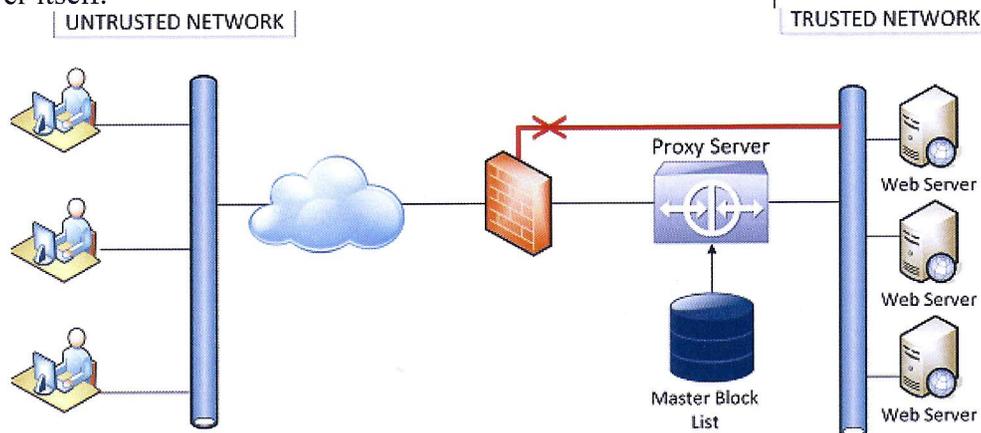
information of the proxy, and **not** the information of the client systems. The following figure illustrates the configuration of a **forward proxy**.



The following points illustrate the implementation of a forward proxy.

- The Active Directory (AD) is used by the proxy to allow access control down to the user and/or AD-group level.
- The BAD file is a black list of sites that are not allowed to be accessed by any of the clients. There could also be a “GOOD” file with special sites for authorized users.
- Direct access to the untrusted network is handled by the firewall. The firewall is configured to **ONLY** allow the proxy to talk to the untrusted network using the specific controlled protocols.

The reverse proxy works in the opposite manner of a forward proxy and proxy's services on behalf of servers in a trusted network. They manage the retrieval of data resources from other systems and returns them to the calling system as though they originated on from the proxy server itself.



Proxy servers can be used for various purposes:

- Allow specific protocol between networks, by system/user.
- To speed up Internet surfing by caching favorite pages.
- To hide the IP address of the client (forward proxy) or server (reverse proxy) computers to maintain anonymity.
- To implement Internet access control like authentication for Internet connection.
- Allow access to specific sites to specific users that are normally blocked websites.
- To scan outbound content, e.g., for data leak protection.

Either implementation can be used for load balancing and caching of resources. Note that “web traffic” (i.e., http, https, or ftp) are not the only protocols that can be governed by a proxy. Depending on what proxy you have implemented, other application protocols (e.g., ssh, etc.) can also be handled.

The most popular open source proxy is Squid. It is full-feature HTTP/1.0 proxy and is very close to being a full-feature HTTP/1.1 proxy. It offers rich access control, authorization, and logging to help develop web proxy and content server applications.

*Views of the Network
from the Host*

Intrusion Detection/Prevention Systems

When most people think of IDSs, they only think about network-based IDSs. There are many types of IDSs that can be used in an ICS environment.

- Host: Sensors reside on the host system.
- Network: What traffic is on your network?
- Application: Web application firewall, database firewall, application protocol IDS.
- Logs: What is happening at the OS level? At the application level?
- Paper: Who came in? Operator’s notebook.
- Anomaly: Any combination of the above.

All methods of intrusion detection involve the gathering and analysis of information from various sources within a computer, network, and enterprise to identify possible threats posed by hackers and crackers inside or outside the organization.

IDSs are not silver bullets. They are more of a warning or audit system. IDSs can also alert to possible misconfigured systems on the network. This situation is not an attack, but it is a problem that could leak important data.

Intrusion Prevention Systems (IPSs) vs. Intrusion Detection Systems (IDSs)

IDSs provide a way to review data of interest. They are connected to a span port and provide passive alerting only. The IDS is used only to watch data.

IPSs can be used to block data. They are placed in line, have an active response, and give passive alerting. When intrusion prevention systems were first introduced, it was feared that they would block critical data packets or add latency to the network. The improved IPS technology has overcome these fears, and we are now seeing IPSs specifically designed and deployed for ICS environments.

Intrusion Detection System

- “Watching”
- Passive Alerting ONLY.

Intrusion Prevention System

- Inline
- Passive Alerting
- Active Response.

Potential Complications

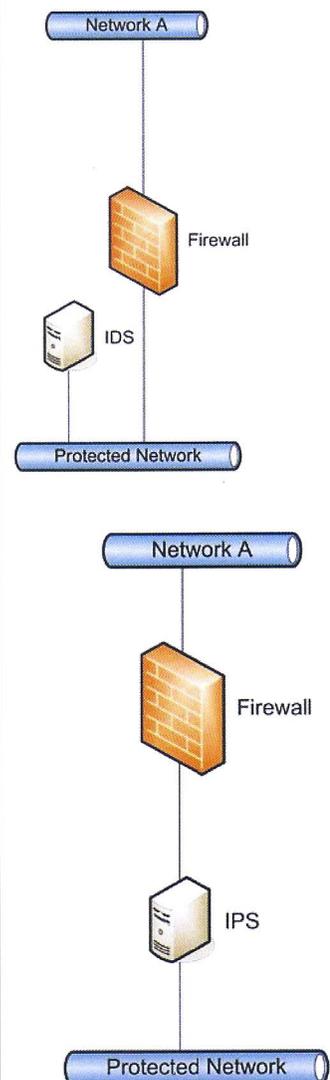
IDSs are technical. A significant amount of work is involved in writing the rules.

IPSs can modify/drop legitimate packets if the rules are written improperly. This could, for example, prevent alarms from reaching the operator stations or configuration changes from reaching their intended destination.

Intrusion Detection Systems

ICS environments provide a unique opportunity. Compared to a corporate environment, an ICS environment is a steady state. Once again, **you must know your environment**. Ask and answer the following questions:

- **WHAT** is normal?
 - You know that host A talks to host B, but not host C.



- **WHEN** does normal become abnormal?
 - Host A is now talking to host C. WHY?
- **WHOSE** applications and services are used and necessary?
- **WHICH** protocols are used?
 - Known IT protocols
 - Vendor proprietary.

Host Intrusion Detection System (HIDS)

Some of the tools installed on a system to protect it and inform the administrator of issues are:

- Virus management
- Local log analysis
- File integrity checking
- Policy monitoring
- Rootkit detection
- Network monitoring (host viewpoint)
- Real-time alerting
- Active response.

AIDE, chkrootkit, Tripwire, AFICK, etc. are some of the file integrity checking tools. Another product is Open-source Host-based Intrusion Detection System (OSSEC), which has a complete menu of sensor/collector options.

The main issue with any HIDS in any environment (especially ICS) is what computer resource penalty you incur in order to get a particular level of security.

Network Intrusion Detection System (NIDS)

Network intrusion detection systems (NIDSs) scan traffic from its networks and look for known patterns in traffic (packets). A NIDS can scan both sides of a conversation and can be reactive by blocking traffic when in IPS mode. NIDS often does not know if the system is Windows, Linux, or a PLC. Simply, traffic is traffic, NIDS reports on what was seen on the network.

NIDS can have a high False-Positive or False-Negative rate based on the information used to generate the signatures.

An IPS generally sits in-line and watches network traffic as the packets flow through it. It acts similarly to IDS by trying to match data in the packets against a signature database or detect anomalies against what is pre-defined as normal traffic. In addition to its IDS functionality, an IPS can do more than log and alert. It can be programmed to react (drop or reject) to what it detects. The ability to react to the detections is what makes IPSs more desirable than IDSs.

There are still some drawbacks to an IPS. IPSs are designed to block certain types of traffic that it can identify as potentially bad traffic. IPSs **do not** have the ability to understand web application protocol logic. At the application layer (OSI Layer 7), IPSs cannot fully distinguish if a request is normal or malformed. This could potentially allow attacks through without detection or prevention; especially newer attacks where signatures are not available (think zero-day).

NOTE: As resources have increased, NIDS vendors have started including capabilities that allow you to describe the systems. This new feature allows you to automatically prioritize an alert based on the possibility of success.

For example, if the NIDS knows that a system is Solaris running Apache and the alert is for an exploit of a Microsoft host running IIS, the priority would be 0.

Web Application Firewall/IDS (WAF)

Web Application Firewalls (WAFs) are designed to protect web applications/servers from web-based attacks that IPSs cannot prevent.

WAFs are typically deployed in some sort of proxy fashion just in front of the web applications, so they do not see all traffic on our networks. WAFs are a special breed that can be used to detect/prevent attacks against web applications in more depth than an IPS.

Unlike IPSs, WAFs interrogate the behavior and logic of what is requested and returned. WAFs protect against web application threats like SQL injection, cross-site scripting, session hijacking, parameter or URL tampering, and buffer overflows.

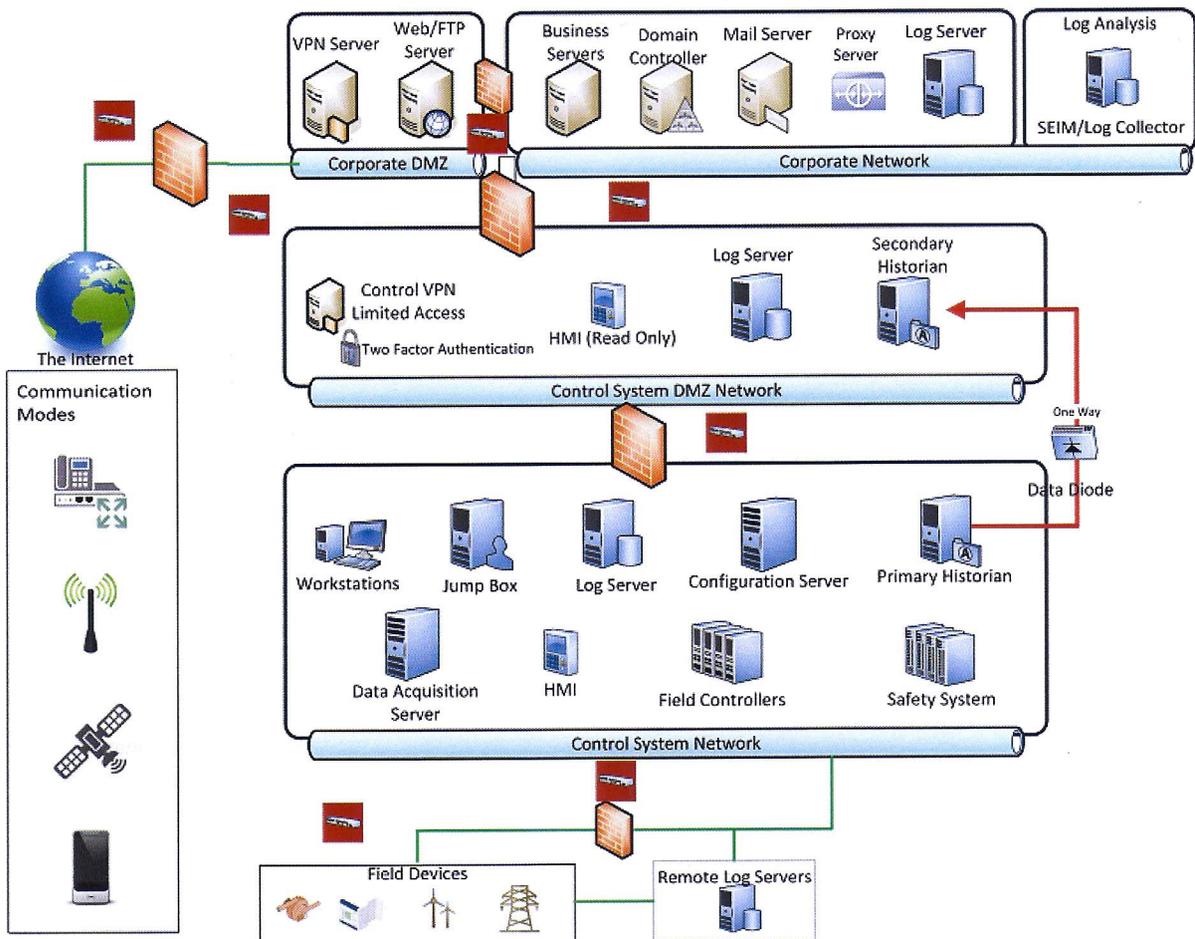
IDS Sensor Placement

The placement for IDS sensors is important. In the image below, the red boxes indicate NIDS and/or IPS deployment locations.

- Any change in trust zones should have an IDS/IPS deployed.
- A data diode should be attached to the historian. The IDS can also be deployed here.
- All points of presence for the external communications should have an IDS/IPS deployed.
- An IDS on either side of firewalls allows you to audit your firewall rules.

See the example on the next page.

You can find a short discussion of some of the OpenSource WAFs available on the web: <http://www.fromdev.com/2011/07/opensource-web-application-firewall-waf.html>



NIDS: Tapping the Network

There are different methods to gather the data off of a network depending on your needs based on amount of traffic, number of monitoring connections, etc.

Port Mirroring, also known as SPAN (Switched Port Analyzer), is a method of monitoring network traffic. When port mirroring is enabled on a managed switch, the switch sends a copy of all the network packets “seen” on one physical port (or an entire VLAN) to another physical port, where the packets can be captured and/or analyzed.

A networking monitoring tap can be used to collect network packets without having to configure a span port on a switch. Think of a tap as a special tee connection that can read data from the network, but not inject any data of its own into the network traffic.

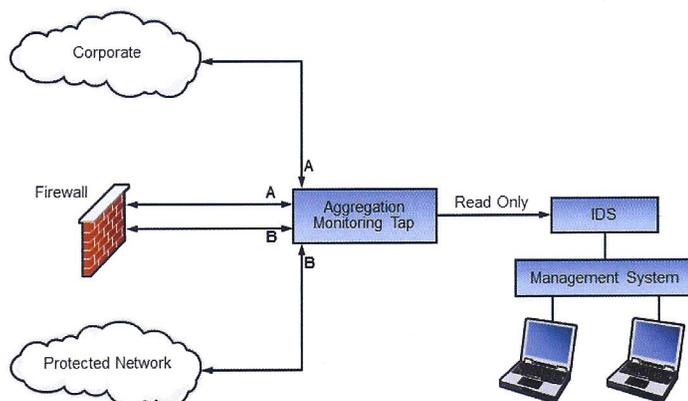
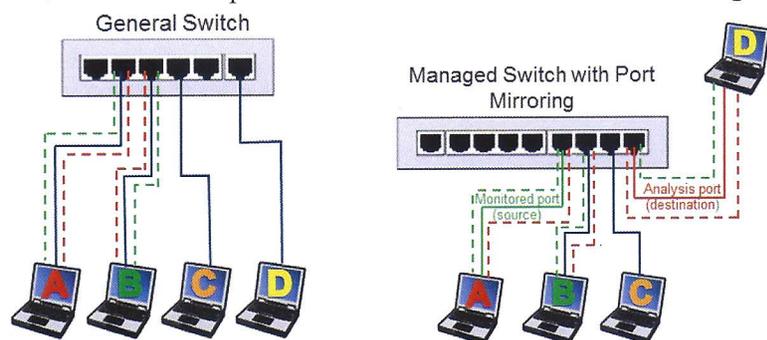
Regeneration Taps are designed to tap one or multiple networks and replicate the traffic to one or multiple monitor ports. Replication of network traffic increases the number of monitor tools that can access any given network. The traffic from the monitored networks

is electrically isolated from each other to prevent any kind of hopping from one to the other.

Aggregation/Distribution Taps allow you the flexibility to use them as either regeneration taps and/or as span distribution/monitoring taps. In this case, one can configure the input span port(s) to go to one/any/all monitoring ports.

Some regeneration/distribution taps provide the option to filter the data streams before sending the data to different monitoring tools. This may be helpful when analyzing the data for different streams. For example, different IDS rule sets might make sense based on the data network.

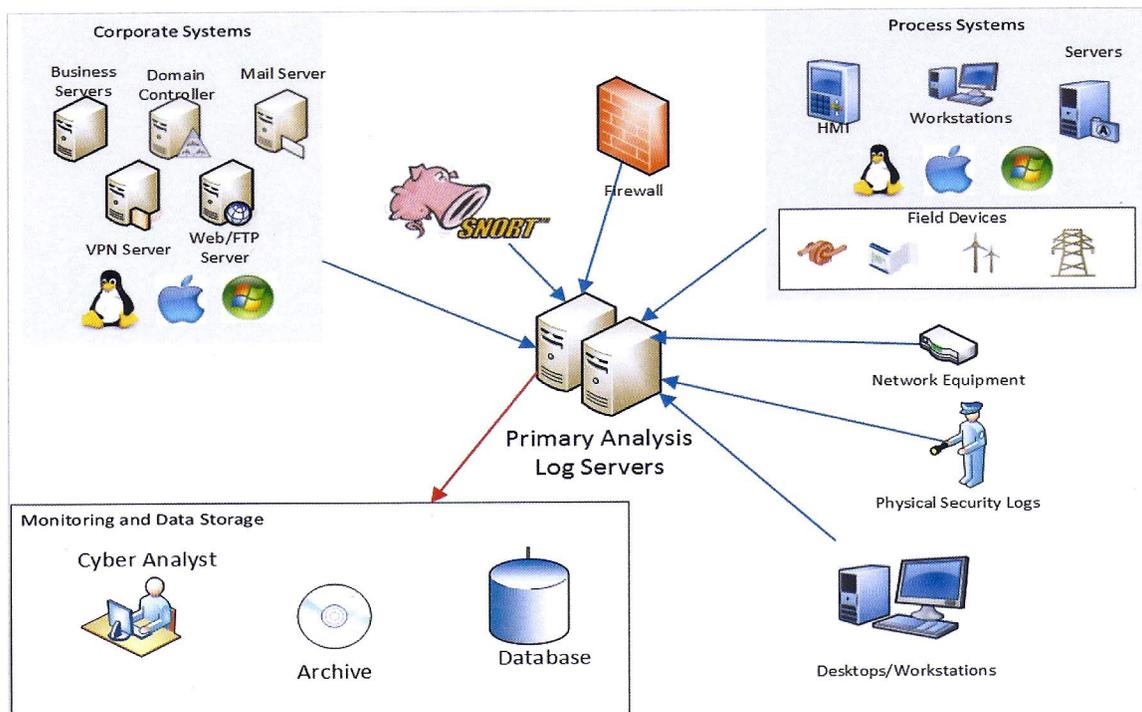
Below is an example of General Switch vs. Port Mirroring.



Logging Architecture

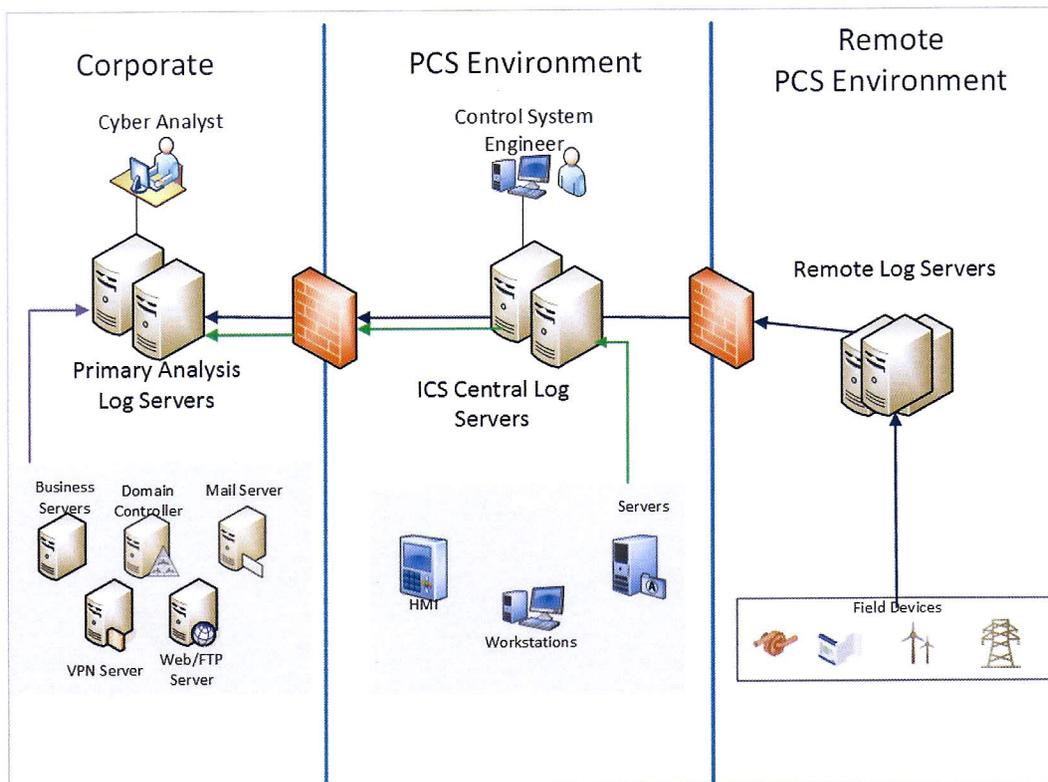
Other types of IDSs are logs of various kinds. A central log server can assist in an incident by providing a chronological list of the events surrounding an incident that give the bigger picture. Multiple systems/sources can send their data to a central log server where it can be correlated with other information.

Correlating with other logs can sometimes make the difference between recognizing an event for what it is (true or false) and then acting accordingly. The same data can provide valuable information (like an IDS) to the security analyst.



There are some considerations in centralizing logs:

1. **Properly prioritize the function of log management.** Define requirements and goals for log performance and monitoring based on applicable laws, regulations, and existing organizational policies. Then, prioritize goals based on balancing the need to reduce risk with the time and resources necessary to perform log management functions.
2. **Create and maintain a secure log management infrastructure.** Identify the needed components and determine how they will interact (e.g. firewall rules, diodes). With the various types of information in one place, the log server becomes a valuable system to target and critical to protect. It should only run the logging service and be in a highly protected area of your network.
3. **Provide appropriate support for staff with log management responsibilities.** All efforts to implement log management will be for naught if the staff members who are tasked with log management responsibilities do not receive adequate training, proper tools, or support to do their jobs effectively. The staff members need to understand what situations are normal, bad, and weird. Providing log management tools, documentation, and technical guidance are all critical for log management staff members to succeed at their jobs.



In the example above, we send our logs in the central control centers and field devices to an ICS log server. This allows control engineers to access the needed data from inside their control environments.

The ICS log server forwards data to the centralized log servers where cyber analysts with cyber security tools can access the data outside of the control system environment.

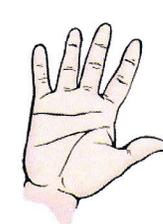
These log servers should be placed in a high security area of the network to limit the chances of compromise.

Network Architecture Exercise

This exercise will allow you to:

- Apply the network security principles discussed in this class into a control system design.
- Utilize a secure model to design a control system with appropriate segmentation.
- Balance the operational and business requirements of a control system with security best practices.
- Consider the architectural challenges IT faces when implementing security measures and methods.

Efficiency vs Security



*"C level Management"
"Security Level Agreements"
between groups*

Introduction

In this exercise we will use Calligra flow, an open source graphics tool for flowcharting and other graphical development (e.g. Visio functionality).

Instructions to access Calligra:

1. In Kali, Open calligra flow from the application launcher in the upper left corner of the screen.
2. Go to applications >> office>> calligra flow
3. Open the Documents/Exercises/ManualCalligraExercise.odg file

Alternatively:

1. Navigate to the Network Diagram file
2. Double click the Documents/Exercises/ManualCalligraExercise.odg file to open its default program.

Step 1 Understanding the Template

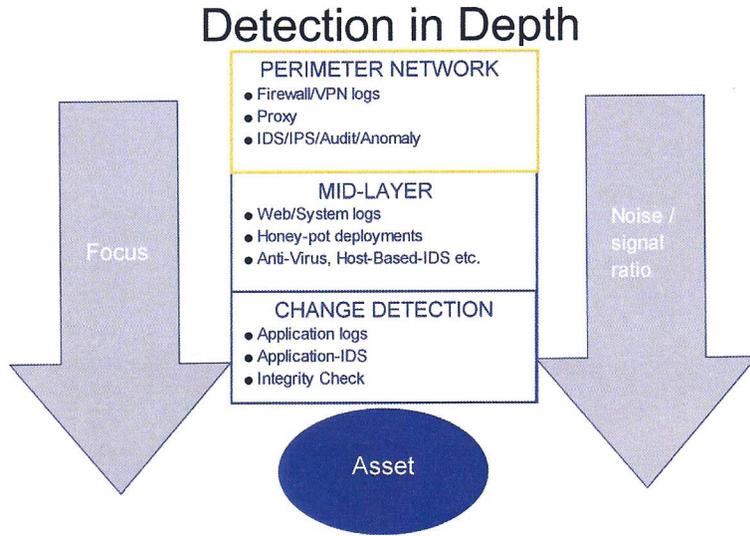
A segmented network model has been developed and layered as a starting point. Consider the objects as puzzle pieces that need to be placed in the segmented model. Each object, including the segmentation zones, can be moved and resized as needed. When objects overlap, the currently selected object will typically be placed on top of the others. Four buttons are available on the right-hand side of the screen to properly layer objects.

Many of the objects are also grouped with labels. These labels can be edited by double clicking them. If the textbox size needs to be adjusted independently from the image, then the group can be split with the ungroup button in the upper right.

The objects from the example diagram are present in the template, but many others are available in the stencil box on the left-hand side of the screen. Outside images can be used to represent objects, but are not required as any needed parts can be represented from the available default list.

To-do: Take some time to move the objects, change zone position, etc., to familiarize yourself with how the program works. Think about what objects are missing that will be needed to implement security measures.

LO14: Describe detection in depth – perimeter network



Philosophy

Logs are just data...

Processed and analyzed, they become *information*

Put another way...

If a tree falls on your network and it shows up in your logs and nobody is reading them - you're still *squished!*

-Marcus J. Ranum

Firewall and VPN Logs

Firewalls, VPN appliances, and Proxy servers provide a wealth of logging information regarding the perimeter of your network. This information can be used to monitor the health of the systems and potentially detect malicious activity.

Log correlations can help locate problems. This information can be used to monitor the health of the system and potentially detect malicious activity. It is important to:

- Look at firewall logs to see what traffic from inside the network is bouncing off the firewall.
- Look for multiple connections from multiple devices in your network to a few target locations.
- Examine VPN logs to find login attempts. Look for unusual situations, such as the president logging in from a Starbucks in England, when he is actually in the middle of a safari in Africa.
- Look for SSH sessions over nonstandard (unusual) ports.

For ensics
 - call ICS CERT
 - memory dump
 - Avoid running ANTIVIRUS
 STEPS:
 Prior to
 Powering down

“If it is not logged, it did not happen!”

- Look for unusual connections from VPN systems to internal systems. VPN systems should not be scanning your internal network.

Proxy Logs

Proxy logs provide valuable information about what your users' Internet activity. It can also show indications of data exfiltration, unknown activity, agents running on systems, etc. By using filters in your log analyzer you can easily pick out suspect activity.

- Identify and trend downloads that caused compromise
- Identify large outbound traffic
- Identify attempts to access blacklisted websites
- Control access to whitelisted sites
- Identify misuse of company resources
- Identify attempts to use unallowed protocols
- Trend user activity

Oct 22, 2012 8:25:26 AM UTC-7:00 tcp://www.google.com/ dilbert 192.168.214.38
 whitelistSearch Engines/Portals 200 Allowed - - 363 1 0 mozilla/4.0
 (compatible; msie 8.0; windows nt 6.1; wow64; trident/4.0; .net4.0c; .net4.0e)

Proxy Log Fields:

Date and Time	Oct 22, 2012 8:25:26 AM UTC-7:00
Url	tcp://www.google.com/
User	dilbert
Client IP	192.168.214.38
Category	whitelistSearch Engines/Portals
Status	200
Verdict	Allowed
Web Application Name	-
Web Application Operation	-
Total Bytes	363
Requests	1
Browse Time	0
User Agent	mozilla/4.0 (compatible; msie 8.0; windows nt 6.1; wow64; trident/4.0; .net4.0c; .net4.0e)
Content Type	-
Malware	-

Netflow Anomaly Detection

NetFlow is a network protocol developed by Cisco Systems for collecting IP traffic information. NetFlow has become an industry standard for traffic monitoring and is supported by platforms other than Cisco. Routers and switches that have the Netflow feature enabled produce UDP data streams that are sent to a Netflow collector (server) where it can be processed and stored. Netflow describes a set of packets sharing the following characteristics:

- Source IP address
- Destination IP address
- Source port for UDP or TCP, 0 for other protocols
- Destination port for UDP or TCP, type and code for ICMP, or 0 for other protocols
- IP protocol
- Ingress interface (SNMP ifIndex)
- IP type of service.

Netflow provides the quantitative view of the network that cannot be provided by signature IDSs.

The router will output a flow record when it determines that the flow is finished. It does this by flow aging. When the router sees new traffic for an existing flow, it resets the aging counter. By analyzing flow data, a picture of traffic flow and traffic volume in a network can be built.

Here is a short list of questions that Netflow data can possibly answer. The blanks are values you have to determine from your knowledge of the environment.

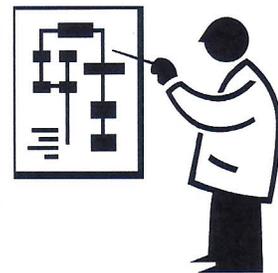
Hosts scanning for services:

- Are there external hosts poking at more than ___ internal addresses?
- Are there external hosts poking at more than ___ ports on 1 (or more) internal hosts?
- Are there unexpected services? – possible Trojan Horses.

Internal infected host scanning/talking to for external hosts:

- Is some internal host poking at ___ external hosts?
- Is some internal host poking at ___ internal hosts?
- Is some internal host poking at dark space (Un-allocated Internet address space)?

Access Squid at:
<http://www.squid-cache.org/Intro/>



Source Port: 46069

There is a complete community of interest in Netflow data. Each year a conference is held in January called FloCon. For more information, visit <http://www.cert.org/flocon/>.

Internal hosts talking to “Interesting Net blocks” (pick your favorite countries here)

- Are there pokes from __net blocks that may be of interest?
- Are there pokes to __net blocks that may be of interest?

Increased network traffic:

- Distributed Denial of Service (**DOS**)
- Unexpected high volume – Data mining, egress?

Using the network data you already have. The spikes you see are from an Nmap scan. You can also create NetFlow data from pcap data. Several software flow tool packages are available, both commercial and open-source. Below is an example of a portscan as seen through Netflow Data.

```
Time, Proto, SrcAddr, Sport, Dir, DstAddr, Dport, TotPkts, TotBytes, State
13:35:27, tcp, 109.162.100.205, 46069, -, 109.162.100.249, ldaps, 2, 114, RST
13:35:27, tcp, 109.162.100.205, 46069, -, 109.162.100.249, ftp, 2, 114, RST
13:35:27, tcp, 109.162.100.205, 46069, >, 109.162.100.249, https, 2, 114, RST
13:35:27, tcp, 109.162.100.205, 46069, -, 109.162.100.249, 3389, 2, 114, RST
13:35:27, tcp, 109.162.100.205, 46069, -, 109.162.100.249, www, 2, 114, RST
13:35:27, tcp, 109.162.100.205, 46069, -, 109.162.100.249, 1723, 2, 114, RST
13:35:27, tcp, 109.162.100.205, 46069, -, 109.162.100.249, ldap, 2, 114, RST
13:35:27, tcp, 109.162.100.205, 46069, -, 109.162.100.249, 127.2, 114, RST
13:35:27, tcp, 109.162.100.205, 46069, -, 109.162.100.249, 2301, 2, 114, RST
```

Intrusion Detection/Prevention Systems - Perimeter

ICS environments provide a unique opportunity. Compared to a corporate environment, an ICS environment is a steady state. Once again, **you must know your environment**. Ask and answer the following questions:

- **WHAT** is normal?
 - You know that host A talks to host B, but not host C.
- **WHEN** does normal become abnormal?
 - Host A is now talking to host C. **WHY?**
- **WHOSE** applications and services are used and necessary?
- **WHICH** protocols are used?
 - Known IT protocols
 - Vendor proprietary.

Expectations for All IDS

An IDS is not a cure-all for network security problems. It is an alerting tool to let you know something has happened. An IDS can do the following:

- Detect activity that are precursors to real attacks
- Alert for active attacks
- Provide post-attack analysis
- Provide “situational awareness”
- Provide intrusion post mortem
- Help develop knowledge of typical behavior.

Placing an IDS outside of the firewall can be helpful for situational awareness and forewarning of activities. The IDS can detect scanning or other precursory attack activities that might be dropped by the firewall.

Signature vs. Anomaly Detection

Below is a quick comparison between signature-based detection and anomaly-based detection methods. Both methodologies have strong and weak points. It depends on what your end goal is as to which method will work best.

Signature	Anomaly
Watches for specific events	Watches for changes in trends
Only looks for what it has been told	Learns from gradual changes
Can deal with any known threat	Can deal with unknowns, but any attack is subject to false-negative
Unaware of network configuration changes	Sensitive to changes in network devices
Highly objective inspection	Subjective, prone to misinterpretations
Predictable behavior	Unpredictable behavior
Easy to tune manually	Must trust the system completely

Snort Rule Dissection

Snort is an open-source network intrusion detection and prevention system. Snort is widely used and has become the standard for IDS/IPS.

Learning to write Snort rules is useful because most IDS/IPS applications will either use the Snort rule format or provide a way to import Snort rules.

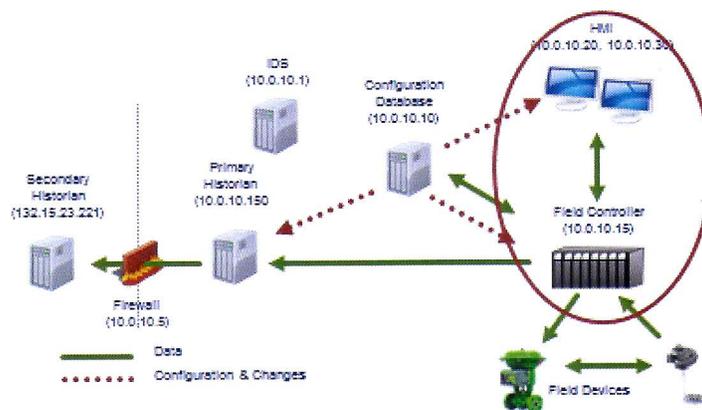
If you are able to understand the data flow in your environment, you will be able to design simple anomalous traffic signatures quickly without regard to the actual details of the protocol used.

Marcus's Laws of Logging and IDS

1. Never collect more data than you can conceive of possibly using.
2. The number of times an uninteresting thing happens is an interesting thing.
3. Collect everything you can except for when you come in conflict with the first law.
4. It doesn't matter how real-time your IDS/logging system is if you don't have real-time administrators/analysts.



Note: There is a tool called Sagan which structure and rules work similarly to the Snort IDS/IPS engine. This was intentionally done to maintain compatibility with rule management software and allows Sagan to correlate log events with your Snort IDS/IPS system. Sagan can also write to Snort IDS/IPS databases via Unified2/Barnyard2.



Snort rules are composed of a rule header and rule options. There are 5 types of rule options:

- Metadata
- Payload detection
- Non-payload detection
- Post-detection
- Thresholding and suppression.

For the purposes of this class we will focus on Metadata and payload detection.

Snort Rule Headers

Snort Rule Example

```

alert ip [10.0.10.20,10.0.10.30] any <> !10.0.10.15 any \
(msg:"ALERT-Field Controller interact with another node"; \
sid:3000001; priority:1; rev:1;
content:"cmd.exe"; tag: session,256,packets;)
  
```

Note: Snort rules are written to be on one line. However, a backward slash “\” can be used to divide long rules into multiple lines.

Snort Rule Headers

<code>alert ip [10.0.10.20,10.0.10.30] any <> !10.0.10.15 any \</code>	
<code>(msg:"ALERT-Field Controller interact with another node"; sid:3000001; \</code>	
<code>priority:1; rev:1;content:"cmd.exe"; tag: session,256,packets;)</code>	
action	alert, log, pass, active, dynamic, or a custom defined type
protocol	ip, tcp, udp, icmp, any
src ip and src port	See below
direction	->, <> direction of the traffic that the rule applies to
dst ip and dst port	See below

Source and Destination OP addresses can be defined as:

- A predefined variable (e.g., \$HOMENET)
- Complete IP address
- Network using CIDR notation
- A list of IP addresses and/or networks [ip1, ip2, ...]
- The keyword **any**.

The “!” can be used as a negation operator to tell Snort to match on any IP address except the one(s) indicated. If you use “!” you must enclose the IP addresses in brackets, for example ![1.2.3.1, 192.168.1.0/24].

When you code a list of addresses they must be comma separated WITHOUT any spaces.

Src and Destination Port numbers can be:

- Individual port numbers
- Port variable (\$HTTP_PORT)
- Port ranges (“135:139”, “:1024”, “1025:”)

Snort Rule Metadata Options

```
alert ip [10.0.10.20,10.0.10.30] any <> !10.0.10.15 any \  
(msg:"ALERT-Field Controller interacts with another node"; sid:3000001; \  
priority:1; rev:1;content:"cmd.exe"; tag: session,256,packets;)
```

<i>msg:description</i>	specifies the human-readable alert message seen by the analyst
<i>sid:number</i>	Signature ID – a unique identifier for the rule
<i>priority:number</i>	Level of urgency on scale of 1 – 10. 1 being the most urgent.
<i>rev:number</i>	Revision number of the rule. All rules start at rev:1.
<i>reference:url,file name</i>	URL or file with information about the alert and rule

The rule options are used to identify the rule for both computer and human “consumption” and provide the analyst with different attributes about the particular network traffic he wants the rule to act on. The only required option in all rules is the Signature ID (**sid:number**). While the message (**msg:text**) may not be required, it is necessary to help the human comprehend why the rule was written in the first place.

Sid is used to uniquely identify rules. This information allows output plugins to identify rules easily. This option should be used with the rev keyword. The value used for the sid falls into one of these ranges:

- <100 Reserved for future use
- 100-1000000 Rules included with the Snort distribution

NOTE: A UNIQUE sid is REQUIRED FOR EACH RULE!

If rule is new version, use the same sid and just add [or increment] revision number (rev:number).



- 1000000 to 2000000 Reserved for Community Rules
- 2000001 to 3000000 Emerging Threat Rules
- 3,000,001 + Used for local rules

The revision number (rev:number) is used to keep track of older rules. The latest revision is the one that is operational. Older rules may need to be re-deployed if the situation warrants it. The revision in the example is 1. If we add more information to the rule, we would make the new rule rev:2;

Payload Options

```
alert ip [10.0.10.20,10.0.10.30] any <> !10.0.10.15 any \
(msg:"ALERT-Field Controller interacts with another node"; sid:3000001; \
priority:1; rev:1; content:"cmd.exe"; tag: session,256,packets;)
```

content: "ASCII String" or content: "[HEX]"	searches for specific data within the payload. The content can be defined as either ASCII or HEX.
uricontent: "ASCII String" or uricontent: "[HEX]"	only looks for content within the Uniform Resource Identifier (URI) of a web request.
pcr: "regular-expression";	Allows use of Perl compatible regular expressions (see www.pcre.org for info)
tag: type, count, metric	Capture of a specific number of packets after alerting.

Content Example

Multiple **contents** can be included in a rule; however contents are searched in the order listed.

```
content:"/cmd.exe";
content:"|2f 63 6d 64 2e 65 78 65|";
```

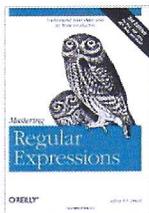
Uricontent Example

In 2000-2001, a new type of attack emerged that took advantage of Unicode and its original poor implementation and lack of any use in IDS engines. These worms were Red Worm, Red Worm II, and Nimda worm – that exploited Unicode vulnerabilities in the IIS server in order to achieve phenomenal growth. Because of the increased use of Unicode, Snort has an http_inspect pre-processor that helps to “normalize” any Unicode so ASCII content matching can occur.

Nimda and Code Red also proved that general full packet analysis code results in an overabundance of false-positives. The w199eb attacker sent requests with the string: ..\..\system32\cmd.exe and all the security bulletins were telling the analyst to look for ..\..\system32\cmd.exe. As a result, the IDS alerted on both of these strings. This led to the implementation of an option for just checking the actual browser request string [GET http://...].

For more information on the regular expressions, visit <http://www.regular-expressions.info/>. They offer a good tutorial for those just getting started.

You also look for Jeffrey Friedl’s “Mastering Regular Expressions.”



uricontent:"cmd.exe";

uricontent:"|2f 63 6d 64 2e 65 78 65|"

PCRE Example

Perl Compatible Regular Expression (PCRE) was inspired by the regular expression capabilities in the Perl programming language.

The PCRE syntax is much more powerful and flexible than either of the POSIX regular expressions and many classic regular expression libraries. The name is misleading though because PCRE and Perl each have capabilities not shared by the other.

When using PCRE options, be sure to precede them with a content option. PCRE evaluations are resource intense. There is no sense to fire it up if the basic content is not there. A poorly written PCRE rule can cause throughput issues that result in IDS dropping packets or possibly blocking traffic in an IPS setting. Be sure to test complex PCRE rules for accuracy and for throughput.

Content:"XHX"; pcre:"/^UAIIA\s+XHX\s+YANER/smi":

Tag Example

Sometimes having an alert fire is just the tip of the iceberg. What you really want to see is what happened during the session after the packet that triggered the alert. If you are doing full packet capture you can scan through the tcpdump files to find your answer. Usually this is not totally practical. Snort has the capability to "tag" the packets in a session containing the packet that alerted. All the associated packets will be written to the pcap logging file `snort.log.nnnnnnnnnn` by default, where `nnnnnnnnnn` is the UNIX EPOCH seconds since January 1, 1970 00:00:00.

The format for the option is: **tag: *type*, *count*, *metric***;

- Where *type* is either **host** or **session** depending on whether you want just the data for the just host, or the total session
- *count* is the maximum number of *metric* units to dump
- *metric* is either packets, seconds, or bytes.

Using the **tag** option gives you the ability to capture the additional packets without having to do a continuous full packet capture.

In this example on page 185, the rule will capture 256 packets after alerting.

Consequential Effects

Consequential effects can affect what you want to capture and what you actually capture. There are **two ways to write alert rules**, depending on the traffic you want to capture. Both are correct and are used based on your needs.



The first alert rule is:

- alert tcp any any <> x.x.x.x 80 (sid:3000001; msg:"traffic");

This alerts on all packets to OR from x.x.x.x 80 to any host.

The second alert rule is:

- alert tcp any any <> x.x.x.x 80 (sid:3000001;msg:"traffic"; content:"cmd.exe");

This rule alerts ONLY for those packets containing the string "cmd.exe."

Snort Preprocessors for ICS

A number of attacks cannot be detected by signature matching alone in the detection engine, so protocol "examine" preprocessors step up to the plate and detect suspicious activity. These preprocessors include packet fragmentation, TCP stateful inspection, portscans, and many other Network/Application protocol specific activities.

Others modify packets by normalizing traffic so that the detection engine can accurately match signatures. These preprocessors defeat attacks that attempt to evade Snort's detection engine by manipulating traffic patterns.

Snort cycles packets through every preprocessor to discover attacks that require more than one preprocessor to detect them. If Snort simply quit checking for the suspicious attributes of a packet after it had set off a preprocessor alert, attackers could use this deficiency to hide traffic from Snort.

Preprocessor parameters are configured and tuned via the snort.conf file. The same snort.conf file lets you add or remove preprocessors as you see fit. Of particular interest to the ICS community are the DNP3 and Modbus preprocessors. Consult the Snort documentation for more information on the preprocessors.

Arpspoof	map	sdf
bo (BackOrifice)	perfmonitor.	sip
dcerp2	pop	smtp
Dns	portscan	ssh
Frag3	reputation (ipv6)	ssl
ftptelnet	rpc_decode	stream5
httpinspect	rzv_saac	

DNP3

Modbus

DNP3 Preprocessor Rule Options

The new features in Snort Version 2.9.2 are:

- dnp3_func - Matches Function Code inside an Application-Layer request/response header

lot a snort rules

- dnp3_ind - Matches on the Internal Indicators flags in Application Response Header (Similar to TCP flags)
- dnp3_obj - Matches on request or response object headers
- dnp3_data - Reassembled Application-Layer Fragments.

DNP3 Preprocessor

Here are some examples of the new DNP3 preprocessor rule options:

Alerts on DNP3 Write Request:

```
alert tcp any any -> any 20000 (msg:"DNP3 Write request"; \
dnp3_func:write; sid:3000001;)
```

Alerts on reserved_1 OR reserved_2 being set:

```
alert tcp any 20000 -> any any (msg:"Reserved DNP3 Indicator \
set"; dnp3_ind:reserved_1,reserved_2; sid:3000002)
```

Alerts on Content in Re-assembled Application-Layer Fragment:

```
alert tcp any any -> any any (msg:"badstuff' in DNP3 message"; \
dnp3_data; content:"badstuff"; sid:3000003;).
```

Notice in the third rule, dnp3_data sets the content buffer to the beginning of the Re-assembled Application-Layer Fragment then looks for the content: "badstuff."

Modbus Preprocessor Rule Options

These new Modbus preprocessor rule options are similar to the functionality of the DNP3 preprocessor, which makes writing rules for these two ICS protocols easier and more accurate.

- modbus_func - Matches against the Function Code inside of a Modbus Application-Layer request/response header
- modbus_unit - Matches against the Unit ID field in a Modbus header
- modbus_data - Sets the cursor at the beginning of the Data field in Modbus request/response.

Some examples are:

Alerts on specific Modbus function:

```
alert tcp any any -> any 502 (msg:"Modbus Write Coils request"; \
modbus_func:write_multiple_coils; sid:3000004;)
```

Alerts on unauthorized host var MODBUS_ADMIN

```
192.168.1.2 alert tcp !$MODBUS_ADMIN any -> any 502 \
(msg:"Modbus command to Unit 01 from unauthorized host"; \
modbus_unit:1; sid:3000005;)
```

Alerts on Content in modbus data field

```
alert tcp any any -> any any (msg:"String 'badstuff' in Modbus \
message"; modbus_data; content:"badstuff"; sid:3000006;).
```

The second process is a product called barnyard2 and is available from:

<http://www.securixlive.com/barnyard2/index.php>

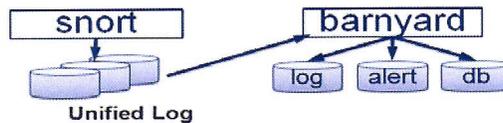
Snort Output Plugins

Alerts:

arubaaction	full	socket	test
Fast	prelude	syslog	unixsock

Logs:

ascii	database	tcpdump	unified2
csv	null		



Most commonly used output plugins:

alert_fast - This will print Snort alerts in a quick one-line format to a specified output file. It is a faster alerting method than full alerts because it doesn't need to print all of the packet headers to the output file and because it logs to only one file.

alert_full - This will print Snort alert messages with full packet headers. The alerts will be written in the default logging directory (/var/log/snort) or in the logging directory specified at the command line. Inside the logging directory, a directory will be created per IP. These files will be decoded packet dumps of the packets that triggered the alerts. The creation of these files slows Snort down considerably. This output method is discouraged for all but the lightest traffic situations.

alert_syslog - This module sends alerts to the syslog facility. This module also allows the user to specify the logging facility and priority within the Snort config file, giving users greater flexibility in logging alerts.

log_tcpdump - The log_tcpdump module logs packets to a tcpdump-formatted file. This is useful for performing post-process analysis on collected traffic with the vast number of tools that are available for examining tcpdump-formatted files.

csv - The csv output plugin allows alert data to be written in a format easily importable to a database. The output fields and their order may be customized.

database - The database output plugin logs directly to a relational database of your choice. It supports MySQL, PostgreSQL, Oracle, and UnixODBC-compliant databases such as SAPdb. Outputting to a relational database makes large amounts of intrusion data accessible. When the database plugin is placed into a database, alerts can be sorted, searched for, and prioritized in an organized manner.

unified2 - Unified2 can work in one of three modes, packet logging, alert logging, or true unified logging. Packet logging includes a capture of the entire packet and is specified with `log_unified2`. Likewise, alert logging will only log events and is specified with `alert_unified2`. To include both logging styles in a single, unified file, simply specify `unified2`.

Resources for ICS Vulnerability and IDS Signature Information

- ICS-CERT
 - ICS-CERT Portal account for email notifications of new ICS vulnerabilities
- OSVDB - <http://osvdb.org/>
 - Search for SCADA vulnerabilities
 - Set up “watch” to receive email notifications about new vulnerabilities of interest. Set up “watch” by product.
- NIST – <http://nvd.nist.gov/home.cfm>
 - National Vulnerability Database
- Emerging Threats Pro – <http://emergingthreats.net>

Suricata [<http://suricata-ids.org/>] is a high performance Network IDS, IPS, and Network Security Monitoring engine. Open Source and owned by a community run nonprofit foundation, the Open Information Security Foundation (OISF <http://suricata-ids.org/about/oisf/>).

Suricata is developed by the OISF and its supporting vendors. Suricata is multi threaded. This means you can run one instance and it will balance the load of processing across every processor on a sensor Suricata is configured to use. This allows commodity hardware to achieve 10 gigabit speeds. Most common application protocols are automatically recognized allowing for protocol recognition on different TCP/IP ports.

Example Rules

```
ipvar CANARY 1.2.3.4
ipvar PCS [192.168.0.0/24]
ipvar CORP [1.2.3.0/24]
ipvar IP_PLC 192.168.0.97
ipvar AD 1.2.3.20
ipvar HMI [10.10.10.20,10.10.10.30]
ipvar FC [10.10.10.15]
ipvar HOME_NET [1.2.3.0/24,192.168.0.0/24]
ipvar EXTERNAL_NET [!HOME_NET]
```



Network Defense, Detection, and Analysis Hands-on Exercises



This exercise will allow you to:

- Test and run Snort from a command line
- Create and modify Snort rules to detect an ICS exploit based on information in an ICS-CERT alert.
- Create a Snort rule to detect cmd.exe in pcap traffic.
- Use Wireshark to examine the traffic that was alerted on and determine the bad actors.

Exercise 1: Using and Testing Snort

The basis of Snort is the rules files. Without these, Snort will not know what to detect. For this exercise you will be modifying, testing, and running existing Snort rules. Then, evaluating the alerts that are generated.

All rules you write will be in a file called **local.rules** located in **/etc/snort/rules**. Let's begin.

1. Using an editor of your choice (vi – command line, gedit – gui based), open the local.rules file - **/etc/snort/rules/local.rules**
2. There will be two rules in the file. See if you can understand what each rule is trying to do. Then, save and close the file.
3. Testing Snort Rules
 - a. Open a command prompt.
 - b. Type the following command:

memo

```
snort -c /etc/snort/snort.conf -T
```

- c. If everything runs okay, you will get a message similar to this screen.

```
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Preprocessor Object: SF_POP Version 1.0 <Build 1>

Snort successfully validated the configuration!
Snort exiting
root@kali:~/Desktop#
```

4. Running Snort
 - a. The command for running snort is - snort -r ~/Desktop/pcap_files/filename
 - b. For example purposes we will use the pcap file scada.pcap
 - c. Type the following command at the prompt

```
snort -c /etc/snort/snort.conf -r ~/Desktop/pcap_files/scada.pcap
```

- d. If you look on your Desktop, you will see four files. Open each of these and look at what they tell you. These are the logs that contain the alerts.
- `snort_alerts.csv`
 - `snort_alerts_fast.txt`
 - `snort_alerts_full.txt`
 - `tcpdump.log.nnnnnnnn` (where `nnnnnnnn` = the epoch timestamp of the alert)

NOTE: Before each exercise run the cleanup script (cleanup.sh) located on the desktop.

Exercise 2: Using Snort to Detect an ICS Exploit (Deep Packet Inspection)

Scenario: During your quick daily review of ICS related RSS feeds and mailing lists, you notice that ICS-CERT has released an advisory about an INLDEMO Tag Server Buffer Overflow Vulnerability. You use this tag server at your plant and you contact the vendor for patch information. The INLDEMO vendor does not have a patch available, but Metasploit has exploit code available for the vulnerability.

Use the information provided in the advisory to write an IDS signature(s) to detect this vulnerability. You will use the full capture file `tagserverBad.pcap`, found in `/root/Desktop/pcap_files`, to test your files.

ICS-CERT ADVISORY

ICSA-xx-xx-xx—INLDEMO Tag Server Buffer Overflow

March 18, 2015

OVERVIEW

A Metasploit module has been released for the INLDEMO Tag Server buffer overflow.

AFFECTED PRODUCTS

INLDEMO Tag Server 0.0.0

IMPACT

The impact of the vulnerability could range from a DOS to system level access on the tag server.

VULNERABILITY DETAILS

The tag server code does not apply proper boundary checking. If a "SYNC" packet is sent with a "large" payload the overflow will occur.

EXPLOITABILITY

This vulnerability is exploitable from a machine on the network by sending a "large" SYNC packet to the tag server's listening port (2000/TCP).

EXISTENCE OF EXPLOIT

Publicly available exploits are known to exist for this vulnerability.

DIFFICULTY

Once the malcontent reaches a computer associated with tag sever, the buffer overflow is moderate/low difficulty.

ICS-CERT CONTACT

For any questions related to this report, please contact ICS-CERT at:

E-mail: ics-cert@dhs.gov

Toll Free: 1-877-776-7585

For Control System Security Program Information and Incident Reporting: www.ics-cert.org

A. Write a Short rule using information from the ICS bulletin.

From the ICS-CERT bulletin, we know the following information:

- The tag server traffic port is 2000
 - The protocol used is TCP
 - The tag server payload (content) we are interested in contains "SYNC"
 - The alert message should be "INLDemo TAG Server buffer overflow"
1. Use the information from the ICS-CERT bulletin to modify the first snort rule in **local.rules** to reflect the known information. NOTE: The rules are on one line.

```
alert tcp any any -> any 2000 ( sid:3000001; msg:"INLDEMO Tag Server Buffer Overflow"; content:"SYNC"; reference:url,localhost/rules_doc/3000001; )
```



2. Test your new rule. Then, run this rule in snort using the pcap file *tagserverBad.pcap* located in /root/Desktop/pcap_files.

```
snort -c /etc/snort/snort.conf -T
snort -c /etc/snort/snort.conf -r ~/Desktop/pcap_files/tagserverBad.pcap
```

3. Review the alert files created on your desktop.
 - What can you learn for the alert files?
 - What is the IP of the attacker?
 - What is the IP of the tag server?

NOTE: Before each exercise run the cleanup script (cleanup.sh) located on the desktop.

Exercise 3: Using Snort to find a cmd.exe compromise

Scenario: Based on information from your forensics team, a computer on your network has been compromised and the attacker used the command line. It is your job to find the ip address of the attacker.

1. Edit the *local.rules* file. Place a hash mark at the beginning of the first rule so it looks like this.
#alert tcp any any -> any 2000...
2. Look at the second rule in the file. Is it written to detect “cmd.exe” on the network?
3. Save and close the local.rules file. Test then run snort using the pcap file *msfconsoleexercise.pcap*.

```
snort -c /etc/snort/snort.conf -T
snort -c /etc/snort/snort.conf -r ~/Desktop/pcap_files/msfconsoleexercise.pcap
```

4. Examine the alert files on your Desktop for the alert generated by snort.
 - How many alerts were generated?
 - What IP addresses were involved?
 - What ports were used?
5. Now to see what happened after the attacker sent the “cmd.exe” across the network, you need to capture the sessions after the alert. To do that we use the **tag** option. Once again, edit the **local.rules** file. Add “**tag:sessions,256,packets**” to the end of the rule. Your rule should look like this:

