11TH INTERNATIONAL CONFERENCE
ON CRITICAL INFORMATION
INFRASTRUCTURES
SECURITY

10-12 October 2016
UIC HQ Paris

CRITIS
2016

Domain Specific Stateful Filtering with Worst-Case Bandwidth

Maxime Puys, Marie-Laure Potet, and Jean-Louis Roch

Verimag, University Grenoble Alpes/Grenoble-INP, Gières, France
Firstname.Name@imag.fr

Industrial Systems (SCADA)

- Hot topic:
  - Increasing number of attacks showed in the media since Stuxnet.
    - Complex attack ending up in increasing speed of Iranian centrifuges to damage them.
    - Also attacked the process monitoring to trick operators.
  - Protection becoming a priority for government agencies.

Specificities of Industrial Systems

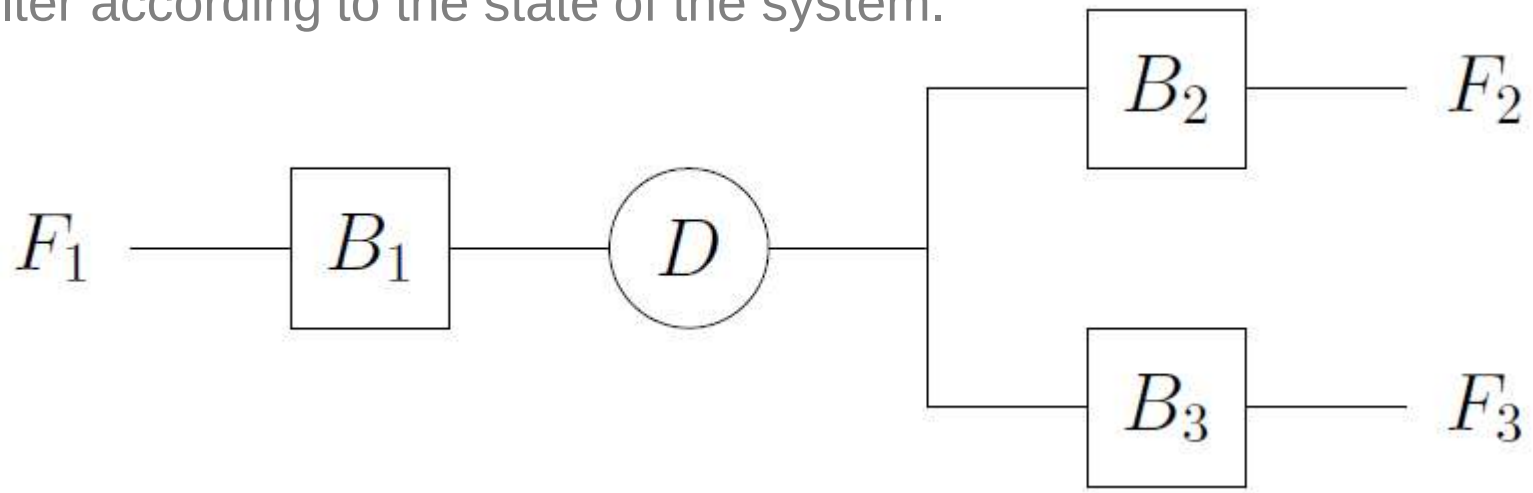
- Security objectives are different from traditional systems:
  - **Availability, integrity**, authentication and non-repudiation.

- Messages are READ/WRITE commands to PLCs.
  - Sometimes SUBSCRIPTIONS, RPCs or grouped commands.
  - Industrial protocols: MODBUS, OPC-UA.

- Attack examples: change the value of a WRITE request to change a temperature, change a READ response to mislead operators.

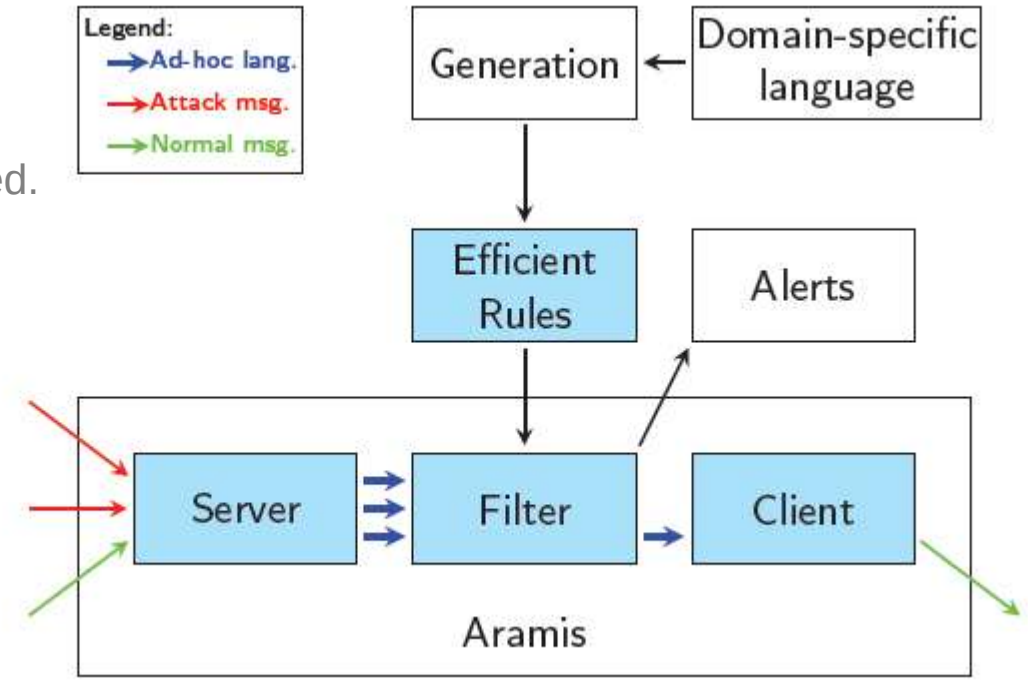
Motivating Example

- An electrical disconnector D:
  - Between three feeders F_1, F_2, F_3.
  - Each separated by a breaker.
- No manipulation of D while current is passing through.
  - Either B_1 is open OR (B_2 AND B_3 are open).
  - Need to filter according to the state of the system.



PIA ARAMIS Project

- Architecture Robuste pour les Automates et Matériels des Infrastructures Sensibles.
- Partners: ATOS World Grid, SecLab, CEA-leti, University of Grenoble Alpes.
- Objectives:
  - Build a transparent device to filter industrial flows.
  - Communication protocols are stopped and re-created.
  - Applicative filtering with domain specific constraints



Contributions

- Protocol-independent low-level language.
  - SCADA specific protocols (e.g.: MODBUS, OPC-UA).
- To describe a stateful type of domain specific filtering.
  - Automatically keeps track of the value of predetermined variables.
- Worst-case bandwidth and memory to meet real-time constraints.

Language 1/2

- Servers and Variables:
  - Variables configured on the real server.

```
# A MODBUS server.
Declare Server 1 Protocol Modbus Addr 10.0.0.1 Port 502

# A MODBUS coil (read / write Boolean).
Declare Variable 1 Server 1 Type Boolean Addr coils:0x10

# An OPC-UA server.
Declare Server 2 Protocol OpcUa Addr 10.0.0.2 Port 48010

# An OPC-UA unsigned integer 5×10 matrix.
Declare Variable 2 Server 2 Type UInt32 Addr numeric:5000 Dims 5 10
```

Language 2/2

- Domain specific safety rules to be verified:
  - Assert (= Block) or Warning (= Only logs).
  - Boolean predicates separated by AND and OR operators.

```
# Variable 1 should never been set to its current value
# (e.g.: opening a currently opened circuit breaker).
Declare Rule Variable 1 Assert NotEqual(NewVal, LocalVal[1])
```

Language 2/2

- Local state variables:
  - Local copy of server variables within the filter.
  - Updated on READ responses and WRITE responses (with requested values).
  - Only target scalar variables (use of Index keyword).

```
# A local variable on a MODBUS coil.
Declare LocalVal 1 Variable 1
```

  - Limited to local vision of the filter (at least the same as the client since we intercept communications).

Implementation

- Non-shared variables between clients:
  - Prevent isolation problem.

- When loaded into the filter, rules are indexed by (client, server) and then by variable:
  - Fast access to rules to verify.

- Embedded on the ARAMIS device.

Worst-Case Bandwidth

Both conditions and actions have to be processed in constant time:

- Conditions are O(1) boolean predicates.
- Actions are : (i) Block or transmit the message, (ii) Log information, (iii) Update a local variable,

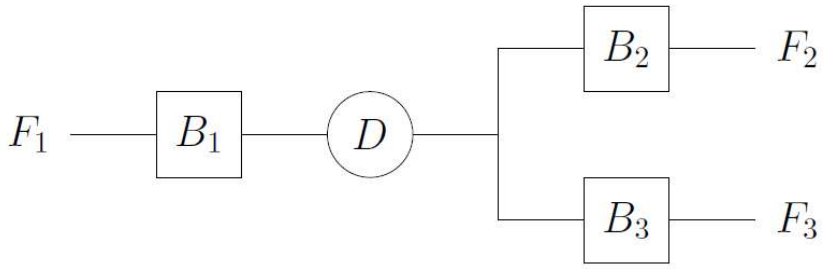
Thus processing one command only depends on the number of rules:

- For all predicates P , worst case processing time T of a message is $T = \sum \tau_i n_i$
- With τ_i the processing time of predicate P_i
- And n_i number of occurrences of predicate P_i

In practice, as only relevant rules are tested for a message, performances are better than upper-bound.

Motivating Example

D can be manipulated if and only if either B_1 is opened or if both B_2 and B_3 are open.




```
Declare Server 1 Protocol Modbus Addr 10.0.0.1 Port 502

Declare Variable 1 Server 1 Type Boolean Addr coils:0x11 # B1
Declare Variable 2 Server 1 Type Boolean Addr coils:0x12 # B2
Declare Variable 3 Server 1 Type Boolean Addr coils:0x13 # B3
Declare Variable 4 Server 1 Type Boolean Addr coils:0x14 # D

Declare LocalVal 1 Variable 1 # Local variable on B1
Declare LocalVal 2 Variable 2 # Local variable on B2
Declare LocalVal 3 Variable 3 # Local variable on B3

Declare Rule Variable 4 Assert        \ # Rule on variable 4 = D
            Equal(LocalVal [1], False ) OR \ # Using local variable on B1, B2,
B3
            Equal(LocalVal [2], False ) AND Equal(LocalVal[3], False )
```

Related Work

- Classical safety monitoring properties, e.g.: Roşu, 2012
  - Our language goes into an embedded device with high constraints.
  - Restrictions for filtering SCADA.
- Lot of IDS/IPS (e.g.: Snort or Bro):
  - Such IDS/IPS are unable to keep track of variable values.
- Chen et al., 2014:
  - Estimation through machine learning, MODBUS only
- Stergiopoulos et al., 2015:
  - Code review in industrial softwares to predict failures.

Conclusion

- Stateful filtering language:
  - Restriction of classic monitoring properties.
  - Keeps track of the value of predetermined variables.
- Bounded memory space and execution time:
  - Processing one command only depends on the number of rules.
  - Upper-bound allows to compare the worst-case path of the filter's configuation file with field constraints.
- Future work:
  - Handle more complex arithmetics : "Equal(2*NewVal+1, LocalVal[1]**2)".
  - Rules to avoid domain specific denial of service.
  - More precise upper-bound.

References

[PPL16] Maxime Puys, Marie-Laure Potet and Pascal Lafourcade. Formal Analysis of Security Properties on the OPC-UA SCADA Protocol. In SAFECOMP'16, 2016.

[Snort] Snort Team. Snort: Open source network intrusion prevention system. https://www.snort.org, April 2016.

[Pax99] Bro: A System for Detecting Network Intruders in Real-Time, Vern Paxson, 1999.

[Roş12] On safety properties and their monitoring. Scientific Annals of Computer Science, 22(2):327–365, December 2012.

[CA14] Qian Chen and Sherif Abdelwahed. A model-based approach to self-protection in scada systems. In IWFC'14, Philadelphia, PA, June 2014.

[STG15] George Stergiopoulos, Marianthi Theocharidou, and Dimitris Gritzalis. Using logical error detection in software controlling remote-terminal units to predict critical information infrastructures failures. In ICHAISPT'15, 2015.

Conclusion

Thanks for your attention!

Maxime.Puys@imag.fr