What’s the DFIRence for ICS?

Chris Sistrunk, PE
Senior Consultant, FireEye
@chrissistrunk

Josh Triplett
Senior Reverse Engineer, FireEye
Agenda

• Digital Forensics and Incident Response Overview
• DFIR for ICS
  • What’s the DFIRence?
• Embedded Devices
  • What to Collect
  • What to Analyze
• RTU Examples
  • GE D20MX
  • VxWorks DFIR Tool
  • SEL-3530 RTAC
Incident Response Overview
“Find Evil”

• Assess the situation
• Define objectives
• Collect evidence
• Perform analysis
• Communicate
• Develop remediation plan
• Document findings

http://www.cumbriafire.gov.uk/about/photo/engines/incident-response.asp
Digital Forensics Overview

• Data Collection
  • Data Files
  • OS (volatile and non-volatile)
  • Network Traffic
  • Applications
• Examination
• Analysis
• Reporting

NIST SP 800-86
Traditional DFIR tools

- Mature
  - Tools
    - Redline
    - Volatility
  - Websites
  - Cheatsheets
  - Books
What’s the DFIRence for ICS?

<table>
<thead>
<tr>
<th>IT/OT Differences</th>
<th>IT/OT Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Similar</td>
<td>Similar</td>
</tr>
<tr>
<td>Physical Processes</td>
<td>Physical Processes</td>
</tr>
<tr>
<td>Must be collected manually</td>
<td>Must be collected manually</td>
</tr>
<tr>
<td>No ICS-specific DFIR tools</td>
<td>No ICS-specific DFIR tools</td>
</tr>
<tr>
<td>ICS devices have constraints</td>
<td>ICS devices have constraints</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Steps</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assess the situation</td>
<td>• When/Where/How is the ICS affected?</td>
</tr>
<tr>
<td>Define objectives</td>
<td>• Return the ICS to normal quickly and safely</td>
</tr>
<tr>
<td>Collect evidence</td>
<td>• ICS devices have RTOS and ICS protocols</td>
</tr>
<tr>
<td>Perform analysis</td>
<td>• Analysis must be done to verify anomalies</td>
</tr>
<tr>
<td>Communicate</td>
<td>• Regularly report status to management</td>
</tr>
<tr>
<td>Develop remediation plan</td>
<td>• How/When to regain control of the ICS</td>
</tr>
<tr>
<td>Document findings</td>
<td>• Write a report of what exactly happened</td>
</tr>
</tbody>
</table>
ICS anomaly → incident?

• An anomaly of some kind has occurred
  • Increased network activity, strange behavior, failure
• Now we need to **investigate** the anomaly
• Is it known bad?
• Is it unknown bad?
• Do we **escalate** this to a security incident?
• Who do we call?
  • Engineers, Admins, PR, Safety
  • Vendors
Don’t!

HAVE YOU TRIED TURNING IT OFF AND ON AGAIN?
ICS forensics collection tools

- No ICS-specific DFIR tools...especially embedded devices
- But, we can collect data manually using other tools
Embedded devices: What to collect?

**Physical data**
- Exact location of device
- Device description
- Identifying info (manufacturer, S/N, P/N, name)
- Connections (serial, ethernet, USB)
- Front/back panel LED status
- Power consumption
- Temperature (if running hot)
- Evidence of tampering

**Digital data**
- Running configuration (including user accounts)
- Last-known good configuration
- Running firmware, approved firmware
- CPU usage %, Memory usage % (RAM, Storage)
- Running processes
- Active ports (serial, ethernet, USB, etc)
- Logs (security, events)
- Memory dump (if possible)
Embedded devices: What to analyze?  
Find Evil...or ways for evil to do evil

First Responders: ICS Engineer or Technician, Network Engineer, Vendor

- What do the user and event logs reveal? (these need to be viewed first as they may rollover)
- Does the configuration match the firmware? Is the firmware approved from FAT/SAT?
- Running config / last known good config / standard config
- Is the configuration and logic correct for the process?
- Are communications (serial, ethernet, USB, wireless) normal as compared with known good?

Vendor, Digital Forensics Specialist, Embedded Systems Analyst

- Analyze embedded OS files, captured data at rest, captured data in transit
- Volatile memory if possible (to look for code injection and potential rootkits)
Let’s do DFIR on two substation RTUs
Time to…RTFM
Data Collection: D20MX

Specs
- 667 MHz embedded PowerQUICC II Pro
- 1024 MB of 266 MHz DDR2 RAM with ECC
- 16 MB NVRAM for persistent event storage
- 8 MB boot flash, 256 MB firmware flash
- VxWorks RTOS

Tools to use
- D20MX Product Documentation Binder.pdf
- GE SGconfig software
- Terminal (Tera Term, PuTTY)
- WinSCP

D20MX Substation Controller
Chapter 11: Troubleshooting

This chapter describes how to troubleshoot:
- Serial communications
- Firmware version mismatches
- D20MX Shell commands
- D20MX Logs
You will need three manuals from the binder pdf:

1. **994-0140** D20MX Substation Controller Instruction Manual
   - Chapter 11: Troubleshooting

2. **B014-1NUG** Westmaint II+ for D20MX User’s Guide
   - Shows how to use the D20 console interface, menus, error and user logs

3. **SWM0080** D20MX Shells User’s Guide

### Shell | Prompt
--- | ---
D20M | D20M>
C | ->
CMD | [vxWorks]#
Data Collection: D20MX

Error Log and User Log

The **error log** tells what’s wrong with the configuration.

The **user log** shows logins, logouts, and all user activity. Can be exported to CSV. This data also gets put into the **syslog**.
The power of the 3 Shells

- You can access the shell remotely with SSH, but the most powerful access is through the front serial port.
- Some of these commands require assistance from GE unless you really know what you are doing.

6.3 D20MX Shells

The D20MX Shells (formally called the “68K Monitor”) are three troubleshooting and diagnostics tools that give you low-level access to your equipment, as mentioned in GE System Maintenance and Configuration Tools.

The three “shells” available on the D20MX are the “D20M” shell, the “C” shell and the “CMD” shell. The first shell, “D20M” shell, is accessed via the WESMAINT menus. Once inside the D20M shell the second shell, “C” shell, may be accessed. From there the third shell, “CMD” shell, may be accessed.

HE DOESN’T KNOW HOW TO USE THE THREE SEASHELLS
The main shell

- D20M Shell is the main shell
- Very similar to 68k monitor shell in older D20s
- Incident Responders will want to collect data from this shell
- All of the commands are explained in detail in the D20MX Shells UG
Data Collection: D20MX

• Running configuration
  ✓ Use SGConfig, ConfigPro, or TeraTerm
  ✓ Very common task
• Last-known good configuration
  ✓ Look in email, config database, engineer’s laptop, or it may be on a USB in the cabinet
• Running firmware - **img**

| D20MX>img |
| GE D20MX Build ID:93c525b9c6e92f4f1b |
| BSP v1.4/0 Created: Aug 29 2014, 20:01:56 |
| Bootrom Version: 1.4/0 [93c525b9c6e92f4f1b] <Aug 29 2014 20:02:07> |
| Firmware: SAN001/2.140 |
| JMON: 1.0.631-0002 <GE/DE> |
| value = 0 |

| D20MX>pr |
| Performance Monitor (pr) |
| CPU usage %, Memory usage % |
| pr – performance monitor |
| qr – query ram (volatile and nvram) |

<p>| D20MX&gt;qr |</p>
<table>
<thead>
<tr>
<th>Total bytes</th>
<th>Total bytes used</th>
<th>Ave. bytes used</th>
<th>Max bytes used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Free</td>
<td>97934216</td>
<td>23</td>
<td>42580094</td>
</tr>
<tr>
<td>alloc internal</td>
<td>87670768</td>
<td>11786</td>
<td>7491</td>
</tr>
<tr>
<td>cumulative alloc</td>
<td>87738516</td>
<td>16476</td>
<td>5446</td>
</tr>
<tr>
<td>Peak alloc</td>
<td>87717016</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

| D20MX>qp |
| Running processes - qp |

<table>
<thead>
<tr>
<th>NAME</th>
<th>PRI0</th>
<th>STKSZ</th>
<th>USER</th>
<th>Process</th>
<th>LIST</th>
<th>PID</th>
<th>TID</th>
<th>PARENT TID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT</td>
<td>240</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>UMLD</td>
<td>35</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>UIN</td>
<td>51</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>UMT</td>
<td>51</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BO19</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Data Collection: D20MX

Serial analyzer

• Very popular shell command (what’s Wireshark?)
• In the D20M shell, use
  ✓ `sa com#`
    where # is the port number
  ✓ Turn on logging in TeraTerm beforehand to save the traffic
  ✓ This example is DNP3
Data Collection: D20MX

- Dump memory
  - \textbf{si} – shows system information including the memory base addresses
  - \textbf{d} – dumps memory, but you have to tell it where to start and stop (only available over serial connection)
- Hand the output to someone who understands VxWorks for analysis
- Look for strings, injected code, or rootkits
Data Collection: D20MX

VxWorks C Shell
- OS level shell only accessible from the RS-232 port (access is denied from SSH)
- Mainly used by GE customer support for troubleshooting

VxWorks CMD Shell
- OS 2nd level shell, accessed by typing `cmd`
- VxWorks Kernel Shell Command Reference 6.9
- We can use some commands for forensics
  ✓ `d` (dump), `netstat`, `ipt` (firewall), `syslog`, `show devices`, `show drivers`, `show history`, `ifconfig`, `route`, and even `pcap`!

Example:
```
pcap -f /ram/temp.cap qefcc0 start
Wait a few minutes...
pcap qefcc0 stop
```
Now use a program such as WinSCP to transfer the file from the D20MX to a PC. Then use Wireshark on the PC to view the file.
Example of live memory code injection & mem dump on the D20MX

• Inject code via VxWorks C shell memory edit command `m` to simulate a rootkit

```plaintext
-> m mem,1
0x052eaa88: 0x00-de
0x052eaa89: 0x00-ad
0x052eaa8a: 0x00-be
0x052eaa8b: 0x00-ef
0x052eaa8c: 0x00-.
value = 0 = 0x0
```

• Collect volatile memory using the dump memory command `d`

```plaintext
-> d mem,8
NOTE: memory values are displayed in hexadecimal.
0x052eaa80: dead beef 0000 0000 * .........*
0x052eaa90: 0000 0000 0000 0000 * .................*
value = 0 = 0x0
```
Data Collection: VxWorks DFIR Tool – Problem

• We need tools that enable us to perform DFIR on ICS and embedded devices.
Data Collection: VxWorks DFIR Tool - Solution

A collection of utilities that enable us to:

- Read (and write) to memory on the device programmatically
  - We don’t want to have do dump memory manually
- Cache the live memory locally
  - We shouldn’t need to fetch the same memory twice to check for different issues.
- Compare the system image
  - Knowing the image is good is the first step toward looking somewhere else.
- Provide the ability to read/write and cache device data to other tools
  - Tools can be written more generically when they don’t need to worry about how to get the data
Data Collection: VxWorks DFIR Tool - Cool Features

- Can easily accommodate different transport mechanisms
  - Serial
  - TCP/Serial bridges
  - Protocols specific to other dumping utilities
- Supports caching
  - Allows resuming if connectivity is lost
  - Sparse memory dumping
- Comparative analysis works on
  - Anything that looks like a seek-able Python File Object
    - Cache Files
    - Memory Dumps
    - Sparse Memory Maps
    - Special Objects that request live memory
### Data Collection: VxWorks DFIR Tool – Validating the host image

```bash
user$ python validate_image.py --disk_image vxworks --mem_image d20mx.cache
```

<table>
<thead>
<tr>
<th>Section Name</th>
<th>Address</th>
<th>Size</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>.text</td>
<td>0x10000</td>
<td>0x393e50</td>
<td>[!!! MISMATCH !!!]</td>
</tr>
</tbody>
</table>

0017aed0 ipfirewall_start
---------------------|---------------------|---------------------|
0017aed4: lwz       | r0, 0x4c(r28)       | 0017aed4: stwu      | r1, -0x10(r1) |
0017aed8: rlwinm.   | r0, r0, 0, 0xa, 0xa | 0017aed8: li       | r3, 0        |
0017aedc: bne       | 0x17af00            | 0017aedc: addi      | r1, r1, 0x10 |
0017aeed: lwz       | r0, 0x1c(r31)       | 0017aeed0: blr      |               |

- .init$00 0x3a3e50 0x1c [MATCH]
- .init$99 0x3a3e6c 0x10 [MATCH]
- .fini$00 0x3a3e7c 0x1c [MATCH]
- .fini$99 0x3a3e98 0x10 [MATCH]
- .wrs_build_vars 0x3a3ea8 0x1c8 [MATCH]
- .sdata2 0x3a4070 0x340 [MATCH]
- .data 0x3a5000 0x55260 [!!! MISMATCH !!!]
- .sdata 0x3fa260 0x1350 [!!! MISMATCH !!!]
- .sbss 0x3fb5b0 0x7f0 [NOT_PROGBITS]
- .bss 0x3fbda0 0x155ea0 [NOT_PROGBITS]
- .PPC.EMB.apuinfo 0x0 0x18 [NOT_PROGBITS]
- .debug_ranges 0x0 0x1760 [NO_ALLOC]
- .debug_pubnames 0x0 0x7499 [NO_ALLOC]
Data Collection: VxWorks DFIR Tool - Cool Projects We Used

• CLE Loads Everything – (angr/CLE)
  • Loads our system image and provides an abstraction to a process memory space
  • Identifies architecture, endianness, etc.
  • Will soon support relocatable images (important for modules like appl.out)

• Capstone - Nguyen Anh Quynh
  • Easy access to disassemble exactly what we needed
Data Collection: VxWorks DFIR Tool – Plans for the Future

• Documentation
• Expand the tool to work on other devices
• Refine the scripts into easy-to-use modules
• Moving the code to GitHub
• Allow for feedback / feature requests / bug submissions
Data Collection: SEL-3530 RTAC

Specs
- 533 MHz Power PC
- 1024 MB DDR2 ECC RAM
- 2GB Storage
- Embedded SEL Linux

Tools to use
- SEL-3530 RTAC Instruction Manual
- SEL-5033 Instruction Manual
- SEL-5033 software
- Web Browser (Chrome, FireFox, etc)
- Terminal for SSH (Tera Term, PuTTY)
Data Collection: SEL-3530

**Digital data**
- Running configuration
- User Accounts
- Running firmware
- CPU usage %
- Memory usage %
- POST checks
- Reports (several)

**Physical Data**
- Password jumper
These are the screenshots from when I sent a malformed DNP3 message that caused the RTAC to lose the configuration.

Data Collection: SEL-3530

- Section 3: Testing and Troubleshooting
- Section 5: Web HMI and Logging
- Section 6: Security
- There are tags in the RTAC database that are assigned to help troubleshoot but are also useful for forensics as well.
- Several log types
  - SOE report
  - IED report
  - syslog
Data Collection: SEL-3530

- Example of IED Report
- Can be accessed via web or ODBC (MS Access)
- No Linux Shell
- Pros & cons
  - No SSH Interface with RTAC
  - SSH used for engineering remote access to relays
Data Collection: SEL-3530

- The RTAC can capture ethernet and serial traffic
  - SEL-5033 software and the Comm Monitor
- AG2012-15 Using Wireshark® to Troubleshoot Protocol Communications Issues on an RTAC
  - DNP3 example
- AG2015-15 Using Wireshark® to Decode RTAC Serial Line Messages and SEL Protocols
  - SEL Fast Messaging example
- SEL published several serial Wireshark dissectors
  - SELFM, Telegyr 8979
For Further Reading…

• HD Moore’s blogpost on VxWorks from 2010.
  • https://community.rapid7.com/community/metasploit/blog/2010/08/02/shiny-old-vxworks-vulnerabilities
  • Metaspoilt module for VxWorks remote memory dump (wdbrpc_memory_dump)

• David Odell’s blogpost on QNX from 2012.
  • https://www.optiv.com/blog/pentesting-qnx-neutrino-rtos

• ICS-CERT recommended practices for ICS forensics
  • https://ics-cert.us-cert.gov/sites/default/files/recommended_practices/Forensics_RP.pdf
For Further Reading…

• Travis Goodspeed’s embedded device work on the MSP430 family

• Ralph Langner’s forensics work on Stuxnet payloads for Siemens PLCs

• The Dec 23, 2015 Ukrainian Power Grid attack included writing over firmware of embedded Ethernet-serial converters.
  • [https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf](https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf)
QUESTIONS?