Challenges to Penetration Testing ICS

• **Challenge 1:** Risk to safety and reliability of processes
  – **Easiest solution:** Don't pentest production ICS networks
    • Test connectivity up to the ICS networks (enterprise perimeter and remote access)
    • Only test smaller solutions before deployment
    • Only test ICS networks before or during SAT testing, depending on risk tolerance
  – **More difficult:** Build your company's experience
    • Limit penetration test activities, scope of tests, scope of systems/networks
    • Build experience with each system/network tested, with all hands on deck the first few times

• **Challenge 2:** Hiring and/or training experienced ICS penetration testers
  – Availability of new hires and consultancies is constantly getting better
  – SANS Hosted Class: Assessing and Exploiting Control Systems course

• **Challenge 3:** Tools to use for ICS specific components and protocols
  – Use a lot of scripting wrapped around engineering tools, if you can get them
  – Use IT penetration testing tools generically in ICS or extend to add ICS features
  – Play Russian Roulette and try random tools that come up on GitHub and Google searches
Limitations of Existing ICS Tool Solutions

• Scripting wrapped around engineering tools
  – Have you ever tried using graphical macro tools to control graphical applications?!?
  – Often lack many features needed for proper penetration tests
  – Almost always limited to Windows

• Generic or extending IT penetration testing tools
  – IT penetration testing tools are overly focused on network (especially TCP/IP)
  – There are so many attack vectors (inputs) beyond TCP/IP...
  – Usually leave non-web maintenance interfaces, serial, and proprietary untested
  – Extending tools require lots of time and experience

• Random tools on GitHub and from Google searches
  – Unknown and untrusted random authors, hard to audit their code
  – Protocol tools often only implement a small subset of request types
Features Needed in Pentest Tools

- Flexible interface, for command line (remote) and graphical use
- Ability to export and import projects for liability and team use
  - History of commands used
  - History of traffic generated
- Protocol specific tools
  - Flexible client mode to send and fuzz requests
  - Server mode for response fuzzing or client tool calibration
  - Proxy mode with features such as bi-directional injection, fuzzing, and scripting
  - Ability to create, export, and import device profiles
- Generic protocol tools (RF, TCP/IP, Ethernet, Serial/Fieldbus, USB, etc...)
  - Capture mode with configurable protocol decode
  - Impersonation mode to impersonate client or server interactions
  - Macro and fuzzing features to facilitate testing
ControlThings Tools (www.controlthings.io/tools)

**ControlThings Serial**
An assessment tool to interact with binary serial devices, allowing you to impersonate ICS vendor tools

**ControlThings Modbus**
An assessment tool to interact with Modbus devices, both TCP/UDP and serial (RTU/ASCII)

**ControlThings SPI**
An assessment tool to interact with EEPROMs, Flash chips, and other embedded chips using SPI

**ControlThings I2C**
An assessment tool to interact with EEPROMs, Flash chips, and other embedded chips using I2C

**ControlThings Velocio**
An assessment tool to interact with Velocio PLCs, and an example of tools we plan to create for other vendors

**ControlThings UI**
The library that make these tools possible, made freely available to build your own tools
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>read_coils</td>
<td>Read digital outputs in format: 30,50,70-99,105.</td>
</tr>
<tr>
<td>read_discrete_inputs</td>
<td>Read digital inputs in format: 30,50,70-99,105.</td>
</tr>
<tr>
<td>read_holding_registers</td>
<td>Read digital inputs in format: 30,50,70-99,105.</td>
</tr>
<tr>
<td>read_id</td>
<td>Read device identification data.</td>
</tr>
<tr>
<td>read_input_registers</td>
<td>Read analog inputs or internal registers in format: 30,50,70-99,105.</td>
</tr>
</tbody>
</table>
CT Modbus Read Holding Registers

2019-03-18 12:35:38.805784 - Modbus Function 3, Read Holding Registers

1:  0078 006e 0082 0000 0000 0000 0000 0000
9:  0000 0000 0000
17: 0000 0000 0000
25: 0000 0000 0000
33: 0000 0000 0000
41: 0000 0000 0000
49: 0000 0000 0000
57: 0000 0000 0000
65: 0000 0000 0000
73: 0000 0000 0000
81: 0000 0000 0000
89: 0000 0000 0000
97: 0000 0000 0000

Modbus Function 3, Read Holding Registers: 1-100

<table>
<thead>
<tr>
<th>Addr</th>
<th>Int</th>
<th>HEX</th>
<th>ASCII</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>120</td>
<td>0078</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>110</td>
<td>006e</td>
<td>n</td>
</tr>
<tr>
<td>3</td>
<td>130</td>
<td>0082</td>
<td>&lt;82&gt;</td>
</tr>
<tr>
<td>4-100</td>
<td>0</td>
<td>0000</td>
<td>@</td>
</tr>
</tbody>
</table>
CT Modbus Commands (1 of 2)

> connect tcp:10.10.10.1
> connect rtu:/dev/serial
> connect ascii:com2
> connect udp:10.10.10.1:10502
> read id
> read discrete_inputs 1
> read coils 1,3,5,7
> read input_register 5,10-30,90-99
> read holding_register 50 9
> write holding_register 1000 14302 188 305
> write holding_register 1000 "My name is Mud"
> write holding_register 1400 DEADBEEF
> poll holding_register 1-10,15-19 1
> tags add input1 input_register 1
> tags add config2 holding_register 50-69
> tags add config3 holding_register 70 20

# start a client session
# works with serial too
# and and windows
# even udp with custom ports
# read device identifiers
# read coils and registers
# with comma separated values
# and ranges
# or start address and count
# write single values
# or multiple values
# registers support int
# and strings
# or raw hex
# poll registers every second
# define tag names
# tags can define ranges
# and work with start & count
CT Modbus Commands (2 of 2)

- read tags input1 config2 config3
- tags group configs config1 config2 config3
- tags export saved.tags
- tags import saved.tags
- clone tcp:10.10.10.10 coils 1-100
- clone tcp:10.10.10.10 all 1-100
- simulate tcp:127.0.0.1:10502
- proxy tcp:10.10.10.1:10502 rtu:com4
- function 33 0000 DEADBEEF
- function 8 [0000-FFFF] 0000
- function 8 [0000-00FF] (0000)5
- raw 1234 0001 06 01 0000 0010
- tunnel listen tcp::6666
- tunnel connect tcp:10.1.1.1:6666
- tunnel send exfiltration.txt
- tunnel shell
- historian tcp:10.1.1.1:9300

# tags simplify reads/writes
# create tag groups
# export and share tags
# import other's tags
# clone coils from a device
# or all types of values
# so you can later simulate
# proxy requests to device
# send custom functions
# brackets for enumeration
# parenths for random fuzzing
# or full raw modbus payloads
# setup modbus tunnel service
# connect from another comp
# send files through tunnel
# or open a terminal session
# transactions to cthistorian
# ControlThings Tools Roadmap

<table>
<thead>
<tr>
<th>Tool Name</th>
<th>Availability</th>
<th>Projects</th>
<th>Client</th>
<th>Server</th>
<th>Proxy</th>
<th>Capture</th>
<th>Profiles</th>
<th>Fuzzing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Generic Tools</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ctserial</td>
<td>YES</td>
<td>✔</td>
<td>✔</td>
<td>In Dev</td>
<td>In Dev</td>
<td>In Dev</td>
<td>✔</td>
<td>In Dev</td>
</tr>
<tr>
<td>ctethernet</td>
<td>In Dev</td>
<td>✔</td>
<td>✔</td>
<td>Planned</td>
<td>Planned</td>
<td>✔</td>
<td>Planned</td>
<td>Planned</td>
</tr>
<tr>
<td>ctip</td>
<td>In Dev</td>
<td>✔</td>
<td>✔</td>
<td>Planned</td>
<td>Planned</td>
<td>✔</td>
<td>Planned</td>
<td>Planned</td>
</tr>
<tr>
<td>ctrf</td>
<td>In Dev</td>
<td>✔</td>
<td>In Dev</td>
<td>In Dev</td>
<td>Planned</td>
<td>In Dev</td>
<td>In Dev</td>
<td>Planned</td>
</tr>
<tr>
<td><strong>Protocol Specific Tools</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cti2c</td>
<td>YES</td>
<td>✔</td>
<td>✔</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>In Dev</td>
<td>Planned</td>
</tr>
<tr>
<td>ctspi</td>
<td>YES</td>
<td>✔</td>
<td>✔</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>In Dev</td>
<td>Planned</td>
</tr>
<tr>
<td>ctmodbus</td>
<td>YES</td>
<td>✔</td>
<td>✔</td>
<td>In Dev</td>
<td>In Dev</td>
<td>In Dev</td>
<td>In Dev</td>
<td>✔</td>
</tr>
<tr>
<td>ctprofinet</td>
<td>Planned</td>
<td>✔</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
</tr>
<tr>
<td>ctethernetip</td>
<td>Planned</td>
<td>✔</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
</tr>
<tr>
<td>ctdnp3</td>
<td>Planned</td>
<td>✔</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
</tr>
<tr>
<td>ct104</td>
<td>Planned</td>
<td>✔</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
</tr>
<tr>
<td>cticcp</td>
<td>Planned</td>
<td>✔</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
</tr>
<tr>
<td>ctopc</td>
<td>Planned</td>
<td>✔</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
</tr>
<tr>
<td>ctbluetooth</td>
<td>Planned</td>
<td>✔</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
</tr>
<tr>
<td>ctwifi</td>
<td>Planned</td>
<td>✔</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
<td>Planned</td>
</tr>
</tbody>
</table>
ControlThings Platform (www.controlthings.io/platform)
Author Contact Information

Justin Searle

personal: justin@controlthings.io
work: 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 and Tools Releases:
https://www.controlthings.io