Flexible Intrusion Detection Systems for Memory-Constrained Embedded Systems

Farid Molazem, Karthik Pattabiraman
University of British Columbia
Embedded Systems
Attacks spreading

Hackers Remotely Kill a Jeep on the Highway—With Me in It

The FBI said in a cyber intelligence bulletin obtained by KrebsOnSecurity that attacks perpetuated against so-called “smart meter” installations over the past several years may have cost a single U.S. electric utility hundreds of millions of dollars annually. The law enforcement agency said this is the first known report of criminals compromising the hi-tech meters, and that it expects this type of fraud to spread across the country as more utilities deploy smart grid technology.

Smart meters are intended to improve efficiency, reliability, and allow the electric utility to charge different rates for...
Attacks spreading

Need an Intrusion Detection System
Existing solutions

- **Statistical Techniques**
  - Neural networks [Moradi et. al.]
  - Hidden Markov Models [Warrender et. al.]
  - Support Vector Machines [Wenjie et. al.]

- **Static analysis**
  - Call-graph, NDPDA [Wagner et. al.]
  - Dyck [Giffin et. al.]
Challenge

- False positives
Challenge

- Memory

```c
{    
a = receive();  
if (a > 0)    
    foo(a);  
else    
    bar(a); 
}

void foo(int a) {    
if (a % 2 == 0)    
    even(a);  
else    
    odd(a); 
}

void bar(int a) {    
if (a == -1)    
    error1();  
else if (a == -2)    
    error2(); 
}
```

```
a > 0  
  
a % 2 == 0  
  a % 2 == 1  
  a == -1  
  a == -2  
  a <= 0
```
Idea

- Can’t fit everything in memory
- Quantify security
- Optimize security for the memory we have
Overview

Coverage function

Our work

Software Design Documents (SDD)

Code

Invariants

Monitoring Software trace

IDS
What we do

1- Study Software Design Document
2- Generating abstract Invariants
3- Static Analysis
4- Generating concrete invariants
5- Select optimized invariants
Steps 1-2

- Storage/Retrieval integrity

Sensor data must eventually be stored on flash memory

$\Box (\text{getting sensorData} \Rightarrow (\Diamond \text{store on flash}))$

Temporal Logic

$\Diamond$: Something eventually happens

$\Box$: Something always happens

$\circ$: Something happens the next time
Steps 3-4

1. Study Software Design Document
2. Generating abstract Invariants
3. Static Analysis
4. Generating concrete invariants
5. Select optimized invariants

Abstract invariants

Concrete invariants (contain system calls)
Steps 3-4

□ (getting sensorData(data) ⇒ (store on flash(data)))

□ (receive(d) ⇒ (write(d)))
Step 5

1- Study Software Design Document
2- Generating abstract Invariants
3- Static Analysis
4- Generating concrete invariants
5- Select optimized invariants

Software Design Documents (SDD)

Code

Coverage function
Coverage, Example

Define a coverage function on the graph and maximize it.
Define a coverage function on the graph and maximize it.
Coverage, Example

MaxMin Coverage IDS:
Intuition: Make the weakest coverage as strong as possible
Coverage, Example

MaxMin Coverage IDS:
Intuition: Make the weakest coverage as strong as possible

1- Study Software Design Document
2- Generating abstract Invariants
3- Static Analysis
4- Generating concrete invariants
5- Select optimized invariants

Intuition: Make the weakest coverage as strong as possible
Building the IDS

Select the invariants from the graph

Automatically convert it to Buchi Automaton
Iterative process, optimize for memory.
Building the IDS

\[(a \rightarrow F b) && (a \rightarrow F s) && (b \rightarrow F c) && (e \rightarrow F c) && (t \rightarrow F (a && X b)) && (v \rightarrow F (e && X d)) && (s \rightarrow F d) && (s \rightarrow F e) && (s \rightarrow F w)\]
Building the IDS

\[(a \rightarrow F b) \&\& (a \rightarrow F s) \&\& (b \rightarrow F c) \&\& (e \rightarrow F c) \&\& (t \rightarrow F (a \&\& X b))\]
Building the IDS

\[(a \rightarrow F b) \&\& (a \rightarrow F s)\]
Evaluation

- SegMeter: Smart meter, an important device used in smart homes
- Meter:
  - Arduino board
    - ATMEGA 32x series microcontroller
    - Sensors
  - Gateway board
    - Broadcom BCM 3302 240MHz CPU
    - 16 MB RAM
    - 4 MB available for IDS
    - OpenWRT Linux
- IDS runs on the Gateway board
- No attack database available => We use fault injection to simulate attacks
Fault injection

• Flipping branches

if (data_file ~= nil) then
    big_string = data_file:read("*all")
...
end

if (data_file == nil) then
    big_string = data_file:read("*all")
...
end
Research questions

• How close is the estimated coverage at design time to the coverage at run-time?
  • Shows whether the theoretical optimization is useful

• What is the performance overhead
  • Shows whether it is practical to implement and use
Results (MaxMin IDS)

- How good is the coverage of the IDS?
- How good the graph-based optimization is reflected at run-time?

\[
\text{Detection (\%)} = \frac{\text{number of detected attacks}}{\text{total number of injected attacks}} \times 100
\]
Performance overhead
Discussions

- Quantifiable coverage provides flexibility

- How to pick a coverage function?

- Having a complete software engineering life cycle can help producing automated security solutions
Conclusions

- Traditional solutions don’t work
- **We can quantify security**
- We can use different security measurement functions

1- Study Software Design Document
2- Generating abstract invariants
3- Static Analysis
4- Generating concrete invariants
5- Select optimized invariants

Coverage function

Code

faridm@ece.ubc.ca
MaxProperty IDS:
Maximize security properties that are fully covered
Results (MaxProperty IDS)

- How good is the coverage of the IDS?
- How good the graph-based optimization is reflected at run-time?