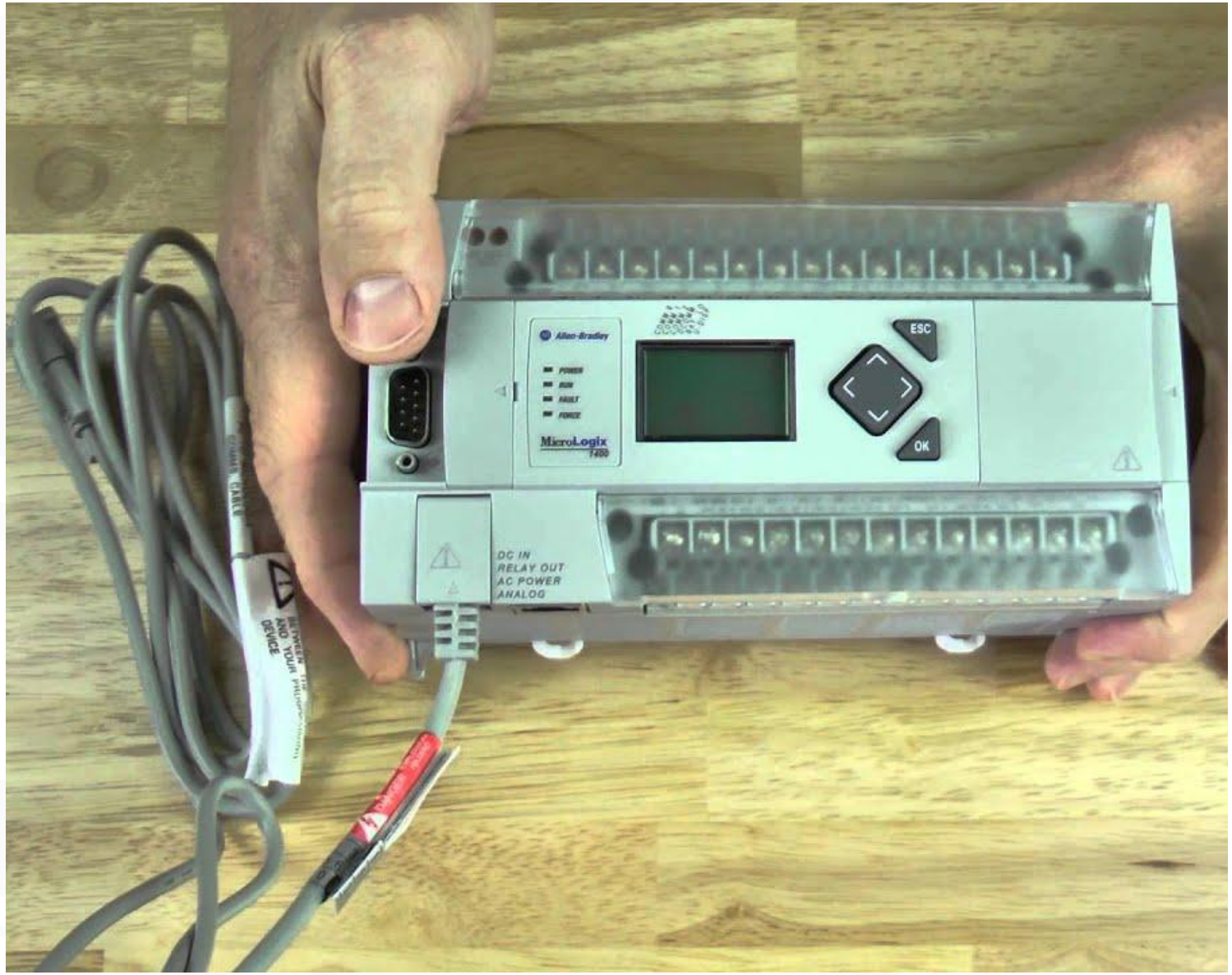
Assessing and Exploiting Maintenance Interfaces

Version 38

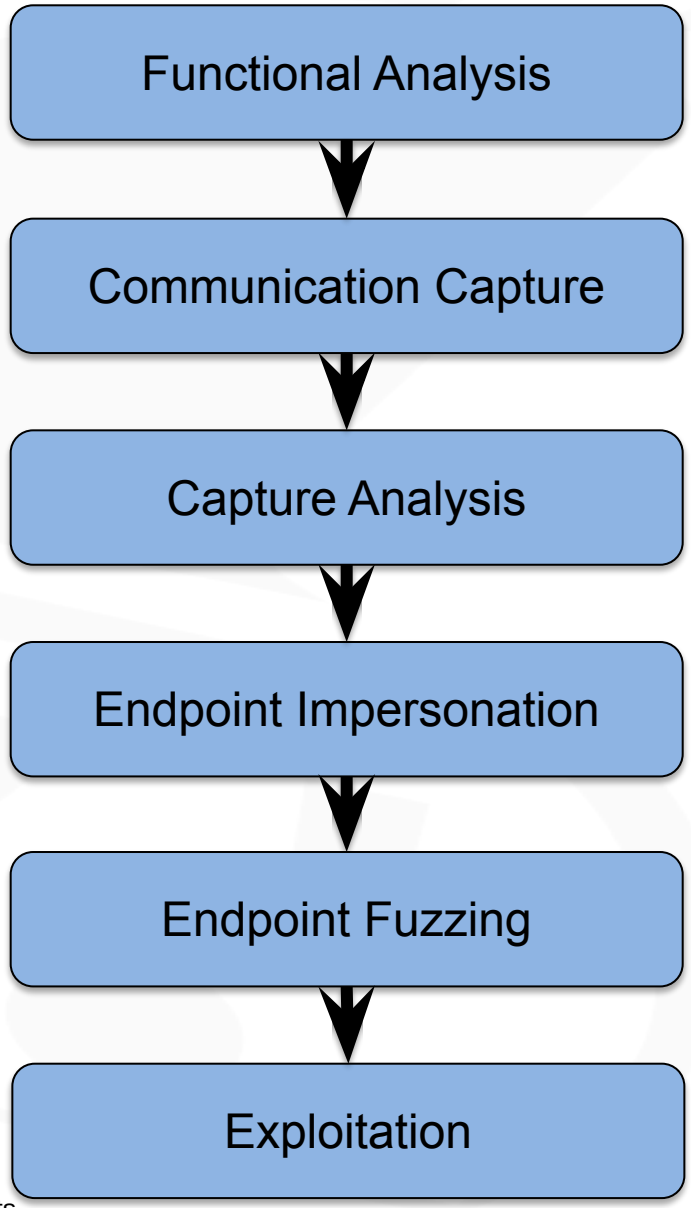
801-784-2052 // justin@controlthings.io

Examples of Maintenance Interfaces



https://t.me/learningnets

Local Maintenance Interface Testing



Task 1: Interface Functional Analysis

Level of Effort: Low

Task Description: Obtain required software and hardware to establish an appropriate connection to the field device, be it a serial port, infrared port, or digital display. Identify the intended functionality and features of the interface. Identify any unprotected or high- risk functions that attackers may be interested in exploiting, such as firmware updates, configurations, or security table reads.

Task Goal: Gain an understanding of the interface feature set and identify functions that should be targeted for later tasks.

Functional Analysis of vBuilder

- On your Windows machine, open our ChemicalMixerSFC-Ace1600-Completed project in vBuilder
- Connect your PLC to it
- Identify the functionality that you think causes immediate interaction with the PLC on the USB bus
- Before we move to the next section, please go ahead and reprogram your PLC, but do not hit the Run/Play button

Task 2: Communications Capture

Level of Effort: Low

Task Description: Use a hardware or software tool to intercept normal communications on the interface. Capture all identified target functions from previous tasks.

Task Goal: Obtain low-level capture of targeted functions.

Ways to Capture Maintenance Interfaces

- Most modern technician/maintenance tools connect via USB or serial, so.....
- Hardware: Total Phase Beagle, ZeroPlus, or USB Proxy + BBB
- Wireshark: Windows (via USBpcap) and Linux (via usbmon)
- Windows XP, 7: Portmon from Sysinternals
- Windows 10: HHD Software Device Monitoring Studio
- Linux: usbmon driver from modern kernels (since 2.6.11)
  - Can access through `/sys/kernel/debug/usb/usbmon`
  - Example of what you'll see: `0s 0u 1s 1t 1u 2s 2t 2u`
  - Number is USB bus, `0` represents all busses, `u` is for full data captures
- VMware Virtual Machines
  - Add the following lines to your virtual machine's vmx file

```
monitor = "debug"
usb.analyzer.enable = TRUE
```

  - USBIO logs are added to the vmware.log file

Windows USB Bus and Device IDs

1. Make sure a recent version of Wireshark and USBpcap are installed re: course setup
2. We are assuming vBuilder is still open and your PLC is connected to it
3. Using Windows Device Manager
    1. Properties of VelocioComm (COM3) on the General tab next to Location
4. Using USBPcap
    1. Navigate to C:\Program Files\ USBPcap
    2. Run the USBPcapCMD.exe
5. USBPcap method is a little better since it shows what other devices are on the same bus
    1. Device IDs in both Windows Device Manager and USBPcap are consistent (n++ across all busses)
    2. Wireshark uses a different device IDs as we'll see in a bit (n++ for each bus)

Capturing the Velocio Protocol Traffic

1. Open vBuilder with the ChemicalMixerSFC-Ace1600-Completed project
2. Start Wireshark
    1. On the main startup screen, look in the Capture section for USBPcap# devices
    2. Doubleclick on USBPcap# that corresponds the the bus number your Velocio PLC is on
    3. Add a Wireshark filter of usb.capdata to show only the packets with serial traffic
    4. In the dissected tree-view, right-click on Leftover Capture Data and select Apply as Column
    5. You won't see anything until we start using vBuilder and vFactory (unless you have another device generating it, which you can ask the instructor or TA to help filter it out)
3. Change back to the vBuilder window and click on the green triangle Run/Play button
4. Go back to your Wireshark window and look at the serial traffic it captured when you pressed the Run/Play button
5. Go back to vBuilder and try hitting the Run/Play button again
    1. Anything different in Wireshark?
6. Go back to vBuilder and try hitting the Stop botton
    1. How is this different than the Run/Play button?

Comparing Play and Stop Functions

Play Function

> 56 ff ff 00 07 f1 01
< 56 ff ff 00 08 f1 06 01

> 56 ff ff 00 06 f3
< 56 ff ff 00 0f f3 06 01 01 00 00 00 00 00 00

> 56 ff ff 00 06 40
< 56 ff ff 00 09 40 06 00 00

Stop Function

> 56 ff ff 00 07 f1 02
< 56 ff ff 00 08 f1 06 02

> 56 ff ff 00 06 f3
< 56 ff ff 00 0f f3 06 01 02 00 00 00 00 00 00

> 56 ff ff 00 06 40
< 56 ff ff 00 09 40 06 00 00

> 56 ff ff 00 06 f5
< 56 ff ff 00 0c f5 06 00 00 00 00 00

vBuilder Function Capture

- Use Wireshark to capture the following vBuilder functions
  - Play
  - Stop/Pause
  - Reset
  - Enter Debug (Generates lots of packets, so disable within a few seconds)
  - Exit Debug
  - Step Into (Only works in debug mode)
  - Step Over (Only works in debug mode)
  - Step Out (Only works in debug mode)

- Use Wireshark's Coloring Rules and Packet Comment features to annotate your captures
- Add Packet Comments as its own column to make them easier to use
- Other captures you can work on if you have extra time:
  - Realtime clock, setup hardware, programming the PLC, firmware update
    (various firmware versions in Velocio PLC folder, your probably on the most recent)

Task 3: Interface Capture Analysis

Level of Effort: Medium

Task Description: Analyze interface captures, identifying weaknesses in authentication, authorization, and integrity controls. Gain an understanding of how data is requested and commands are sent. If the protocol uses authentication, attempt to identify the passwords or keys being sent before a session is established. For example, in the case of protocols such as C12.18 for AMI meters, attempt to identify the different levels of passwords being sent before each command.

Task Goal: Identify potential vulnerabilities and attacks.

Historical Example: Smart Meter Optical Interface

- This smart meter supposedly ran ANSI C12.18 protocol, so portmon was used on Windows XP to capture this

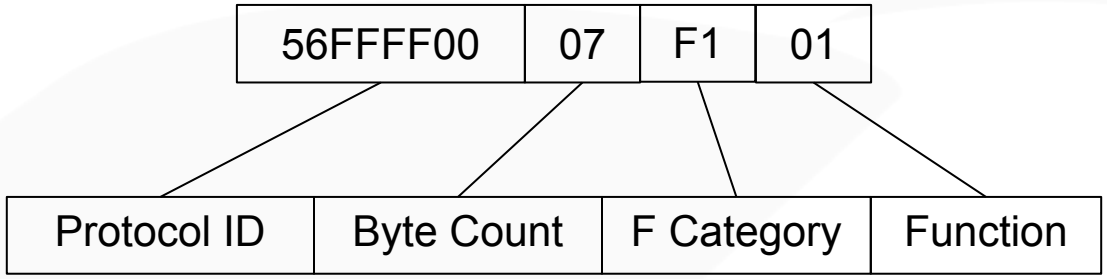
```
Write (SUCCESS):  EE 00 00 00 00 01 20 13 10
Read  (TIMEOUT):
```

- However after four such C12.18 initializations timeouts, then...

```
Write (SUCCESS): B4 00 00 00 00 00 08 3F 08 C9 00 00 00 00 14 C5 4D
Read  (SUCCESS): B4
Read  (SUCCESS): 00 00 00 00 00 17
Read  (SUCCESS): 00 00 14 30 39 37 30 37 35 33 39 00 00 00 00 00 00 00 00 30 39 37 30 0A 2F
Write (SUCCESS): B4 00 00 00 00 00 08 3F 08 CA 00 00 00 00 05 B0 40
Read  (SUCCESS): B4
Read  (SUCCESS): 00 00 00 00 00 08
Read  (SUCCESS): 00 00 05 60 0C 00 02 62 12 EA          (some redacted after this)
Write (SUCCESS): B4 00 00 00 00 00 08 3F 00 54 00 00 00 00 01 C8 A6  (password in red)
Write (SUCCESS): B4
Write (SUCCESS): 00 00 00 00 00 04
Write (SUCCESS): 00 00 01 08 D5 E8
```

- You can see the full exchange in the UserInterfaces folder in Control Things Platform

Velocio Protocol Decode

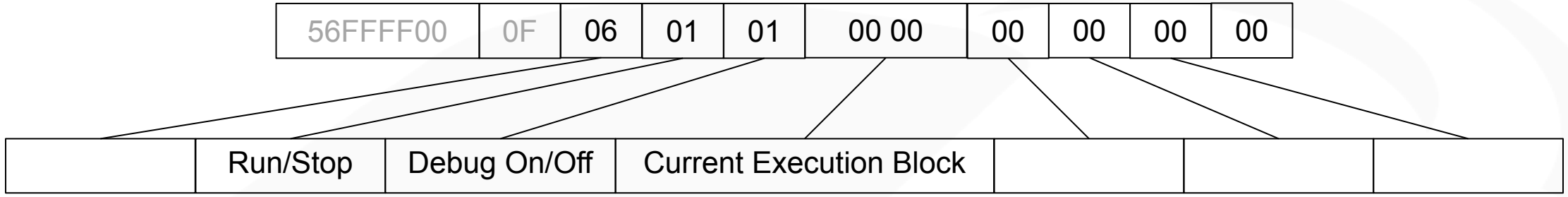


- Bytes 1-4 is repeated on EVERY message we capture, protocol identifier?
- The 5th byte changes in sync with the message length, so...
- The 6th byte we only see three options occur (F1, F2, and F3) which seem to correlate with different groups or function categories
- The 7th byte seems to be unique for each function in each category, so...

Function Categories and Functions in vBuilder

- F0: debug commands
  - 01: Exit Debug
  - 02: Enter Debug
- F1: normal commands
  - 01: Play
  - 02: Stop/Pause
  - 03: Step Into
  - 04: Step Out
  - 05: Step Over
  - 06: Reset
- F3: status commands
- F5:
- 40:
- We can only go so far with our analysis until we get to impersonation

Velocio Protocol Status Response



- We have a lot more to decode here, but we'll leave that as homework

vFactory Capture and Analysis

1. Make sure your PLC is running (not paused), then close vBuilder
2. Start capturing with Wireshark
3. Open the ChemicalMixerHMI.vfy file in vFactory
   - Analyze the communication with PLC when vFactory first starts
   - Analyze the communication while the HMI is monitoring the PLC
   - Analyze the signals sent for pause, emergency, and batch reset controls
   - Analyze any other buttons and other controls you may have added
4. Use Wireshark's Coloring Rules and Packet Comment features to annotate your captures

vFactory Startup Communication with PLC

```
>  56 ff ff 00 08 14 FF FF
<  56 FF FF 00 1B 14 06 FF FF EE AA 26 00 00 44 FF FF FF FF 00 C3 40 00 00 10 00 00
    V  ÿ  ÿ  .  .  .  .  ÿ  ÿ  î  ª  &  .  .  D  ÿ  ÿ  ÿ  ÿ  .  Ã  @  .  .  .  .  .
>  56 ff ff 00 06 AC
<  56 ff ff 00 0A AC 06 FF 00 15

>  56 ff ff 00 08 0A 00 01
<  56 FF FF 00 25 0A 06 00 01 53 65 72 69 61 6C 54 58 43 2D 31 20 20 20 20 20 01 00 00 00 00 01 00 00 FF FF 1F FF
    V  ÿ  ÿ  .  %  .  .  .  .  S  e  r  i  a  l  T  X  C  -  1                 .  .  .  .  .  .  .  .  ÿ  ÿ  .  ÿ
>  56 ff ff 00 08 0A 00 02
<  56 ff ff 00 25 0A 06 00 02 53 65 72 69 61 6C 52 58 43 2D 31 20 20 20 20 20 01 00 00 00 00 02 00 00 FF FF 1F FF
    V  ÿ  ÿ  .  %  .  .  .  .  S  e  r  i  a  l  R  X  C  -  1                 .  .  .  .  .  .  .  .  ÿ  ÿ  .  ÿ
>  56 ff ff 00 08 0A 00 03
<  56 ff ff 00 25 0A 06 00 03 52 75 6E 50 61 75 73 65 5F 53 77 20 20 20 20 20 01 00 00 01 00 01 00 00 FF FF 62 01
    V  ÿ  ÿ  .  %  .  .  .  .  R  u  n  P  a  u  s  e  _  S  w                 .  .  .  .  .  .  .  .  ÿ  ÿ  b  .

   (output redacted)

>  56 ff ff 00 08 0A 00 15
<  56 FF FF 00 25 0A 06 00 15 50 61 75 73 65 43 6E 74 54 72 69 67 67 65 72 20 01 00 00 03 00 04 00 00 FF FF 9F FF
    V  ÿ  ÿ  .  %  .  .  .  .  P  a  u  s  e  C  n  t  T  r  i  g  g  e  r     .  .  .  .  .  .  .  .  ÿ  ÿ  Ÿ  ÿ
```

HMI Changing PLC Tag Values

Pause On, Pause Off, Pause On, Pause Off

```
>  56FFFF00 15 11 01 00 01 00 00 09 01 00 00 03 00 01 00 00 01    Set PauseProcess = 1
>  56FFFF00 15 11 01 00 02 00 00 09 01 00 00 03 00 01 00 00 00    Set PauseProcess = 0
>  56FFFF00 15 11 01 00 03 00 00 09 01 00 00 03 00 01 00 00 01    Set PauseProcess = 1
>  56FFFF00 15 11 01 00 04 00 00 09 01 00 00 03 00 01 00 00 00    Set PauseProcess = 0
```

Emergency On, Emergency Off, Emergency On, Emergency Off

```
>  56FFFF00 15 11 01 00 05 00 00 09 01 00 00 03 00 02 00 00 01    Set EmergencyStop = 1
>  56FFFF00 15 11 01 00 06 00 00 09 01 00 00 03 00 02 00 00 00    Set EmergencyStop = 0
>  56FFFF00 15 11 01 00 07 00 00 09 01 00 00 03 00 02 00 00 01    Set EmergencyStop = 1
>  56FFFF00 15 11 01 00 08 00 00 09 01 00 00 03 00 02 00 00 00    Set EmergencyStop = 0
```

Set various registers to different values

```
>  56FFFF00 15 11 01 00 09 00 00 09 11 00 00 00 00 00 00 00 00    Set BatchCounter = 0
>  56FFFF00 15 11 01 00 0A 00 00 09 11 00 00 00 00 00 00 00 80    Set BatchCounter = 180
>  56FFFF00 16 11 01 00 0B 00 00 0A 21 00 00 01 00 00 00 00 03 E8       Set State = 1000
>  56FFFF00 18 11 01 00 0C 00 00 0D 41 00 00 00 00 00 00 00 00 00 03 E8    Set Timer = 1000
>  56FFFF00 18 11 01 00 0D 00 00 0D 41 00 00 01 00 00 00 00 00 00 03 E8    Set ETimer = 1000
```

Task 4: Interface Endpoint Impersonation

Level of Effort: Low to Medium

Task Description: Use an attack tool to impersonate either end of the field technician interface. For instance, this attack tool could simulate the field technician tool while communicating with the field device interface, or the attack tool could simulate the field device interface while communicating with the field device tool.

Task Goal: Obtain a usable attack point to perform later tasks.

Sending the vBuilder Commands with ctserial

- Inside VMware, pass control of your PLC to your Control Things VM
- Use dmesg or lsusb to verify it is connected to Control Things Platform
- Open a terminal and run the `ctserial` application

  `connect /dev/ttyACM0`
- Use the sendhex command to experiment sending the following commands:
  - Play: `56ffff00 07 f101`
  - Stop/Pause: `56ffff00 07 f102`
  - Reset: `56ffff00 07 f106`
  - Enter Debug: `56ffff00 07 f002`
  - Exit Debug: `56ffff00 07 f001`
  - Step Into: `56ffff00 07 f103`
  - Step Over: `56ffff00 07 f105`
  - Step Out: `56ffff00 07 f104`
- Does sending these commands by themselves work? vBuilder send triples...
- Does sending Enter Debug cause a flood of data like in vBuilder?
- Does Step Into work without being in debug mode? How can you tell?

Task 5: Interface Fuzzing

Level of Effort: Medium to High

Task Description: Use or create a fuzzing tool to send both valid and invalid communications to the target interface, analyzing the results and identifying anomalies. This task includes items such as password guessing, invalid input testing, data enumeration, etc.

Task Goal: Identify vulnerabilities in the interface implementation and obtain data not otherwise available from any field device vendor tool provided to the asset owner.

Sending the vFactory Commands with ctserial

- Velocio commands extracted from vFactory
  - Get Tag Count: 56ffff00 06 ac
  - Get Tag Name: 56ffff00 08 0a <2 byte number>
- What happends when you enumerate tag names past 0x05ff?
- What other commands did you collect that you can enumerate?

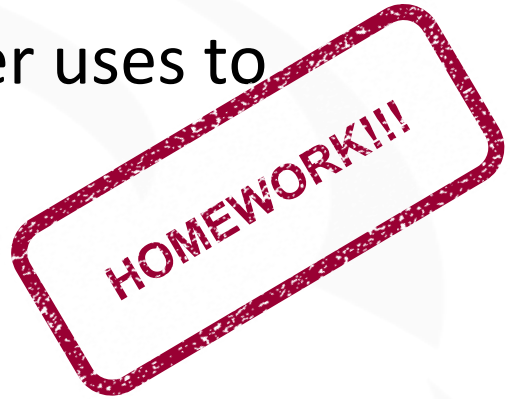
Task 6: Interface Exploitation

Level of Effort: High to Extremely High

Task Description: Based on the findings from previous tasks, determine feasible attacks that can be launched on the field technician interface. Attempt to use any authentication or cryptographic keys retrieved from one meter on a different meters to identify shared passwords and cryptographic keys.

Task Goal: Create proof of concept attacks to demonstrate the feasibility and business risks created by the discovered vulnerabilities.

Exploit Homework



- Challenge 1: Try to figure out the series of commands vBuilder uses to program your logic to the PLC
  - Hint: Start with a new project with a 1 or 2 functions in it, like 1 input switch that turns one output on and off
- Challenge 2: Try to figure out the series of commands vBuilder uses to flash a new firmware to the PLC
  - Hint: You must flash a different firmware than is currently on the PLC, such as a newer or older version, or vBuilder recognize it does not need to perform a flash operation
  - Hint: There are multiple commands/steps in the process. Make sure you know EXACTLY which bytes are firmware and which bytes are commands by comparing the bytes in the firmware you select and the bytes in the commands seen on the USB bus. Bad firmware pushes shouldn't hurt it if you don't affect the bootloader
  - Note: To replace the PLC itself (not the full PLC kit) is only $49 from Velocio.com

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, github.com/ControlThingsTools

Control Things Platform Releases:

https://goo.gl/wwqxqb