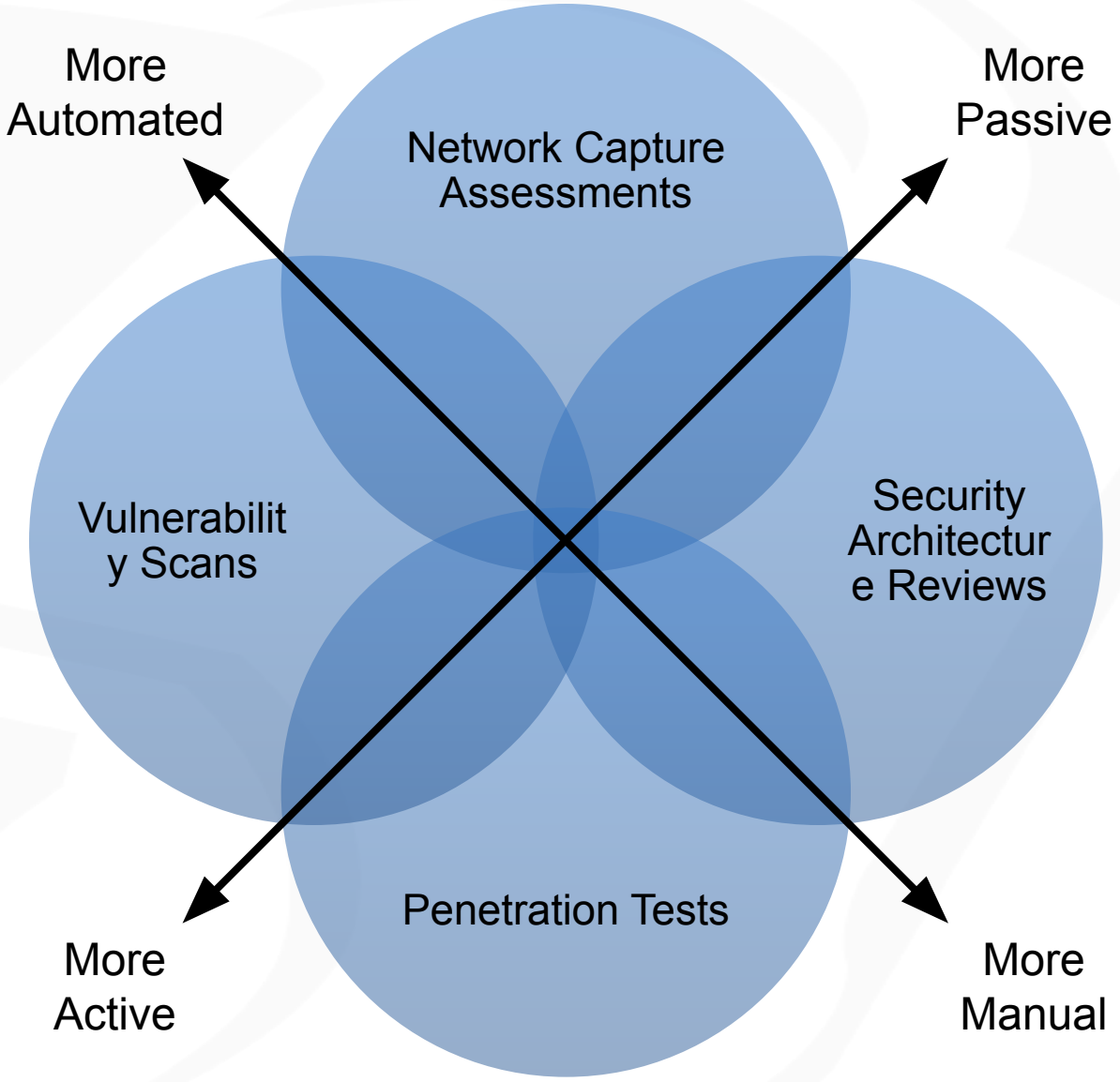
Assessing and Exploiting Production Networks

Version 38

801-784-2052 // justin@controlthings.io

Types of Security Assessments

- We can perform many different types of security assessments to discover vulnerabilities in our systems and weaknesses in our defenses
- Each assessment type fills looks at the system from different perspectives and angles
- All types should be performed to gain a more complete picture
  - Some vulnerabilities might only be found using one type
  - Some tests increase system risk for increased visibility
  - Each type can be adapted to system and company needs



Start with an Security Architecture Review

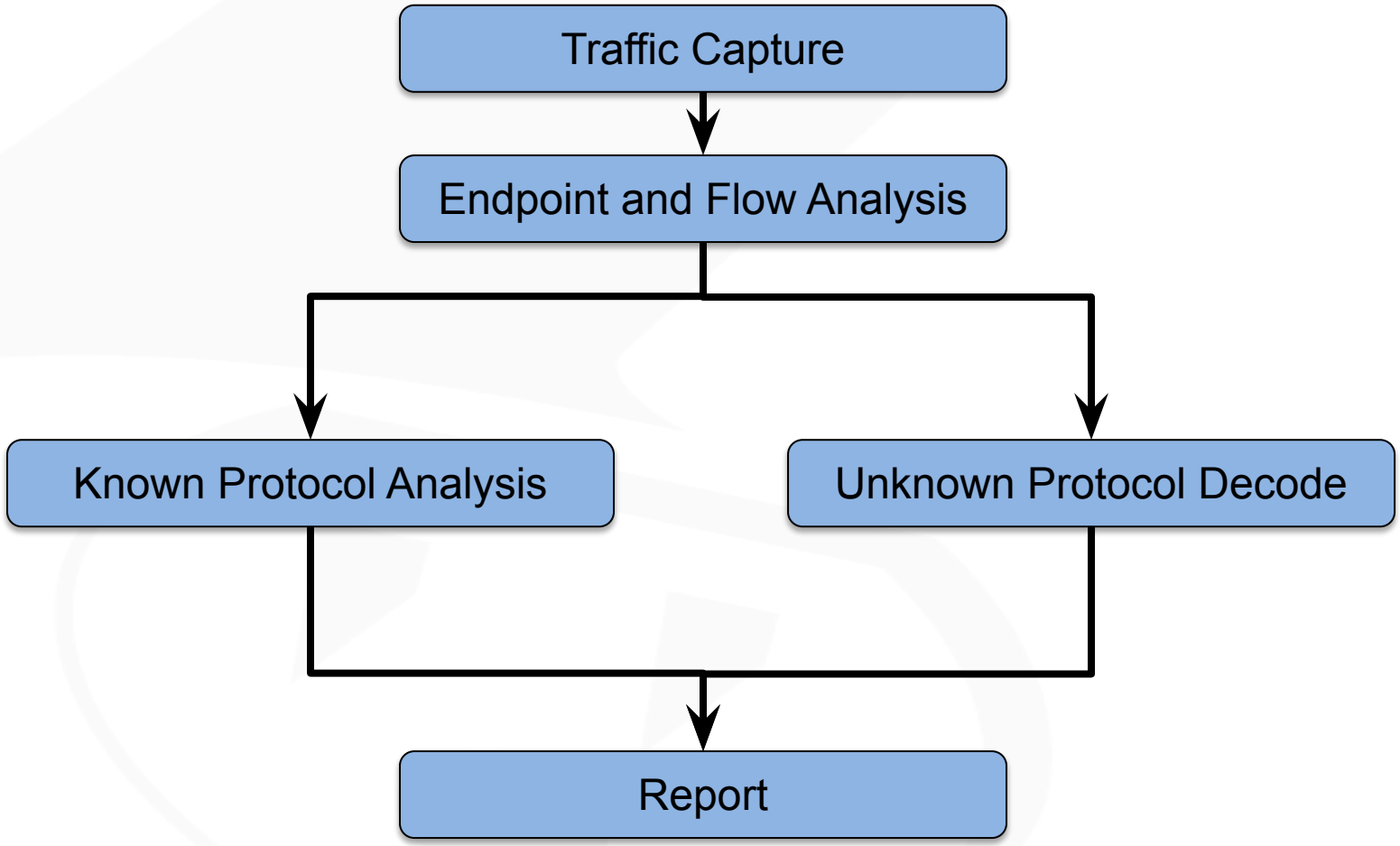
- **Goals**: gain context for testing and identify probable weaknesses in the system to target first
- Review of system documentation
- Perform interviews with key persons
  - Vendor security engineers
  - Architecture teams
  - System Administrators
  - Users of the Systems
- Demonstration of system functionality
  - Do it onsite, via webex, or take a CBT if available
- Review of system configurations
- Review of all other devices that make up the system

Other Information Gathering

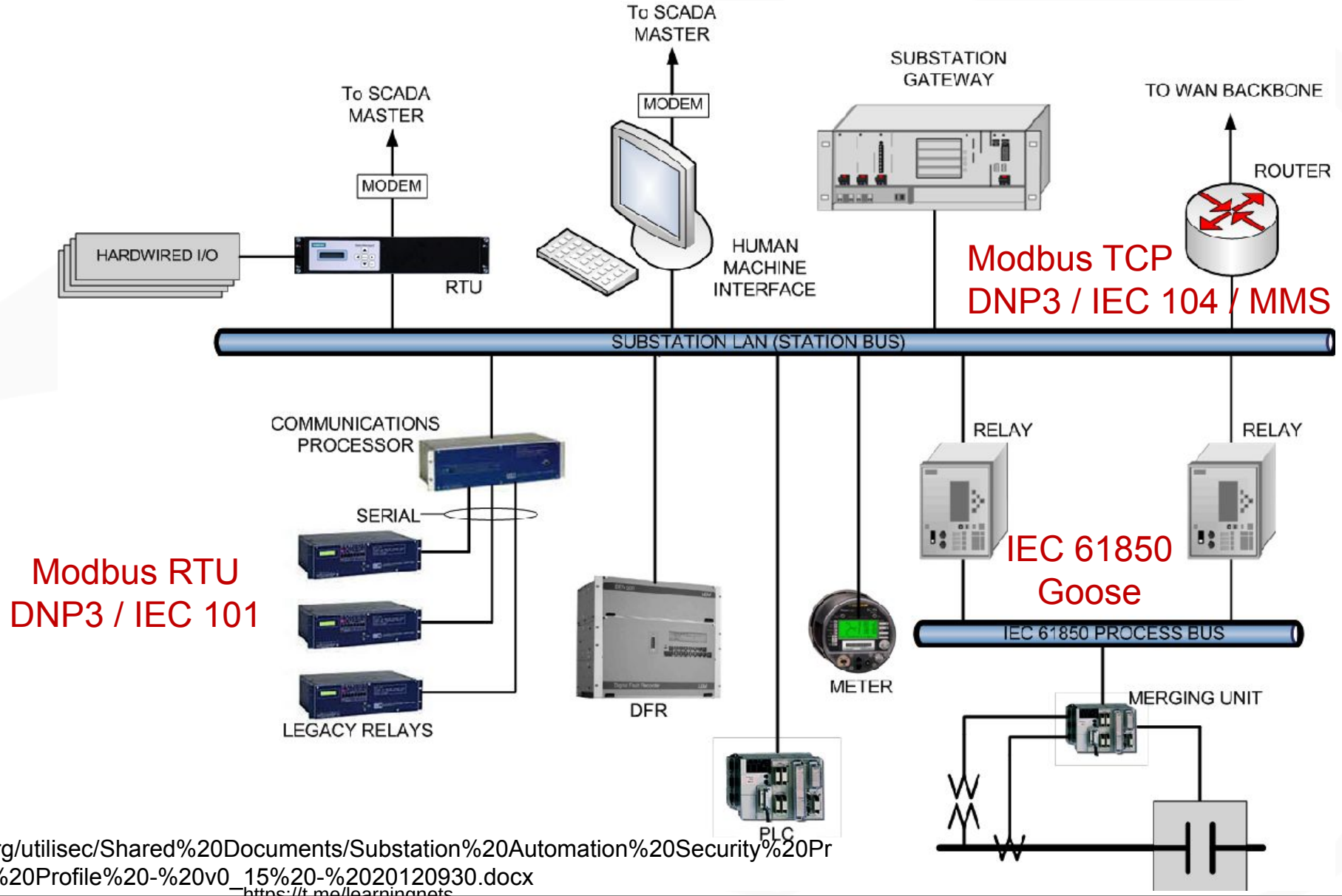
- Company Information Gathering
  - Website provides details about locations, personnel, technologies, and partnerships
  - News Websites
  - Google
- Personnel Information Gathering
  - Social Websites: Linked In, Google+, Facebook, Technology Forums
  - Search Tools: Recon-NG, Maltego, Google
- Technology Information Gathering
  - Vendor Websites
  - Google
  - Shodan

Passive Network Capture Assessment

Network Capture Assessments



Example: Substation Network



Source - http://osgug.ucaiug.org/utilisec/Shared%20Documents/Substation%20Automation%20Security%20Profile/SA%20Security%20Profile%20-%20v0_15%20-%2020120930.docx

Task 1: Control Traffic Capture

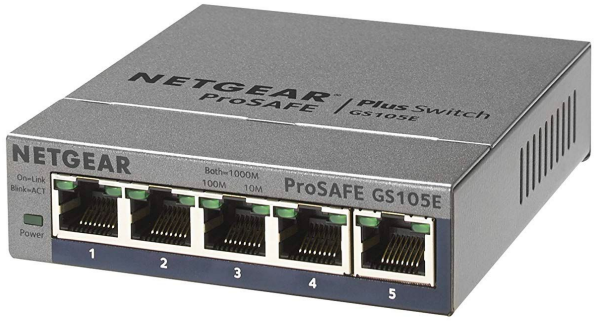
Level of Effort: Low

Task Description: Use a tool to capture sample communications. Attempt to cause known actions that result in communications between devices, such as firmware updates, and capture this communication individually to facilitate later analysis. Obtain samples of all target functionality.

Task Goal: Obtain data for the following tasks.

Network Capture Tools

- Ethernet
  - Thanks to the IT world, capturing Ethernet is easy
  - Hardware:
    - SPAN port on existing switches
    - Active or passive tap
    - Devices like Hak5 Packet Squirrel and Plunder Bug
    - Ethernet port on existing machine
    - Dual ethernet ports on your laptop



  - Graphical Software:
    - `Wireshark`
  - Commandline Software: (all three examples show new files every 100MB or 1 hour)
    - `tcpdump -i eth0 -w file.pcap -C 100 -G 3600 -n -K`
    - `tshark -i eth0 -w file.pcap -b filesize:100000 -b interval:3600 -n`
    - `daemonlogger -i eth0 -n file.pcap -s 100000000 -t 3600`

Task 2: Endpoint and Flow Analysis

Level of Effort: Low

Task Description: Use a tool to analyze the captured traffic. Identify all control protocols contained. Attempt to identify all associated endpoints and their relative functions. Associate protocols with endpoints and actions you took during capture.

Task Goal: Understand functionality and use of network.

Common ICS Network Protocols by Purdue Level

Level 0/1 Protocols (Mostly Fieldbus Management)

- EtherCAT: UDP/34980
- EtherNet/IP: TCP/44818, UDP/2222,44818
- FL-net: UDP/55000 to 55003
- FOUNDATION HSE: TCP/1089-1091, UDP/1089-1091
- HART-IP: TCP/5094, UDP/5094
- PROFINET: TCP/34962-34964, UDP/34962-34964

Level 2/3 Protocols (Mostly Regional SCADA)

- Modbus TCP: TCP/502
- DNP3: TCP/20000, UDP/20000
- WITS: TCP/20000, UDP/20000
- DLMS/COSEM: TCP/4059, UDP/4059
- IEC 104: TCP/2404
- IEEE C37.118: TCP/4712, UDP/4713
- MMS: TCP/102

Level 2/3 Protocols (Mostly Supervisory)

- **HMI, Alarm Servers, and Historians**
  - OPC UA: TCP/4840
  - OPC UA XML: TCP/80, TCP/443
- **Regional SCADA Control Centers**
  - ICCP/TASE2: TCP/102
- **Building Automation Specific Protocols**
  - BACnet/IP: UDP/47808
  - LonTalk: UDP/1629, UDP/1628
  - Fox (Tridium/Niagara): TCP/1911
  - KNXnet/IP: TCP/3671, UDP/3671

Endpoint and Flow Analysis

- Start Wireshark and open the following capture:
  `~/Protocols/Combined/Plant1.pcap`
- Using Wireshark, answer the following questions
  - How many packets from each control protocol are there?
    - Use Statistics > Protocol Hierarchy tool
  - Are any IPs speaking more than one of those protocols?
    - Use Statistics > Endpoints tool
  - Is one IP an obvious master server of some sort?
    - Use Statistics > Conversations tool
  - If you graph I/O over time by bytes, you will see a traffic spike in the middle of the capture. Which protocol caused it?
    - Use Statistics > I/O Graph tool
- Now try to run that strings command on the PCAP to get a different view

```
$ cd /home/control/Protocols/Combined/
$ strings Plant1.pcap | sort | uniq -c | sort -rn | less
```

Using GrassMarlin for Endpoint Analysis

- Start GrassMarlin
- Select Import Files, add Plant1.pcap, and choose Import Selected
- Right click on 141.81.0.10 and select View Details to learn more about this master server and what it is running
- Right click on the diagram background and select Group By to learn more about the other endpoints

Task 3: Known Protocol Analysis

Level of Effort: Low

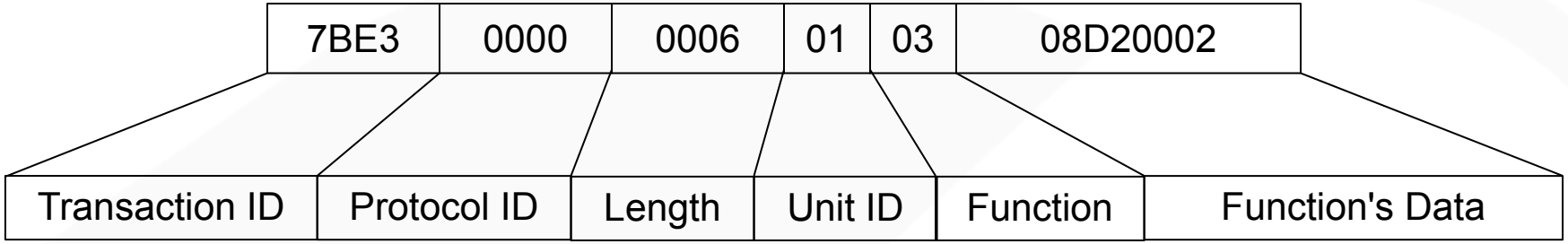
Task Description: Use a tool to analyze the captured traffic. Identify any security enabled on the protocol. Associate individual packets and conversations with actions you took during capture. Look for interesting functions and repeated actions in the protocols. Start mapping out registers/tags being used on the endpoint, especially for control.

Task Goal: Understand functionality and use of network.

Modbus

- Developed by Modicon in 1979
  - 3 serial (fieldbus) versions: **Modbus RTU**, Modbus ASCII, and Modbus Plus
  - 2 TCP/IP version: **Modbus TCP** and Modbus UDP
- Widely accepted protocol (implemented by hundreds of vendors) used in multiple industries
- Master (MTU/HMI/FEP) to field/slave (RTU, PLC, IED) communication
  - Only a simple request/response protocol
  - Master station must poll the field device
  - Field device cannot initiate communications
  - Refers to digital outputs as "coils" (1 bit) and analog input/output as "registers" (all 16 bit)
- Transferred to a foundation and became an 'open' protocol in the early 2000s
- Security was not a part of the design
- While Modbus is one of the most supported, it has serious drawbacks compared to modern control protocols when used with complex systems!

Modbus TCP



Name	Length	Description
Transaction ID	2 bytes	For synchronization between messages of server & client
Protocol ID	2 bytes	Zero for Modbus/TCP
Length Field	2 bytes	Number of remaining bytes in this frame
Unit ID	1 byte	Slave Address (255 if not used)
Function code	1 byte	Function codes as in other variants
Data bytes	n bytes	Data as response or commands

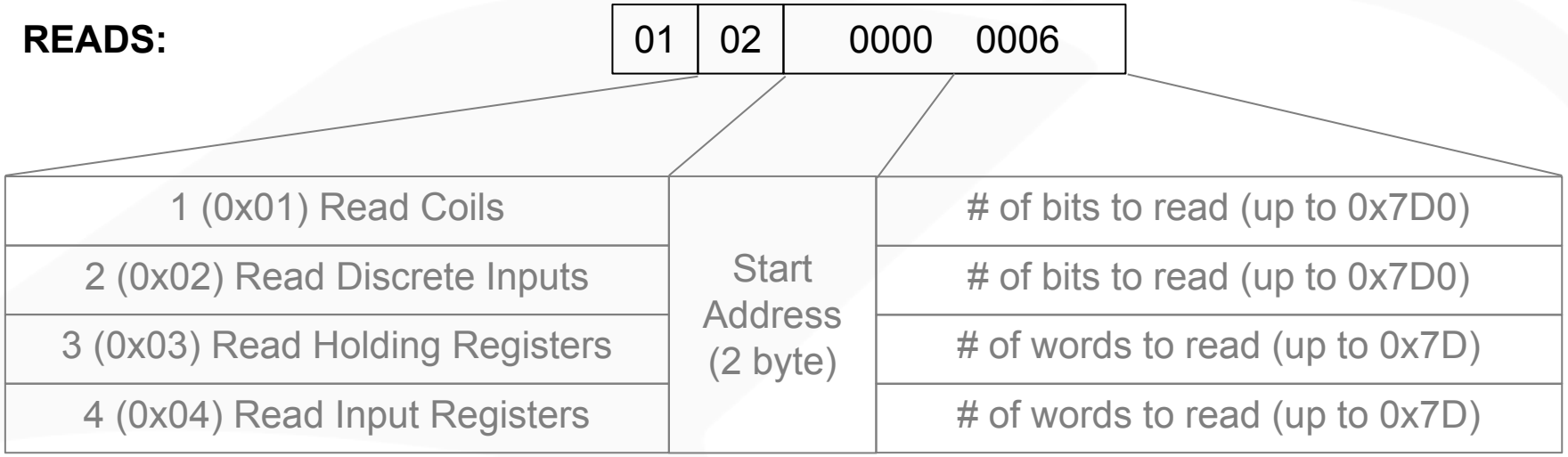
http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

Modbus Data Function Codes

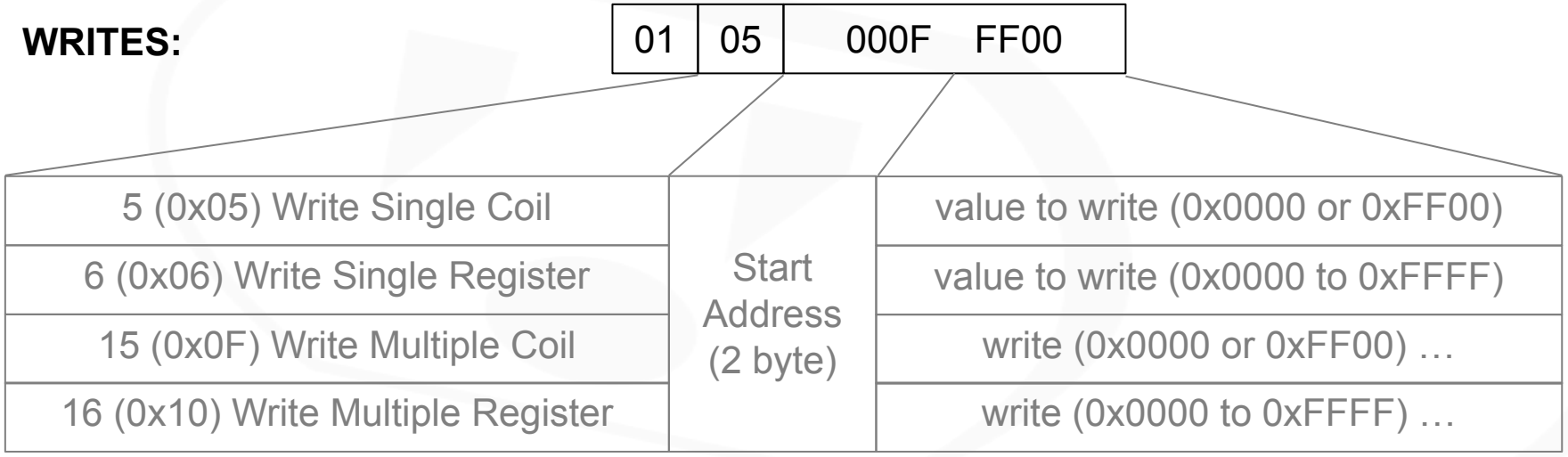
Function Category			Function Name	Code	(Hex)
Data Access	Bit access	Physical Discrete Inputs	Read Discrete Inputs	2	0x02
		Internal Bits or Physical Coils	Read Coils (Outputs)	1	0x01
			Write Single Coil	5	0x05
			Write Multiple Coils	15	0x0F
	16-bit access	Physical Input Registers	Read Input Register	4	0x04
		Internal Registers or Physical Output Registers	Read Holding Registers	3	0x03
			Write Single Register	6	0x06
			Write Multiple Registers	16	0x10
			Read/Write Multiple Registers	23	0x17
			Mask Write Register	22	0x16
			Read FIFO Queue	24	0x18
	File Record Access		Read File Record	20	0x14
			Write File Record	21	0x15

Common Modbus Functions

READS:



WRITES:



Modbus Diagnostic Function Codes

Function Category	Function Name	Code	(Hex)	SubCode	(Hex)
Diagnostics	Read Exception Status	7	0x07		
	Diagnostic	8	0x08	00 | 18 20	0x00 | 0x12 0x14
	Get Com Event Counter	11	0x0B		
	Get Com Event Log	12	0x0C		
	Report Slave ID (serial only)	17	0x11		
	Read Device Identification	43	0x2B	14	0x0E
Other	Encapsulated Interface Transport	43	0x2B	13 14	0x0D 0x0E

Modbus Capture Analysis

- Start Wireshark and open the following file, which is an export from Plant1.pcap:
  `~/Protocols/ModbusTCP/ModbusTCP.pcap`
- Using Wireshark, answer the following questions
  - Look at packet 6 and 7, what is happening here? What about packet 34 and 46?
  - How many Modbus masters and field devices are in this capture?
  - Is the master writing any data to the field devices?
    - What signal is being sent in packet 28?
    - What signal is being sent in packet 6023?
  - How is packet 6463 different to the other packets we've referenced?
    - Having multiple Modbus PDUs per TCP packet make it hard to analyze with Wireshark

- Run bro on the Modbus capture to extract each Modbus req/res to its own line

```
$ bro -r ModbusTCP.pcap
$ less modbus.log
```

- From a terminal, try running strings on this Modbus capture:

```
$ strings ModbusTCP.pcap | sort | uniq -c | sort -nr | less
```

PROFINET (PROcess FIeld NETwork)

- Ethernet implementation of PROFIBUS
- Uses EtherType 0x8892 for optimization
- Has three different versions:
  - TCP/IP for non time-critical data and plant commissioning with 100ms reliability
  - RT (Real-Time) for PROFINET IO applications up to 10ms reliability
  - IRT (Isochronous Real-Time) for PROFINET IO applications in drive systems with 1ms reliability
- Uses PROFINET_DCP (Discovery and Basic Configuration Protocol) to configure station names and IP addresses
- Each field device module has slot and sub-slot identifiers
  - Modules that don't produce network data usually have 0 length data fields
  - Each module's data usually ends with 0x80 for valid data and 0x00 for failures

EtherNet/IP (Industrial Protocol)

- Ethernet implementation of DeviceNet/ControlNet/CompoNet
- Facilitates the use of the Common Industrial Protocol (CIP)
- Used broadcast UDP messages for I/O data
  - Broadcasts can overwhelm devices on the network depending on how many messages are required
  - Only an issue for I/O traffic and not a problem at the workstation/server level.
- Industrial Ethernet features include data rates defined by the engineer
- Uses a Requested Packet Interval Rate (RPI) that can achieve 5 ms rates
- Uses broadcast but newer versions support unicast messages

IEC-104 (IEC 60870-5-104) & DNP3 (IEEE 1815-2012)

- Primarily used in electric sector SCADA, some use in water, oil, and gas
  - IEC 60870-5-104 (called IEC 104 by most engineers) is used in Europe
  - Distributed Network Protocol (DNP3) is used in North America and Australia
  - DNP3 is based on a early IEC 60870-5 draft, created by Westronic (now GE)
- Master (HMI/FEP) to field (RTU, PLC, IED) communication
  - Can use polling and tag subscription models, with timestamps
  - Also supports many other communication models beyond simple read/write
- Both provide optional authentication, HMAC signing, and TLS tunneling

MMS (IEC 61850-8-1)

- Manufacturing Messaging Specification
- Primarily used for modern electrical substation communications
- Expected to eventually replace DNP3 and IEC-104
- Master (HMI/FEP) to field (RTU, PLC, IED) communication
  - Uses symbolic names for data points (aka tag names in the protocol...)
  - Supports self-descriptive services (ask a controller for its available tags...)
  - Can pull metadata with measured data
- Because of these new features, it is recommended to never deploy MMS without cryptographic protections

OPC & OPC UA (IEC 62541)

- An ICS vendor-neutral protocol for middle tier control networks
  - Provides a consolidated or converged view of that with data
  - OPC allows us to gather data and generate views for the business
  - HMIs, Alarm Servers, Historians
- OPC UA is the successor to OPC (OLE for Process Control)
  - Multiplatform support (no DCOM/RPC)
  - Increased scalability
  - Portability to embedded devices
  - Improved security
- OPC UA provides a common framework to interface
  - Devices can send data using their protocol to a server that can translate and serve the information to OPC capable clients

ICCP (IEC 60870-6/TASE.2)

- Inter-Control Center Communications Protocol
- Mostly used by electric Independent System Operators (ISOs) with
  - Transmission Operators
  - Independently run Generators
- Can be used to send controls or as read-only
- Supports authentication and encryption in latest standard via TLS
  - Not always supported by vendor products
  - Not always configured by asset owners

Task 5: Unknown Protocol Decode

Level of Effort: Medium to Extremely High

Task Description: If traffic capture is using an unknown protocol, decode the network captures in an attempt to understand the protocol. Analyze each capture in light of the actions performed to initiate that traffic. For instance, if analyzing a traffic capture of a firmware update, try to identify the firmware being sent in the payload. Additionally, analyze actions such as initial registration between devices to determine if an authentication mechanism is being used.

Task Goal: Identify the purpose of blocks of data that could be used in later analysis.

Finding Unknown Protocols

- Start Wireshark and open the following file:
  `~/Protocols/Combined/Plant1.pcap`
- Using Wireshark to find the unknown TCP and UDP protocols
  - Use the filter `data` to show unknown protocols
  - Use Statistics > Protocol Hierarchy tool (right click on known protocols to filter them out)
- Your new filter should look something like this: `data && !icmp && !tds`
- Use Statistics > Conversations tool
  - make sure to check Limit to display filter
  - TCP 1433 is still showing up … possibly a bad TDS dissector or noncompliant protocol?
- Modify your filter to get rid of the extra TCP 1433 traffic:
  `data && !icmp && !tcp.port==1433`
- How many different unknown protocols can you see based similar TDP/UDP port numbers in conversation view?

Analyze the Unknown Protocols

- Use Wireshark's Export Specified Packets feature to export each of the following filter results to its own PCAP file like: Plant1-port2111.pcap

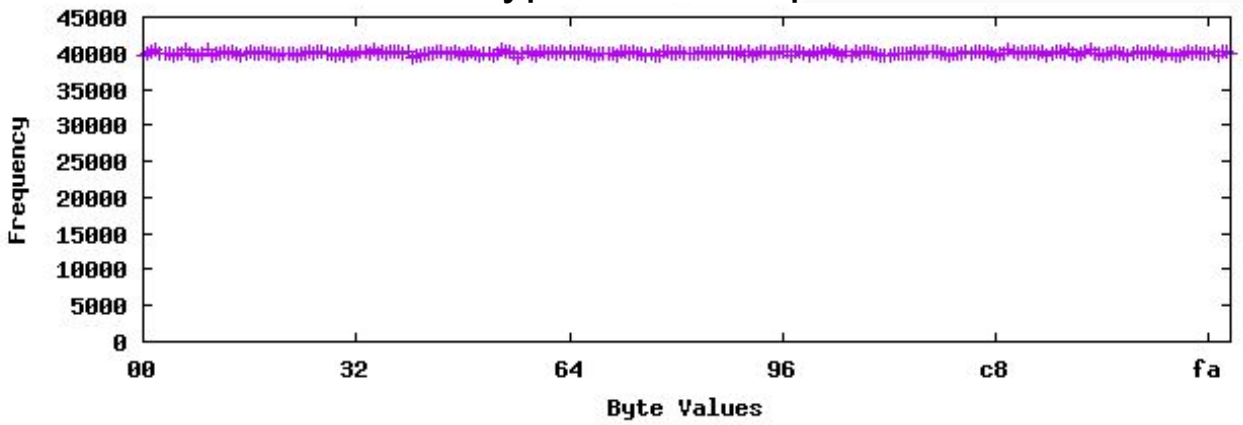
```
tcp.port == 2111
tcp.port == 5026 || udp.port == 5026
tcp.port == 5413
tcp.port == 8000
udp.port == 33000
```

(note: only between .10 <----> .11)

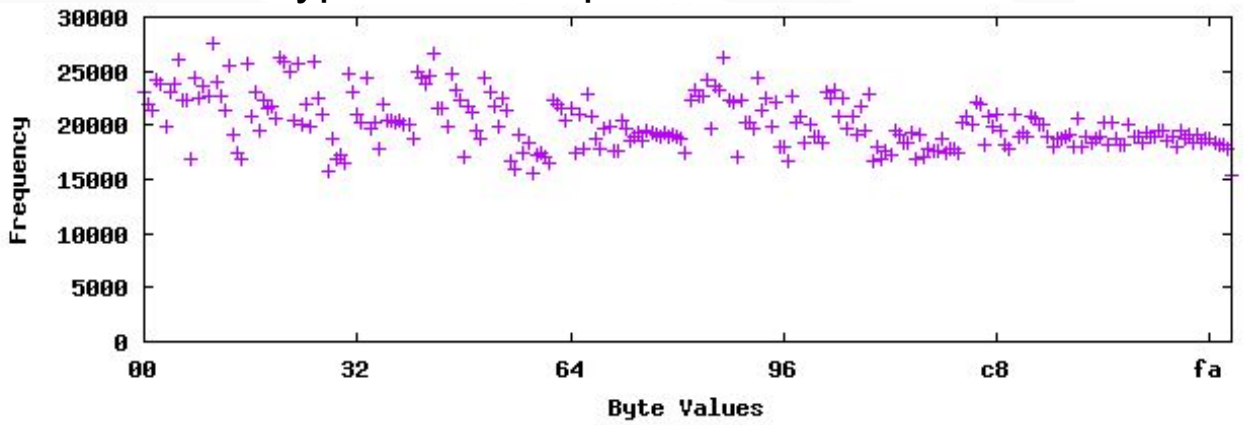
- Use Google to search for each port to see if we can identify a possible protocol
  - TCP 2111: opnet technologies
  - TCP/UDP 5026: archestra
  - TCP 5413: Wonderware HMI (SuiteLink) or wwwio
  - TCP 8000: Weintek HMI
  - UDP 33000: ?EverQuest?

Is the protocol encrypted? pcaphistogram by Josh Wright

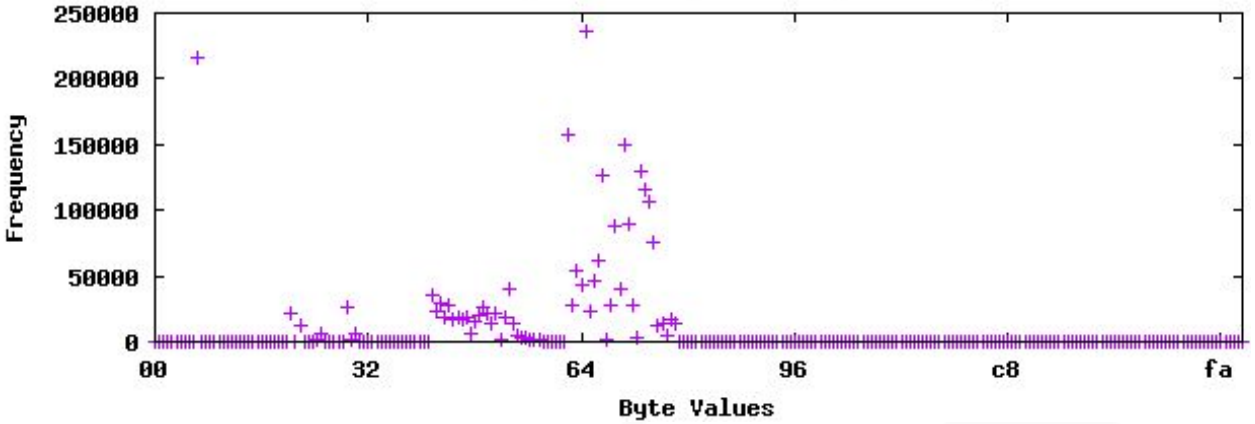
Encrypted or Compressed



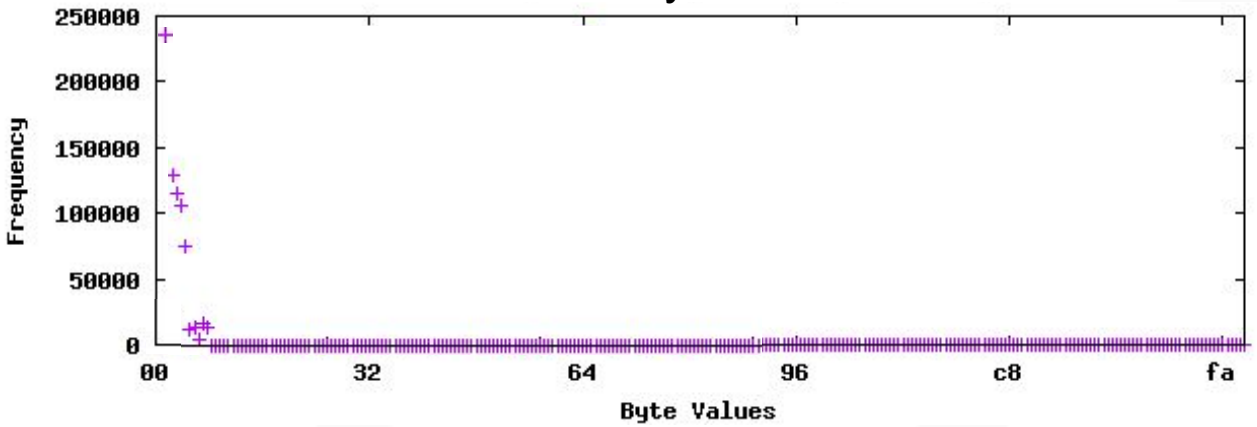
Encrypted or Compressed with IV or Metadata



ASCII Protocol



Binary Protocol



Open a terminal and run pcaphistogram to analyze each exported PCAP:

```
$ pcaphistogram Plant1-port2111.pcap | gnuplot
$ eog Plant1-port2111.png
```

Basic Analysis

- Try using various string encodings on the each of your exported PCAP files:
  - ASCII, 16 bit Unicode (little and big endian), 32 bit Unicode (little and big endian)

```
$ strings Plant1-port2111.pcap | sort | uniq -c | sort -rn | less
$ strings -e l Plant1-port2111.pcap | sort | uniq -c | sort -rn | less
$ strings -e L Plant1-port2111.pcap | sort | uniq -c | sort -rn | less
$ strings -e b Plant1-port2111.pcap | sort | uniq -c | sort -rn | less
$ strings -e B Plant1-port2111.pcap | sort | uniq -c | sort -rn | less
```

- Use GrassMarlin to learn more about these unknown protocols
  - Start GrassMarlin
  - Import and analyze each exported pcap file individually
  - Select Import Files, add `Plant1-port2111.pcap`, and choose Import Selected

Task 6: Compare with Asset Database

Level of Effort: High to Extremely High

Task Description: Compare findings to asset databases

Task Goal: Identify gaps in asset database or gaps in network captures.

Task 7: Walking the wires

Level of Effort: Medium to High

Task Description: For gaps in the asset database, when devices are identified in network captures, you could consider walking the communication wires to identify physical devices with the new findings from your captures.

Task Goal: Identify the physical assets to supplement findings from captures.

Task 8: Reporting

- Executive Summary
  - a brief 1-2 page section discussing the overarching root causes for the vulnerabilities and high level business strategies to address these root causes.
- Introduction
  - a short section describing the goals of the tests, components that were in and out of scope, any special restrictions on the tests, and the team involved with the testing.
- Methodology
  - a short section of the report focuses on the technical reasons for the test as well as the methodology used.
- Findings and Recommendations
  - this section of the report is traditionally the longest, most detailed, and highly technical. This is the core of the report for future use and reference. This section may also discuss the likelihood and impact of each vulnerability within the context of the proposed or existing deployment.
- Conclusion
  - a section similar to the executive summary but at a more technical depth summarizing the major findings and recommendations. This section should also discuss any major questions or goals of the assessment such as the team's recommendations of a go no-go purchase of a product.

Traditional Network Pentests On ICS

Overview of a traditional network penetration test methodology

Dangers of port and vulnerability scanning

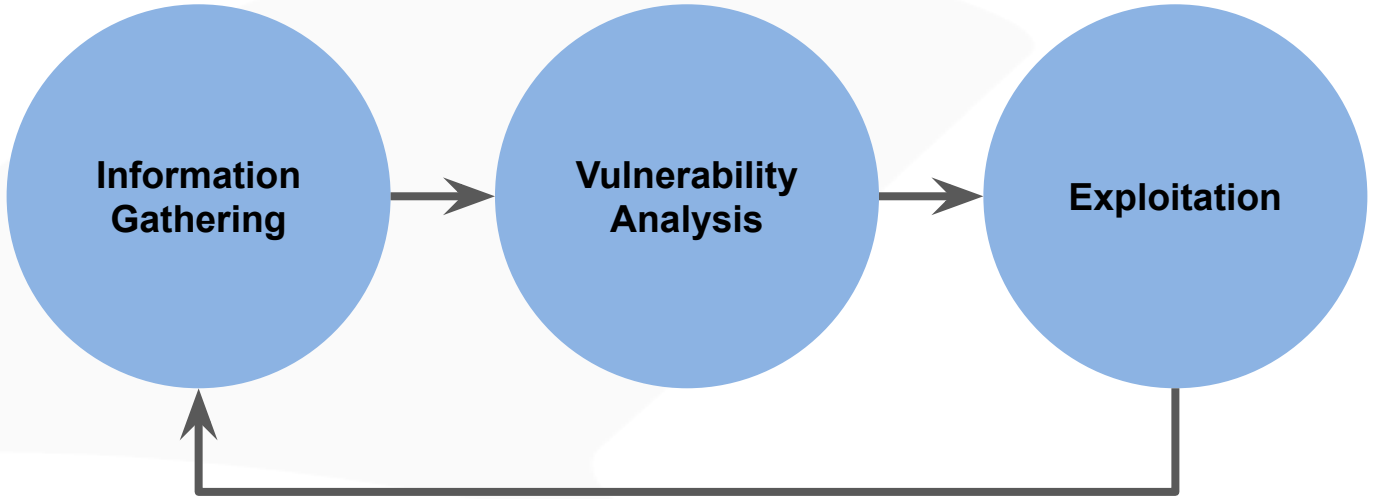
Strategies to perform port and vulnerability scanning

Hands-on network pentesting of master servers

When to do Active Assessments

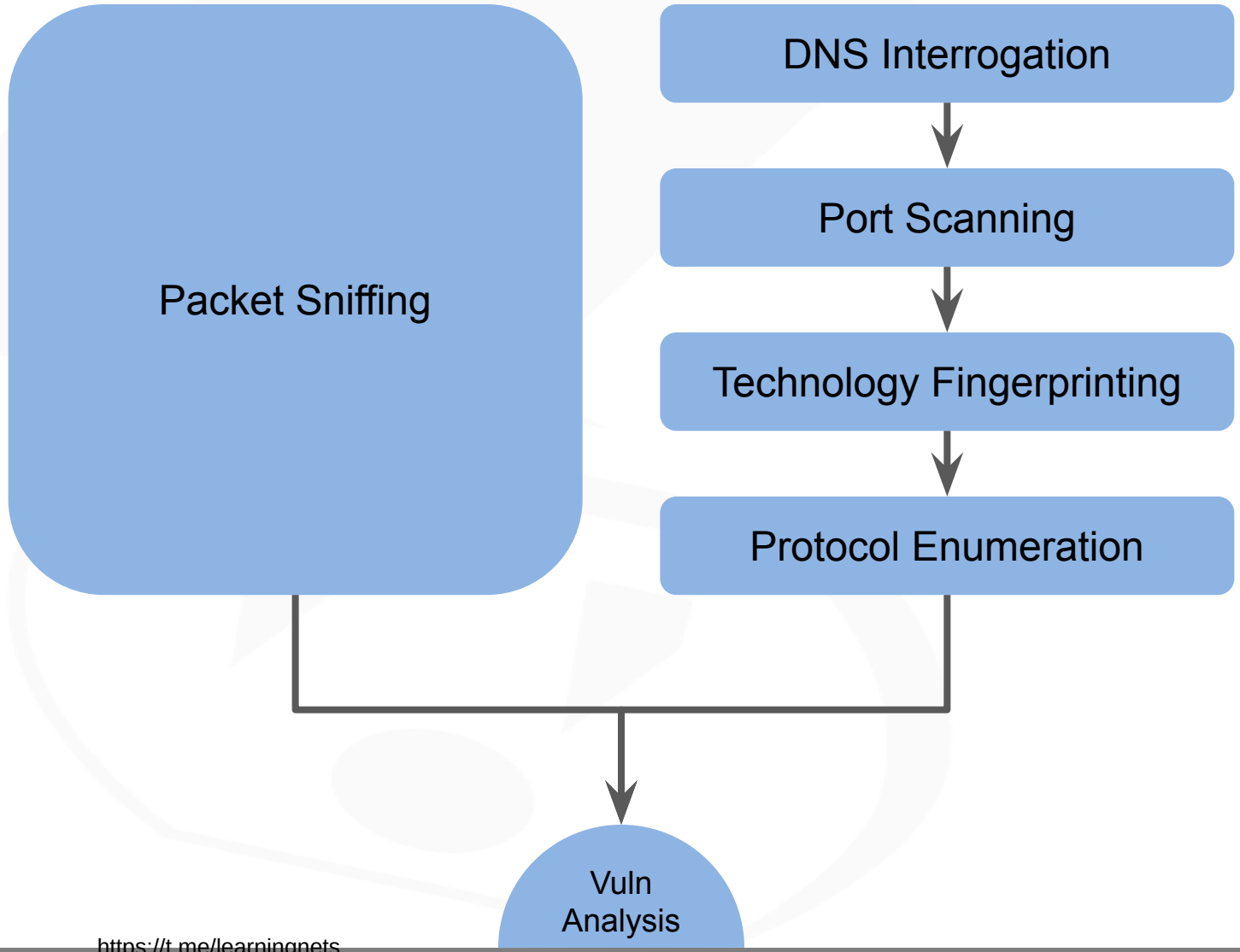
- All connectivity from corporate networks to ICS networks
- All remote access points into the ICS networks
- Any link carrying ICS traffic across public or semi-public links
- Any new system or device in a lab before it is implemented
- Any system changes or updates that are being tested in test or staging environments
- Should we do penetration testing on the sensitive network itself?
  - Maybe. The risk might be too great.
  - Where risk is low, limited penetration testing should be considered
    - Know what equipment is in the subnets you are working with before starting
    - Test single systems slowly with staff on hand in case of a problem
    - Be VERY aware of your tools and what actions they perform
    - Avoid any problematic actions and NEVER perform risky actions
  - Why even consider it? Because attackers attack production, not testing

Vulnerability Scanning and Penetration Testing

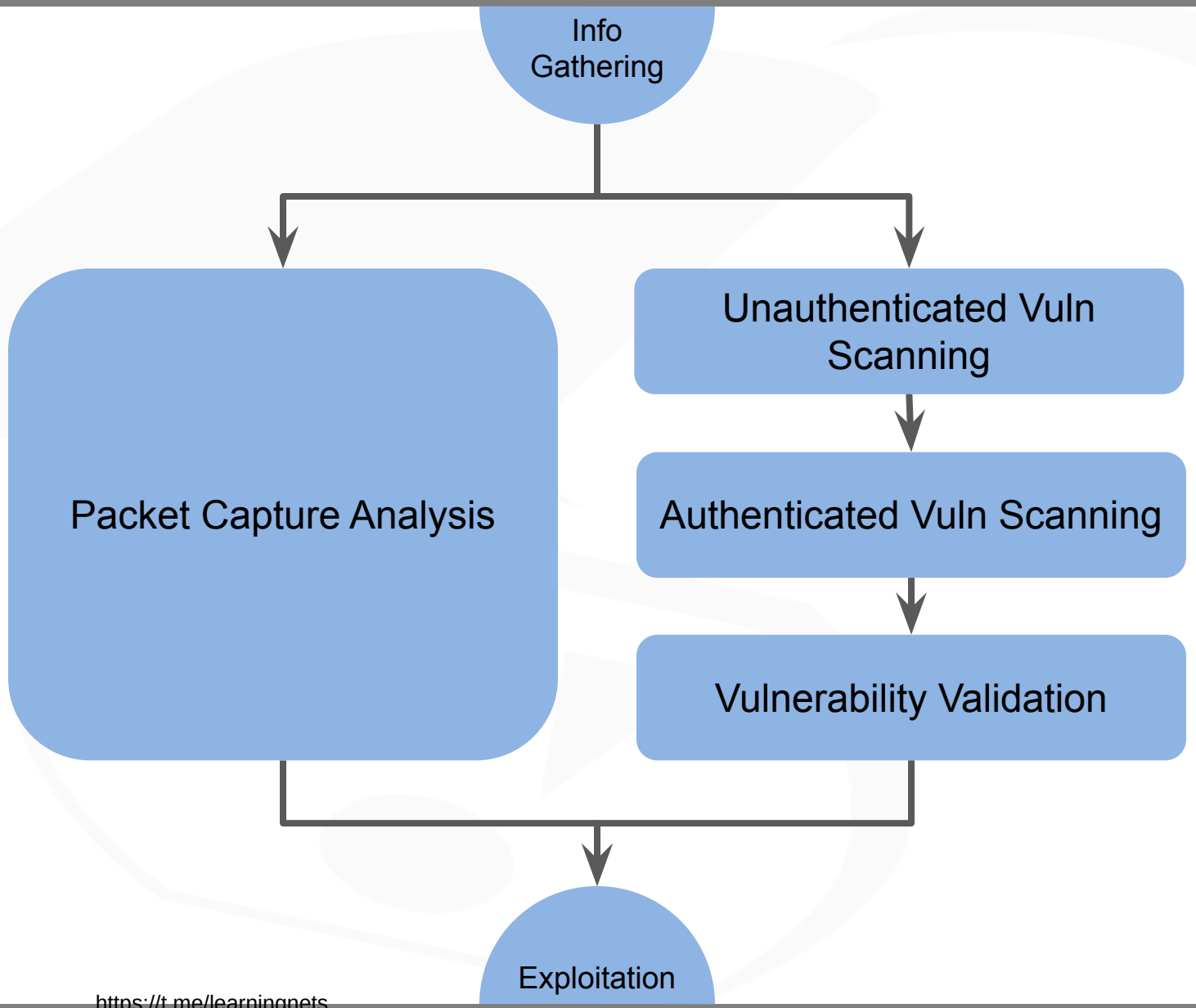


- Vulnerability Scanning: the first part of penetration testing
- Penetration Testing: Enhanced vulnerability discovery and assessment with exploitation

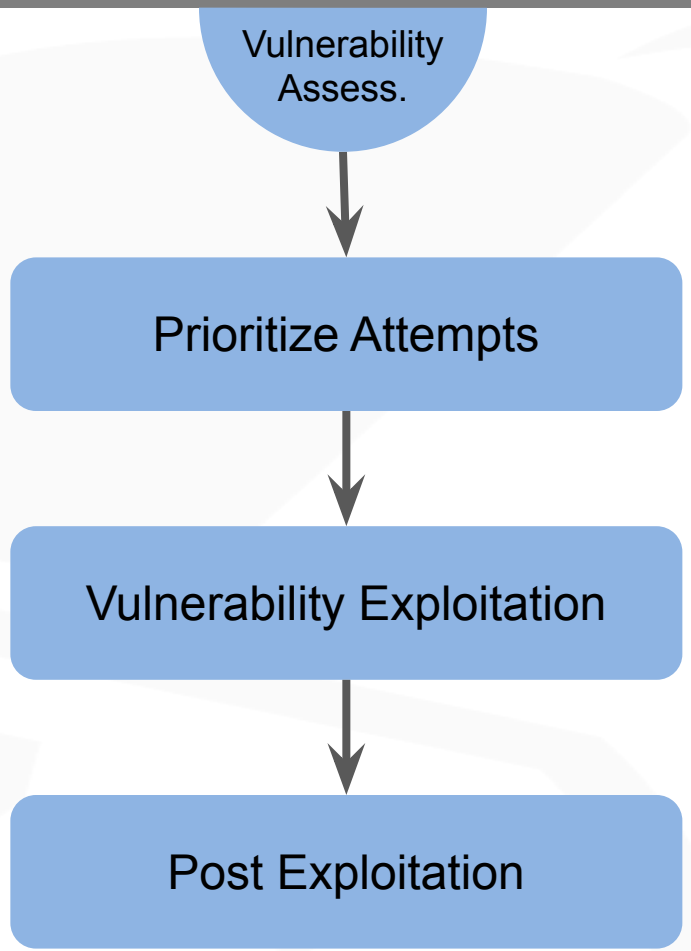
Server OS – Information Gathering



Server OS – Vuln Analysis



Server OS – Exploitation



Dangers of Port Scanning

- Port scanning can crash legacy embedded systems if not careful! Here are the most likely causes:
  - OS Fingerprinting
    - Don't use the `-O` or `-A` flags in nmap
    - By far the moly likely cause of crashed embedded systems
    - Can do ARP scans locally on each subnet and use MAC to ID devices
  - Scanning with SYN scans
    - Default when using nmap with sudo or running it as root
    - Not proper RFC behavior, so only mature TCP/IP stacks handles this properly
    - Always specify `-sT` in your scans to avoid this accident
  - Scanning too fast (yes, the defaults in nmap are too fast)
    - Use nmap's `-T2` setting sets this at 0.4 seconds
    - Or use nmap's `--scan-delay 0.1` or `--max-parallelism 1` to scan 1 port at a time per host
  - Scanning UDP ports with null payloads **(can affect ICS software on Windows and Linux too!!!)**
    - Don't use the `-sU` option in nmap
  - Service fingerprinting usually safe, but can occasionally cause problems
    - Use nmap's `-sV` selectively on new subnets
    - Or use nmap's `--script=banner`

nmap Suggestions

- Always run nmap with sudo with -sT
  - nmap rarely tells you its needed (only says it for `-O`)
  - Requirements vary from OS to OS
  - Required for all ICMP functions
  - Required for OS fingerprinting
  - Required for some NSE scripts
  - If you don't believe me, make and diff some pcaps
- Its always good practice to use `-v` when scanning

Low Risk Portscans

`sudo nmap –n -PR -sn`

- Risk = Almost None (only does ARP request (IP -> MAC) which is required by TCP)
- Value = retrieves MAC address if IP is live, which can be used to fingerprint
- Note = this must be done from the SAME subnet as the IP being scanned

`sudo nmap -n -sn`

- Risk = Very Low (only sends ICMP and TCP80/443 ping requests)
- Value = shows if IP address is responding to pings
- Note = if done on same subnet, will retrieve MAC address

`sudo nmap -n -sT --scan-delay 0.1 --top-ports 100 ...`

- Risk = Low (scans each host's TCP ports serially with one-tenth of a second delay)
- Value = Medium (tests for most common TCP servers...but not sensitive/proprietary protocols)

`sudo nmap -n -sT --scan-delay 0.1 -p ??? ...`

- Risk = Low (scans each host's TCP ports serially with one-tenth of a second delay)
- Value = Medium (tests for whatever services you specify)

Medium to High Risk Port Scans

`sudo nmap –n -sT --max-parallelism 1 -p ??? ...`

- Risk = Medium Low (scans each host's TCP ports serially as fast as possible)
- Value = Medium High (tests for whatever services you specify but quickly)

`sudo nmap –n -sT --max-parallelism 1 -p- ...`

- Risk = Medium (scans each host's TCP ports serially as fast as possible)
- Value = High (scans all possible TCP ports)

`sudo nmap –n -sT --max-parallelism 1 -p- -sV ...`

- Risk = Medium High (scans each host's TCP ports serially as fast as possible)
- Value = High (scans all possible TCP ports)

`sudo nmap –n -sT -p- -A ...`

- Risk = High (likely to crash most old gear and even some modern)
- Value = High (scans all possible ports, fingerprints everything, and runs NSE)

`sudo nmap –n -sT -sU -p- -A ...`

- Risk = Extremely High (likely to crash most old gear and even some modern)
- Value = High (scans all possible ports, fingerprints everything, and runs NSE)

How Vulnerability Scanners Work

- Network Port Scanning
  - basically like what nmap does WITHOUT as many options
- Service Fingerprinting
  - most vulnerabilities are identified this way
- Vulnerability Probing
  - only uses this technique to find some vulnerabilities
- Authenticated Scanning
  - logs in if credentials are provided
  - pulled patch levels
  - pulls listening ports via the netstat command
- Custom Audit Checks
  - script virtually any OS or application check desired

Low Risk Authenticated Scans with Nessus

- Decreases risk by removing TCP/UDP port scans and vulnerability probes
- To use Nessus audit checks:
  - In Nessus, create a new scan profile
  - Disable all Nessus TCP, SYN, UDP, and SNMP port scans
  - Leave Netstat and Ping port scans open
  - Disable all Nessus plugins except Windows/Linux compliance checks
  - In Preferences, configure the compliance checks to use any third party or custom made audit files (windows security policy or plain text file values)
- Create a new scan and tell it to use your new profile
- Some older scan profiles were made for ICS by Digital Bond
  - Part of their Bandolier Project
  - Nessus has changed their audit language, so they would need updating

Deprecated Bandolier Audit Files from Digital Bond

- General Purpose Audit Files:
  - Bandolier Baselines Package
  - NERC CIP-007 R8 Scan Policy Package
- Application Specific Audit Files:
  - ABB 800xA PPA Release Package
  - AREVA e-terra Release Package
  - Control Systems International UCOS Release Package
  - Emerson Ovation Release Package
  - Matrikon Security Gateway/Tunneller Release Package
  - OSIsoft PI Enterprise Server Release Package
  - Siemens Spectrum Power TG 8.2 Release Package
  - SISCO AX-S4 ICCP Release Package
  - SNC GENe Release Package
  - Telvent OASyS DNA 7.5 Release Package
- These files will need to be modified for your custom installs, versions, and policies

Homework

Homework

- ICS Protocol Primers via YouTube Playlists
  - DNP3 Protocol: http://bit.ly/2NaZttt
  - IEC 60870-5 Protocol: http://bit.ly/2SHwpjc
  - IEC 61850 Protocol: http://bit.ly/2SWd8tn
- Check out some of the other ICS network protocol captures in the Control Things Platform

Author Contact Information

Justin Searle
training & opensource: justin@controlthings.io
consulting & testing: justin@inguardians.com
cell: 801-784-2052

twitter: @meeas
Facebook: www.facebook.com/m33as
LinkedIn: www.linkedin.com/in/meeas
GitHub: github.com/meeas