SIDE-CHANNELS FOR CROSS-LAYER SECURITY: A HARDWARE PERSPECTIVE

6TH INTERNATIONAL WORKSHOP ON CYBERSECURITY
JR HAKATA CITY, FUKUOKA, JAPAN
JAN 22, 2018
https://cs.kyushu-u.ac.jp/sicorp/en/info/6th-workshop/

Presented by
Ryan Robucci
Associate Professor
Computer Science and Electrical Engineering
University of Maryland Baltimore County, US
CYBER-PHYSICAL SYSTEMS (CPS) SECURITY

A COLLABORATION BETWEEN UMBC AND UNITED STATES NAVAL ACADEMY (USNA)

UMBC Collaborators –
Prof. Ryan Robucci, Prof. Chintan Patel,
Prof. Nilanjan Banerjee, Prof. Anupam Joshi
Ph.D. Student Collaborators –
Deepak Krishnankutty, Brien Croteau, Zheng Li

USNA Collaborators –
Prof. Kiriakos Kirikadis, Prof. Tracie Severson,
Prof. Erick Rodriguez-Seda

UMBC is attacking the cyber threat from the chip-level
"Micro"-level Detection
USNA is attacking the cyber threat from the process-level
"Macro"-level Detection

This work has been supported in part by the U.S. Office of Naval Research under Awards N00014-15-1-2179 and N0001417WX01442
LEVELS COMPRISING CPS

We seek to combine efforts to monitoring at multiple levels to succeed in security where a layer-wise (layer-by-layer) approach may fail.

Sensors can be considered as IoT devices.
## Layers of Vulnerabilities

<table>
<thead>
<tr>
<th>System Layer</th>
<th>Example Vulnerabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCS (e.g., Vehicular Systems)</td>
<td>Compromised Nodes: infiltration by computer code or physical means (e.g., AoD through CAN bus)</td>
</tr>
<tr>
<td>Node Firmware</td>
<td>Compromised Software, back-door Vulnerability</td>
</tr>
<tr>
<td>Node Hardware</td>
<td>Hardware Trojans, fake/cloned ICs, Supply Chain</td>
</tr>
<tr>
<td>ICs</td>
<td>Side Channel Attacks, IC Trojans</td>
</tr>
</tbody>
</table>

Increasing Scale and Abstraction
SIDE-CHANNEL LEAKAGE: LEAKAGE THROUGH LAYERS

• Modern platforms must be versatile to be marketable; hardware is programmable (e.g. software), typically at the **cost of performance**

• Programmability and versatility can also **compromise security**
  • Fixed security hardened hardware can provide what is known as a hardware root of trust
  • Programmable hardware requires configuration security validation (e.g. software verification)

• (Programmability does allow security patches but it is still good to harden in the initial design as much as possible instead of relying on updates)

SIDE-CHANNEL LEAKAGE: LEAKAGE THROUGH LAYERS, DANGER OF ABSTRACTION

A hardware-software system is built by layers/levels and with the design of each additional level usefulness and security should be considered.

Usually one “use case” is not defined, so excess functionality is provided in the interface.

We also leverage abstraction, so that high-level design optimization can be performed without knowing the details below – this again represents excess capability that can be exploited.
KEY POINTS CROSS-LAYER VULNERABILITIES AND SECURITY

• Abstraction is fundamental to system design: Cross-Layer Design Optimization (Performance) is difficult

• Cross Layer Security Considerations are difficult too

• Mistakes in system function (bugs) related to the interaction of layers designed by different engineers is common, and also likely where security holes are left for attackers

• Summary: Abstraction and “Layer-Wise” Security are prone leave openings for attackers - we will consider at set called “side-channels”
GENERALIZED CHANNELS: EXCESS AND UNINTENDED INPUTS AND OUTPUTS

Unintended/Undesired IO that allow us to learn information
PHYSICAL CHANNELS

- Current
- Voltage
- EM Emissions
- Timing
- Sidechannel Leakage

Data In

Data Out
JTAG programming
/debug Port
(sometimes undocumented)

Manipulating
- Data Timing
- Power Levels

Timing

Current, Voltage

EM Emissions

Sidechannels
GOALS FOR POWER-SUPPLY SIDE-CHANNEL ANALYSIS

Studying and Leveraging Power-Supply Side-Channel Analysis allows

- Stage attacks for secret key extraction
- Vulnerability Analysis to Defend against attacks
- **Cross-Layer Security: Novel Protection at Multiple Levels** and **Across levels**, such as for Malware Detection through power-supply measurements
  - The “physicality” of many side-channels make them a difficult to spoof
TRACKING

• What tracks can be used?
  • Sound?
  • Visual?
  • etc...

• Multiple tracks to be hidden, multiple ways to monitor and verify activity
• What about a power supply measurement?
ELECTRONIC FOOTPRINTS: HOME USE DETECTION USING POWER
ENTROPY OF A DISCRETE VARIABLE (QUANTIFYING UNCERTAINTY)

• Entropy related to:
  • Number of possibilities
  • Probability of each possibility

\[ \sum_{i=1}^{N} p_i \log_2 \left( \frac{1}{p_i} \right) \]

• Example
  • Assume unknown 6 bit secret key – 64 possibilities
  • Assuming all possibilities are equal, each key has probability \( p = \frac{1}{64} \)

\[ \sum_{i=1}^{64} \frac{1}{64} \log_2 \left( \frac{1}{\frac{1}{64}} \right) = 6 \text{ bits} \]
QUANTIFYING GAINED INFORMATION:
SIDE-CHANNELS REPRESENT MUTUAL INFORMATION

• Mutual Information – how to quantify how much knowing one variable reduces the uncertainty of another

• Example: We make an indirect measurement (e.g. power supply), and based on that know that the first bit of the key is “1” with 80% probability and “0” with 20% probability.

\[ \sum_{i=1}^{32} 0.8 \left( \frac{1}{32} \right) \log_2 \frac{1}{0.8 \left( \frac{1}{32} \right)} + \sum_{i=1}^{32} 0.2 \left( \frac{1}{32} \right) \log_2 \frac{1}{0.2 \left( \frac{1}{32} \right)} \]

\[ = 5.72 \text{ bits} \]

• How much information was learned?

6 - 5.72 bits = .32 bits learned

• From this example we understand that we can QUANTIFY information learned not only when we find a definitive truth, but any time we reduce our uncertainty about which possibility might be true.
POWER SUPPLY SIDE CHANNELS
MULTI-VANTAGE POINT IC MONITORING

Not unlike the “telegraph”

Hypothesis: Multi-point monitoring yields more information
A digital chip can be partitioned into two subsystems:
- The linear Power Grid Circuit
- The non-linear Core Logic Circuit

DIFFICULT TO MODEL ENTIRE SYSTEM -> DIFFICULT TO PREDICT SIDE CHANNELS

We’ll rely on an experimental system and data-driven approaches augmented by domain expertise.
PROTOTYPE PLATFORM: FPGA-BASED EMULATION

- Spartan 6
- Spartan 3E
- On-Board ADC
- Transient Current Sensor 1
- Transient Current Sensor 2
- Transient Current Sensor 3
- Transient Current Sensor 4
ANALYSIS TESTBEDS WITH MSP430 AS WELL
DISCOVERING A SECRET USING SIDE-CHANNELS AND HIDDEN MODEL HYPOTHESIS TESTING – DES CRYPTO KEY EXTRACTION

• given an input and an output, can a model test be performed where the model is based on a secret?
• If so, and a limited number of hypothesis must be tested then a secret can be exposed

MODEL TEST: $P(\text{Output}|\text{Input}, \text{Hidden Model n})$?
An interesting result: For multiple implementations of the same intended function (relationship of indented digital I/O)

- Different power pins showed different data
- Different implementations showed different characteristics in the side channel

We then asked what can be inferred from side channels besides data?

- Which implementations?
- What Activity?
- Can it be modeled?
FROM DETECTING DATA THEN HARDWARE TO DETECTING SOFTWARE

• Using a data-driven approach (avoids complexity of physical modeling), we were able to determine from which implementation a side-channel arose even when no observable functional change existed in the intended IO

So then we asked? What could we determine about software running on hardware?
WE COULD FIND INSTRUCTIONS IN A SEQUENCE USING POWER SUPPLY MEASUREMENTS....
WE COULD FIND INSTRUCTIONS IN A SEQUENCE USING POWER SUPPLY MEASUREMENTS....

EVEN WHEN THEY MOVED
LEAKAGE BETWEEN DISPARATE LAYERS

- High-Level Algorithm
  - Assembly
    - Processor Behavior
      - Digital Blocks
        - Digital Gate Abstraction
          - Transistor+RLC Circuit

\[ \text{Transistor+RLC Circuit} \rightarrow \text{Digital Gate Abstraction} \rightarrow \text{Digital Blocks} \rightarrow \text{Processor Behavior} \rightarrow \text{Assembly} \rightarrow \text{High-Level Algorithm} \]

\[ \text{Cross-Layer Inference} \rightarrow \text{Side Channel} \]

- This can be a powerful tool to help monitor a system at
- additional levels in a CPS
- and to verify
  - across layers (validate one layer by measuring another)
CLASSIFYING INSTRUCTIONS FROM A PIPELINE PROCESSOR REQUIRES FINDING TEMPORAL BOUNDARIES

- Effective classification requires determining the start and stop of the instructions.
NEED A WAY TO TEST BOUNDARIES

• We derived a fitness function for a set of boundaries using a “coarse” instruction classifier (clustering based on understanding of what hardware various instructions use)

• One course classifier was derived for each instruction execution length: CCPI Clock Cycles per Instruction
METHODOLOGY (OVERVIEW)

We note power signature observations depend on

- Hardware dependent variations
- Data dependent variations

![Diagram of METHODOLOGY]

**Training**
- Capture power profiles for N-CCPI instruction
- Generate PCA template
- Template classification

**Testing**
- Capture power profiles for test sequence
- Window and generate PCA templates
- Determine minimum Euclidean distances
- Classify instruction Templates
- Dynamic programming for optimal clock cycle sequence determination
TRAINING ROBUSTNESS OF CLASSIFIERS

- Power supply depends on
  - the instruction (the activated hardware)
  - the data
  - So, train with varied data
- Pipeline means more than one instruction is handled at a time
  - -> Train with different instructions entering and preceding and following the instruction of interest

```
mov r8,r9
inc r8
mov r8,r9
inc r8
mov r8,r9
inc r8
mov r8,r9
inc 30(r10)
inc r8
mov r8,r9
inc 30(r10)
inc r8
mov r8,r9
inc 30(r10)
inc r8
```
COARSE CLASSIFIERS BASED ON COMPUTATION AND ADDRESSING MODE HARDWARE

Raw Time Series Data (t Captures) for N-CCPI instruction

\[ N = \{1,2,3,4,5,6\} \]

PCA Templates (N-CCPI)

Averaged PCA Template (N-CCPI)

Template Classification

<table>
<thead>
<tr>
<th>#</th>
<th>Label</th>
<th>Interpretation (Associated Traits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>reg_reg</td>
<td>Source Register, Destination Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Includes Arithmetic and logical instructions</td>
</tr>
<tr>
<td>2</td>
<td>mem_mem_sub</td>
<td>Source Memory, Destination Memory (Complement hardware involved - includes ‘bic’ instruction)</td>
</tr>
<tr>
<td>3</td>
<td>mem_mem_nosub</td>
<td>Source Memory, Destination Memory (No complement hardware)</td>
</tr>
<tr>
<td>4</td>
<td>reg_const_ind_sub</td>
<td>Source Register or a constant, Destination Memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Destination includes a constant and involves indirect addressing</td>
</tr>
<tr>
<td>5</td>
<td>reg_const_ind_nosub</td>
<td>Source Register or a constant, Destination Memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indirect addressing at source, No subtraction hardware</td>
</tr>
<tr>
<td>6</td>
<td>ind_reg_sub</td>
<td>Source Memory, Destination Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indirect addressing at source, Subtraction hardware Used</td>
</tr>
<tr>
<td>7</td>
<td>ind_reg_nosub</td>
<td>Source Memory, Destination Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indirect addressing at source, No subtraction hardware</td>
</tr>
<tr>
<td>8</td>
<td>mem_reg</td>
<td>Source Memory, Destination Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Includes Arithmetic and logical instructions</td>
</tr>
<tr>
<td>9</td>
<td>const_reg</td>
<td>Source is a generated constant, Destination Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Includes Arithmetic and logical instructions</td>
</tr>
<tr>
<td>10</td>
<td>imm_reg_sub</td>
<td>Source involves a constant, Destination Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Complement hardware involved - includes ‘bic’ instruction)</td>
</tr>
<tr>
<td>11</td>
<td>imm_reg_nosub</td>
<td>Source involves a constant, Destination Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(No complement hardware)</td>
</tr>
<tr>
<td>12</td>
<td>imm_ind_sub</td>
<td>Source involves a constant, Destination Memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indirect addressing involving constant at destination</td>
</tr>
<tr>
<td>13</td>
<td>imm_ind_nosub</td>
<td>Source involves a constant, Destination Memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indirect addressing involving constant at destination, No complement hardware</td>
</tr>
<tr>
<td>14</td>
<td>other</td>
<td>Uncategorized</td>
</tr>
</tbody>
</table>
USE OF PCA TO REDUCE DIMENSIONALITY

Raw Time Series Data ($t$ Captures) for $N$-CCPI instruction

$N = \{1,2,3,4,5,6\}$

$\mathbf{t}$ PCA Templates ($N$-CCPI)

1 Averaged PCA Template ($N$-CCPI)

4 Clock Cycle Templates
COARSE INSTRUCTION CLASSIFICATION PERFORMANCE
(FPGA RESULTS SHOWN, ACTUAL MSP430 ALSO TESTED)

Classification over 20,000 instances of each instruction for specific CCPI

![Heatmap Diagram]

- const_reg
  - 97.7
  - 2.3
- reg_reg
  - 0.0
  - 100.0
- mem_reg
  - 100.0
  - 0.0
  - 0.0
- imm_reg_nosub
  - 0.1
  - 79.4
  - 20.5
- imm_reg_sub
  - 0.0
  - 4.7
  - 95.3
- reg_const_ind_sub
  - 99.7
  - 0.3
- reg_const_ind_nosub
  - 14.6
  - 85.4
- mem_mem_nosub
  - 95.7
  - 4.2
  - 0.1
  - 0.0
- imm_ind_nosub
  - 0.0
  - 85.1
  - 0.0
  - 14.9
- mem_mem_sub
  - 0.1
  - 0.3
  - 96.9
  - 2.7
- imm_ind_sub
  - 0.0
  - 27.8
  - 0.0
  - 72.2
TESTING METHODOLOGY

Raw Time Series Data ($t'$ Captures)

Windowing,
Generate 6 PCA Template Sets

1-Clock Cycle Window

Current (mA)

Clock Cycles
TESTING METHODOLOGY

Raw Time Series Data (t' Captures)

Windowing, Generate 6 PCA Template Sets
TESTING METHODOLOGY

Raw Time Series Data (t' Captures)

Windowing,
Generate 6 PCA
Template Sets

6-Clock Cycle Window

Clock Cycles

Current (mA)
TESTING METHODOLOGY

Raw Time Series Data (\(t'\) Captures)

Windowing,
Generate 6 PCA Template Sets

6 Averaged PCA Templates

6 Minimum Euclidean Distances

1-Clock Cycle Window

Min Distance (COST) Table

Clock Cycle of Start of an Instruction

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Min Distances

35
**TESTING METHODOLOGY**

- **Raw Time Series Data** (t' Captures)
- **Windowing**
- **Generate 6 PCA Template Sets**
- **6 Averaged PCA Templates**
- **6 Minimum Euclidean Distances**

**Min Distance (COST) Table**

<table>
<thead>
<tr>
<th>CCPI of Instruction</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>3</td>
<td>-</td>
<td>35</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>8</td>
<td>6</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
TESTING METHODOLOGY

Raw Time Series Data (t' Captures)

Windowing, Generate 6 PCA Template Sets

6 Averaged PCA Templates

6 Minimum Euclidean Distances

Dynamic Programming

Detected Sequence

<table>
<thead>
<tr>
<th>CCPI of Instruction</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>3</td>
<td>-</td>
<td>37</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>8</td>
<td>6</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
TESTING METHODOLOGY

- Raw Time Series Data ($t'$ Captures)
- Windowing, Generate 6 PCA Template Sets
- 6 Averaged PCA Templates
- 6 Minimum Euclidean Distances
- Dynamic Programming
- Detected Sequence

Min Distance (COST) Table
Clock Cycle of Start of an Instruction

- Optimization Puzzle
  - 3,4,5,3 $\rightarrow$ 2+2+1+2 = 7
  - 4,2,5,4 $\rightarrow$ 2+1+2+3 = 8
RESULTS

Analysis was performed on waveforms averaged from 10,000 captures of the power supply signature for each test sequence. The most common issue is a 2-cycle classified as two 1-cycle instructions.

<table>
<thead>
<tr>
<th>Sequence 1</th>
<th>C</th>
<th>C</th>
<th>Power Pin</th>
<th>Optimal Clock-Cycle Sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>pop_mem_reg</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>add_mem_mem_nosub</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>inc_reg_const_ind_nosub</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>mov_mem_ind_nosub</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>add_reg</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>sub_mem_mem_sub</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>dec_const_reg</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>mov_ind_reg_nosub</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>subc_imm_reg_sub</td>
<td>2</td>
<td>1,1</td>
<td>1,1</td>
<td>1,1</td>
</tr>
<tr>
<td>bit_mem_mem_nosub</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>cmp_mem_mem_sub</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>xor_reg_const_ind_nosub</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>inc_const_reg</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Sequence 1
### Classification Rates Based on Robustly Trained Classifiers

<table>
<thead>
<tr>
<th>CCPI</th>
<th>Instructions</th>
<th>CCPI Prediction Rates (%)</th>
<th>Power Sum</th>
<th>Freq. %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Pin 1</td>
<td>Pin 2</td>
<td>Pin 3</td>
</tr>
<tr>
<td>2</td>
<td>pop_mem_reg</td>
<td>100.00</td>
<td>100.00</td>
<td>90.00</td>
</tr>
<tr>
<td>5</td>
<td>add_mem_mem_nosub</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>4</td>
<td>inc_reg_const_ind_nosub</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>5</td>
<td>mov_mem_ind_nosub</td>
<td>100.00</td>
<td>100.00</td>
<td>87.50</td>
</tr>
<tr>
<td>1</td>
<td>add_reg_reg</td>
<td>100.00</td>
<td>100.00</td>
<td>80.00</td>
</tr>
<tr>
<td>6</td>
<td>sub_mem_mem_sub</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>1</td>
<td>dec_const_reg</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>3</td>
<td>mov_ind_reg_nosub</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>2</td>
<td>subc_imm_reg_sub</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>6</td>
<td>bit_mem_mem_nosub</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>5</td>
<td>cmp_mem_mem_sub</td>
<td>100.00</td>
<td>100.00</td>
<td>81.81</td>
</tr>
<tr>
<td>4</td>
<td>xor_reg_const_ind_nosub</td>
<td>100.00</td>
<td>100.00</td>
<td>36.36</td>
</tr>
<tr>
<td>1</td>
<td>inc_const_reg</td>
<td>100.00</td>
<td>100.00</td>
<td>12.50</td>
</tr>
<tr>
<td>1</td>
<td>nop_reg_reg</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td></td>
<td><strong>Overall Prediction Rate (%)</strong></td>
<td><strong>100.00</strong></td>
<td><strong>100.00</strong></td>
<td><strong>84.96</strong></td>
</tr>
</tbody>
</table>
A SHORT DEMONSTRATION OF SOFTWARE AUTOMATING REVERSE ENGINEERING FIRMWARE RUNNING ON A MICRO-CONTROLLER
LAST STEP - INSTRUCTION CLASSIFICATION

- With boundaries determined, we can proceed with coarse-gain instruction classification or exact instruction classification.
  - Both reduce entropy and allow "code meta data" to be extracted from code e.g. use of multipliers, addressing modes, etc.
INTERESTING APPLICATIONS

• PROTECTION – Secure Monitors of Side-Channels to Verify Firmware Code and compare to observed behaviors at high-levels for a cross-layer attack monitor for run-time code attestation
Deception attacks involve a change in the normal sequence of operations within a micro-controller.

### Instruction Sequences

<table>
<thead>
<tr>
<th>Nominal Code</th>
<th>6 cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov 30(r5), 0(r9)</td>
<td></td>
</tr>
<tr>
<td>mov @r10, 30(r6)</td>
<td>5 cycles</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Replay Code</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>mov 30(r4), 0(r9)</td>
<td>6 cycles</td>
</tr>
<tr>
<td>mov @r10, 30(r6)</td>
<td>5 cycles</td>
</tr>
<tr>
<td>inc r4</td>
<td>1 cycle</td>
</tr>
<tr>
<td>inc r4</td>
<td>1 cycle</td>
</tr>
</tbody>
</table>
“REVERSE ENGINEERING”-BASED ATTACKS

- In the following example, by detecting if code A or B is running, we can infer a secret condition and perhaps secret data

```
IF (SECRET CONDITION)
  DO CODE A
ELSE
  DO CODE B
```

- We can infer “code meta-data” e.g. use of multipliers, addressing modes etc., which could be used to infer properties of private code such as the type of encryption being used.

- In larger sequences, we can infer where and when sensitive code might be executed, memory access etc., allowing more targeted attacks.
PROTECTION: EXPLOITING ADDITIONAL GENERALIZED SIDE-CHANNELS IN CPS

- In general, a cyber physical system has many physical “generalized” side-channels across many layers that could be security monitored to verify system behavior.
  - Goal is to link behaviors across layers/levels using machine learning to make hidden attacks difficult and continued degraded function possible
- Ex: Motors speed verified by vibration sensors
  - Human pressing a pedal verified by body motion sensors
CONCLUSIONS

• Presented side-channel analysis allowing redundant measurements and inference across disparate levels based on physical side-channel evidence:
  • Ex: Circuit power-supply <-> software
• Use cases involve protection and attacks
• Cross-Layer validation in CPS can protect systems that are otherwise unprotected by layer-wise security measures
REFERENCE PUBLICATIONS


• Tracie Severson, Erick J. Rodríguez-Seda, Brien Croteau*, Deepak Krishnankutty, Kiriakos Kiriakidis, Chintan Patel, Nilanjan Banerjee, Ryan Robucci “Trust-Based Framework for Resilience to Sensor-Targeted Attacks in Cyber-Physical Systems” 2018 IEEE American Control Conference (accepted for publication)

