A Trusted Approach to Design a Network Monitor

Koichi Shimizu, Teruyoshi Yamaguchi, Tsunato Nakai, Takeshi Ueda, Nobuhiro Kobayashi and Benoît Boyer
• Background
• Whitelisting Network Monitor
• Trusted Approach for Whitelisting
  – Model-based
  – Automation
  – Verification
• Conclusion
Cyber attacks against industrial control systems (ICS)

- A rise after Stuxnet in 2010 and high ever since
- New one always appears
- Power outage in Ukraine(2015,2016), Operation Ghoul(2016), ...
Background

• ICS has its own requirements
  – Availability
    • Security must not slow nor stop the service
  – Long-term operation
    • Security must be effective throughout the lifetime

<table>
<thead>
<tr>
<th></th>
<th>General IT systems</th>
<th>ICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Targets of security</td>
<td>Information</td>
<td>Facilities and Service</td>
</tr>
<tr>
<td>Priority of security</td>
<td>Confidentiality</td>
<td>Availability</td>
</tr>
<tr>
<td>Lifetime</td>
<td>3-5 years</td>
<td>10-20 years</td>
</tr>
<tr>
<td>Operating time</td>
<td>Business hours</td>
<td>24 hours, 365 days</td>
</tr>
</tbody>
</table>
Whitelisting Network Monitor

- Whitelisting network monitor is suitable for ICS
  - Whitelist interprets deviations from the normal behavior as attacks

**Whitelist**
- List of "Allow", all the rest denied
- Potential to detect new attacks

**Blacklist**
- List of "Deny", all the rest allowed
- Accurate detection of known attacks
Whitelisting Network Monitor

• Why suitable for ICS?
  – Availability:
    • Pattern matching is lightweight
    • Whitelist doesn't need updating once defined
    • cf. antivirus software
  – Long-term operation:
    • Potential to detect new attacks in the future
• Consider a simple ICS
  – Consists of HMI, PLC and field devices
  – HMI/PLC sends commands/responses via LAN
  – Fixed "normal behavior" of commands/responses

HMI: Human Machine Interface
PLC: Programmable Logic Controller
## Normal behavior as a whitelist

<table>
<thead>
<tr>
<th>State</th>
<th>Protocol</th>
<th>Sender</th>
<th>Receiver</th>
<th>Data length</th>
<th>Command</th>
<th>Payload condition</th>
<th>Period [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>Control</td>
<td>192.168.0.10</td>
<td>192.168.0.20 59306</td>
<td>1024</td>
<td>QuerySpeed</td>
<td>—</td>
<td>5</td>
</tr>
<tr>
<td>Operation</td>
<td>Control</td>
<td>192.168.0.10</td>
<td>192.168.0.20 59306</td>
<td>2048</td>
<td>ChangeSpeed</td>
<td>0〜90</td>
<td>50</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Motivation for a trusted approach
• A mistake in the whitelist causes a low detection rate or false positiveness
• A bug in the network monitor causes the same problem
• How to be confident about the network monitor?

Trusted Approach for Whitelisting
Trusted Approach for Whitelisting

• Proposal
  – Use of a model-based development framework
    • To clearly define a normal traffic specification
    • In the future, to be integrated into usual model-based development of industrial embedded software
  – Automation
    • The whitelist and the network monitor program are automatically generated to avoid manual mistakes
  – Verification
    • The model and the network monitor program are verified to ensure that there are no mistakes or bugs
Trusted Approach for Whitelisting

Workflow

Modelling → Model (Simulink) → Auto gen → Whitelist

Verif → Model checking

Auto gen of a verifiable network monitor → Network monitor (C language code) → Compile and install to the target

Verif → Static program analysis

Legends:
- Info or data
- Process
Trusted Approach for Whitelisting

Workflow

Modelling → Model (Simulink) → Auto gen → Whitelist

Verif → Model checking

Auto gen of a verifiable network monitor

Network monitor (C language code)

Compile and install to the target

Verif

Static program analysis

Legends: Info or data, Process
Modelling of a control system

Toy example (again)

- Consists of HMI, PLC and field devices
  - but field devices are out of scope of modelling
- HMI/PLC sends commands/responses via LAN

![Diagram of control system](image-url)

FormalISE 2017
Modelling of a control system

Definition of system states and commands in use

- Just for ease
  - Three system states
  - Only "Operation" has a set of commands in use

<table>
<thead>
<tr>
<th>System state</th>
<th>Command in use</th>
<th>Parameter</th>
<th>Period [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>ChageSpeed</td>
<td>0～90</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>QuerySpeed</td>
<td>Device number</td>
<td>5</td>
</tr>
<tr>
<td>Maintenance</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Abnormal</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>
## Modelling of a control system

### Resulting whitelist (again)

<table>
<thead>
<tr>
<th>State</th>
<th>Protocol</th>
<th>Sender</th>
<th>Receiver</th>
<th>Data length</th>
<th>Command</th>
<th>Payload condition</th>
<th>Period [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>Control</td>
<td>192.168.0.10</td>
<td>192.168.0.20</td>
<td>1024</td>
<td>QuerySpeed</td>
<td>Device number</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>any</td>
<td>59306</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td>Control</td>
<td>192.168.0.10</td>
<td>192.168.0.20</td>
<td>2048</td>
<td>ChangeSpeed</td>
<td>0〜90</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>any</td>
<td>59306</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Modelling of a control system

Devices and connections between them at top level
Modelling of a control system

Control model and communication model inside each device

Control model

Communication model
Modelling of a control system

Control model
- Defines control behaviour
- Not needed for whitelist generation

To communication model

COMM ON
COMM OFF
ABNORMAL
MAINTENANCE

Control model

FormaliSE 2017
Modelling of a control system

Communication model is a state machine that defines:

- **System states and transitions between them**
- **Command sequence under each system state**
Modelling of a control system

Communication model is a state machine that defines
• System states and transitions between them
• Command sequence under each system state

![State Machine Diagram]

- Maintenance
- Operation
- Abnormal

- Query Speed
- Change Speed
• Modelling of a period condition in Simulink
  – The guards "after" and "before" are close but not sufficient

  – Using global timers is too ad-hoc
Trusted Approach for Whitelisting

Workflow

1. Modelling
2. Model (Simulink)
   - Auto gen
   - Verif
3. Whitelist
4. Compile and install to the target

- Auto gen of a verifiable network monitor
- Network monitor (C language code)
- Static program analysis
- Verif

Legends:
- Info or data
- Process

© Mitsubishi Electric Corporation
Automated generation of the whitelist

- Extract command data from communication model
  - From cmd_out of HMI to cmd_in of PLC
  - Read1 is sent in the following example

```plaintext
Comm model on HMI side
... / {cmd_out.data = Read1; send(cmd_out)}

Comm model on PLC side
cmd_in[cmd_in.data == Read1] / ...
```
Automated generation of the whitelist

Extracted data parsed, formatted and converted to whitelist

<table>
<thead>
<tr>
<th>Device num=&quot;2&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMI</td>
</tr>
<tr>
<td>PLC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IP num=&quot;2&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP, HMI, 192.168.0.10</td>
</tr>
<tr>
<td>IP, PLC, 192.168.0.20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Communication device=&quot;HMI&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>74: SYS_Maintenance</td>
</tr>
<tr>
<td>75: SYS_Operation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State num=&quot;26&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>74: SYS_Maintenance</td>
</tr>
<tr>
<td>75: SYS_Operation</td>
</tr>
</tbody>
</table>

Conversion to whitelist using detailed information at implementation level

FormaliSE 2017
Trusted Approach for Whitelisting

Workflow

Modelling → Model (Simulink) → Auto gen → Whitelist

Model checking

Verif

Auto gen of a verifiable network monitor

Network monitor (C language code) → Compile and install to the target

Static program analysis

Verif

Legends

Info or data  Process
Automated generation of the network monitor

1- Parsing

Abstract Syntax Tree

Whitelist

2- Generating Frama-C annotations

3- Building Decision Tree

Whitelist as a Decision Tree

4- C data generation

Network monitor (C language code)

monitor.c

spec.h

rules.h

Decision Tree Traversal (generic) For Verification For C Implementation
Automated generation of the network monitor

Whitelisting as a decision tree

System state
- State
- Operation

Network packet data
<table>
<thead>
<tr>
<th>Sender IP</th>
<th>Receiver IP</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.10</td>
<td>192.168.0.20</td>
<td>Read2</td>
</tr>
</tbody>
</table>

Decision tree encoding the whitelist

- State
- Maintenance
- Operation
- Abnormal

- Sender IP
  - 192.168.0.10
  - 192.168.0.20

- Receiver IP
  - 192.168.0.20
  - 192.168.0.10

- Command
  - Write
  - Read1
  - Read2
Verification of the model

The model is the starting point of the trusted monitor, so needs verification by model checking.

The process involves:

1. Modelling (Simulink)
2. Auto gen
3. Whitelist
4. Model checking

Auto gen of a verifiable network monitor

Network monitor (C language code)

Compile and install to the target

Static program analysis

Legends:
- Info or data
- Process
Verification of the model

- Experiment using Simulink Design Verifier (SLDV)
  - Run-time errors and dead logics can be detected
  - The latter is of some use
  - Proof of properties defined in Simulink
    - Takes a lot of time for a model that has a relatively large state machine of our purpose
- As a result, SLDV is not sufficient for our purpose
Verification of the network monitor

The network monitor is responsible for detection and must be free of bugs

- Modelling
- Model (Simulink)
  - Auto gen
- Whitelist
  - Model checking
  - Verif
- Auto gen of a verifiable network monitor
  - Network monitor (C language code)
  - Compile and install to the target
  - Verif

Static program analysis
Verification of the network monitor

1- Parsing

Abstract Syntax Tree

2- Generating Frama-C annotations

Whitelist

3- Building Decision Tree

Whitelist as a Decision Tree

4- C data generation

Network monitor (C language code)

- monitor.c
- spec.h
- rules.h

Decision Tree Traversal (generic)  For Verification  For C Implementation
Verification of the network monitor

• Frama-C/WP
  – Static analysis and verification of C programs
  – The WP plugin for formal reasoning
  – The following example asserts
    • i < MAX_INT to avoid integer overflow
    • the pointer "data" is valid

```c
//@assert signed_overflow: i+1<=2147483647;
i++;
//@assert mem_access: ¥valid_read(data);
if (*data == 1) {
  ...
}
```
Verification of the network monitor

Whitelisting as a decision tree

System state

<table>
<thead>
<tr>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
</tr>
</tbody>
</table>

Network packet data

<table>
<thead>
<tr>
<th>Sender IP</th>
<th>Receiver IP</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.10</td>
<td>192.168.0.20</td>
<td>Read2</td>
</tr>
</tbody>
</table>

Decision tree encoding the whitelist

- **State**
  - Maintenance
  - Operation
  - Abnormal

- **Sender IP**
  - 192.168.0.10
  - 192.168.0.20

- **Receiver IP**
  - 192.168.0.20
  - 192.168.0.10

- **Command**
  - Write
  - Read1
  - Read2
  - Write response
  - Read1 response
  - Read2 response
Verification of the network monitor

```c
/*@requires ¥valid((tree_t*) rules) && valid_tree_t(*rules);
@requires parsed;
@ensures ¥result == 1 ==> matched;
@assigns ¥nothing; */
int monitor(void);
```
Verification of the network monitor

@predicate rule_2 =
  state == 0 &&
  send_info[IP] == 0x0a80001 &&
  0 <= send_info[TCP] <= 65535 &&
  recv_info[IP] == 0xc0a80014 &&
  recv_info[TCP] == 0x0c &&
  command == 0x2112 &&
  timers[0] == 5;
Unverified parts of the workflow

No verification about conversion is not a big issue

Modelling (Simulink) → Model checking → Auto gen → Whitelist

Auto gen of a verifiable network monitor

Network monitor (C language code) → Compile and install to the target

Static program analysis

FormalSE 2017

Legends: Info or data, Process
Conclusion

• Proposal
  – Trusted whitelisting network monitor
    • Automated generation of the whitelist based on a model-based development framework, where the model can be verified
    • The network monitor is automatically generated from the whitelist and can be proven to be free of bugs
  – Open problems / future works
    • Modelling: period conditions can't be encoded but are supposed to be given as an implementation detail
    • Automation: whitelist generation is not fully automated
    • Verification: model verification needs a different tool
    • Others: evaluation of the detection rate, etc.