A Security Specification Language (SSL) for Run-Time Policy Enforcement

Topic Area: Design approaches and Run Time Assurance for Highly Dynamic Systems

Sandeep Shukla
FERMAT Lab,
Centre for Embedded Systems for Critical Applications
Bradley Department of Electrical and Computer Engineering
Virginia Tech, USA.

shukla@vt.edu

Caution: This research is in a preliminary state

Multi-Layer Security

Cyber Attacks can happen at any layer of a System

1. Firmware / HAL
2. System Libraries
3. O/S
4. Middleware
5. Application components

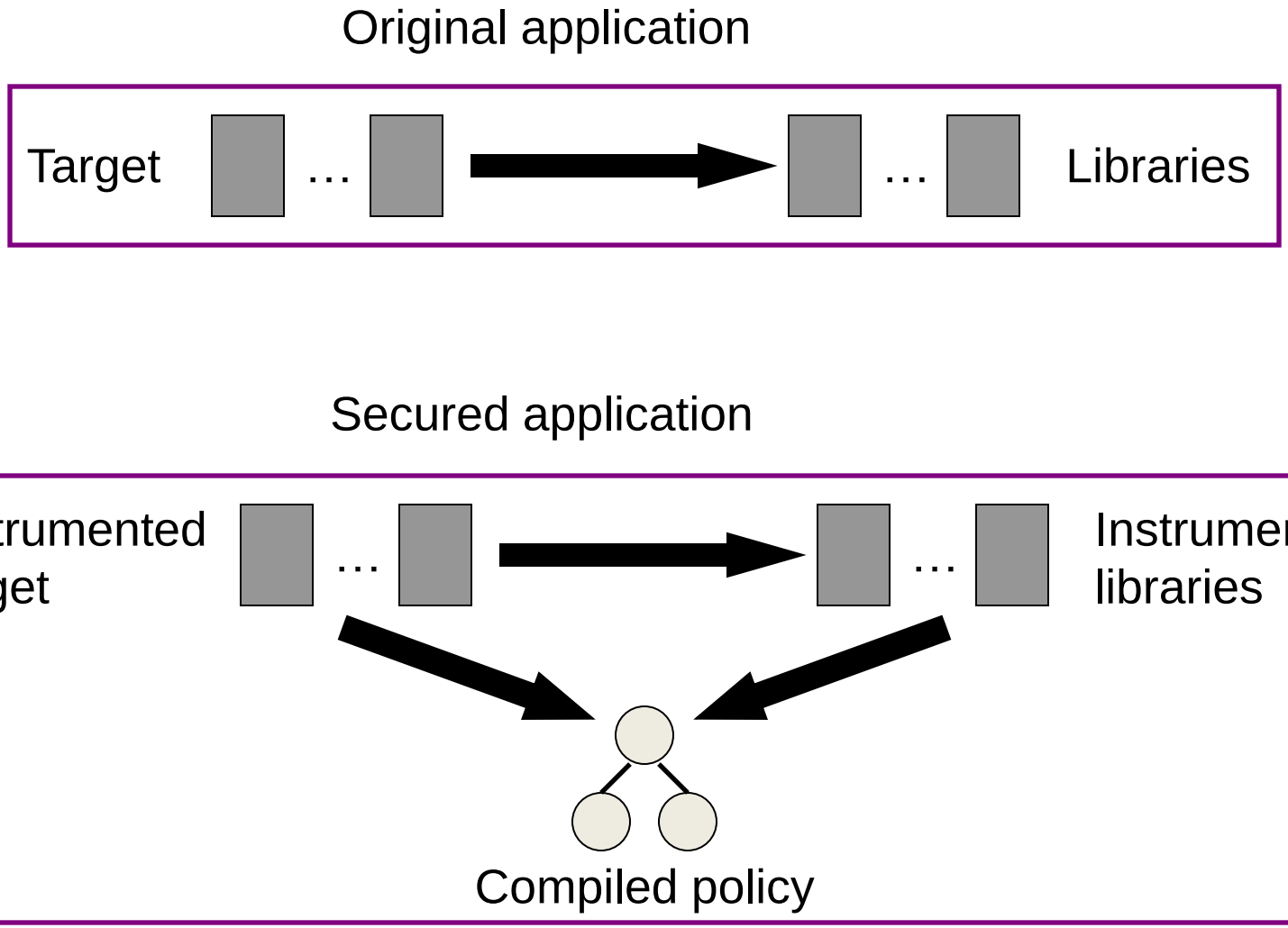
Strategies to Cope with Security Vulnerabilities

- Testing for known attacks (e.g., Penetration Testing)
- Formal static analysis of code for finding known vulnerabilities
- Run time monitoring for events symptomatic of known attacks
- Correlation of monitored events to determine higher order event indicative of an ongoing attack
- All of the above are needed

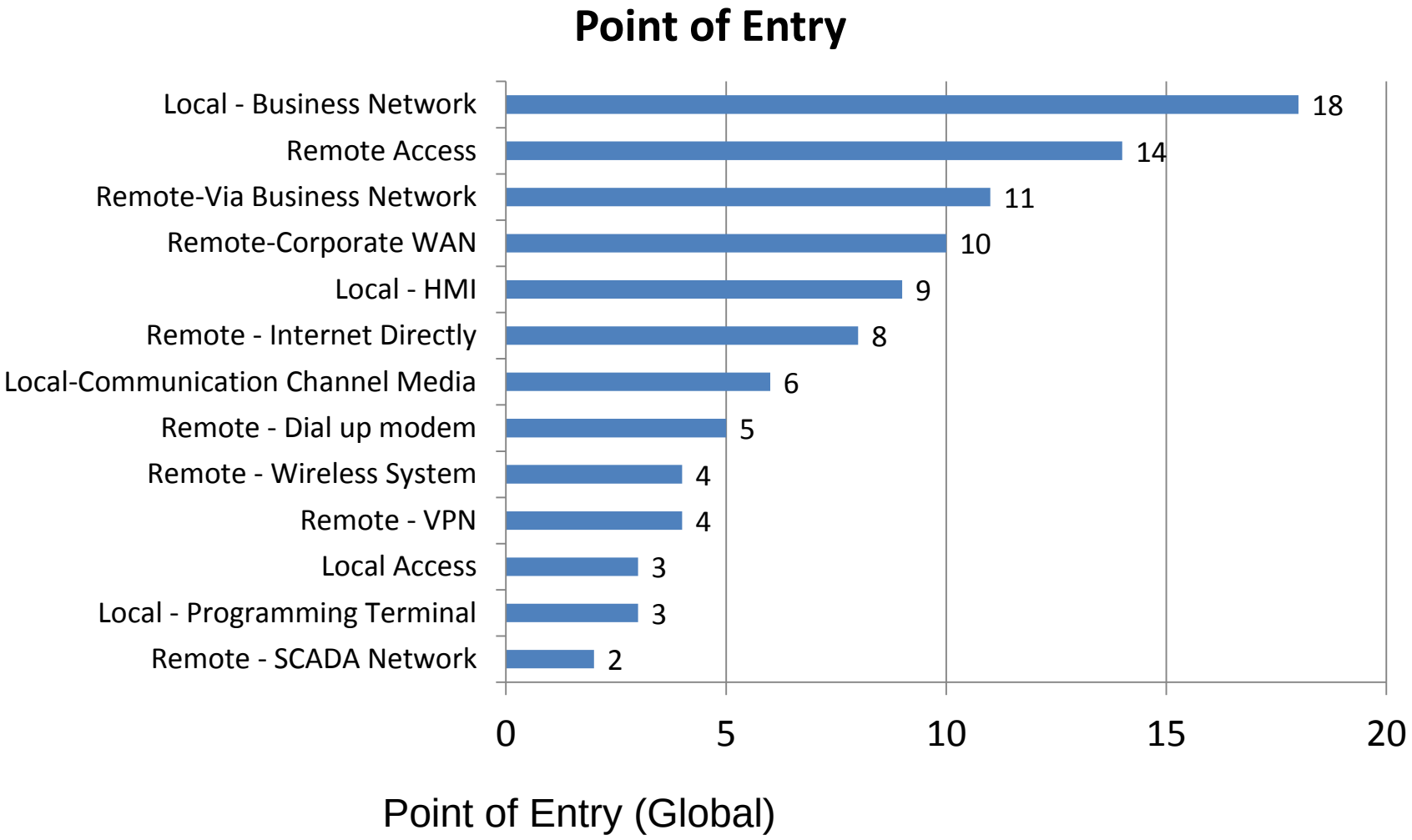
Monitoring Approach

- Identify events or event patterns that would violate security policy during run-time
- What do we need to know for effective monitoring?
  - Attack surfaces and threat models
  - Security policies to stop threat scenarios to happen
  - Monitors
    - Invasive
    - External

Run-time monitoring for Security Violations



SCADA Cyber Incidents – Points of Entry



Source: RISI Report

Detecting individual violations is not always enough

Situational Awareness

Need Event correlation.

Spatial Correlation

Temporal Correlation

Spatiotemporal Correlation

Event Hierarchy

Primitive Events

- Detectable Directly

First Order Events

- Events comprising of Primitive events

nth Order Events

– Events comprising of less than nth order events with at least one n-1th order event

Spatio-Temporal Correlation

- Temporal Correlation
  - A set of events which are partially ordered in time
  - Indicative of an incident
- Spatial Correlation
  - A set of events happening during the same time frame at various locations of the system
  - Indicative of an incident
- Spatio-Temporal Correlation
  - A set of events with both temporal and spatial diversity
  - Indicative of an incident

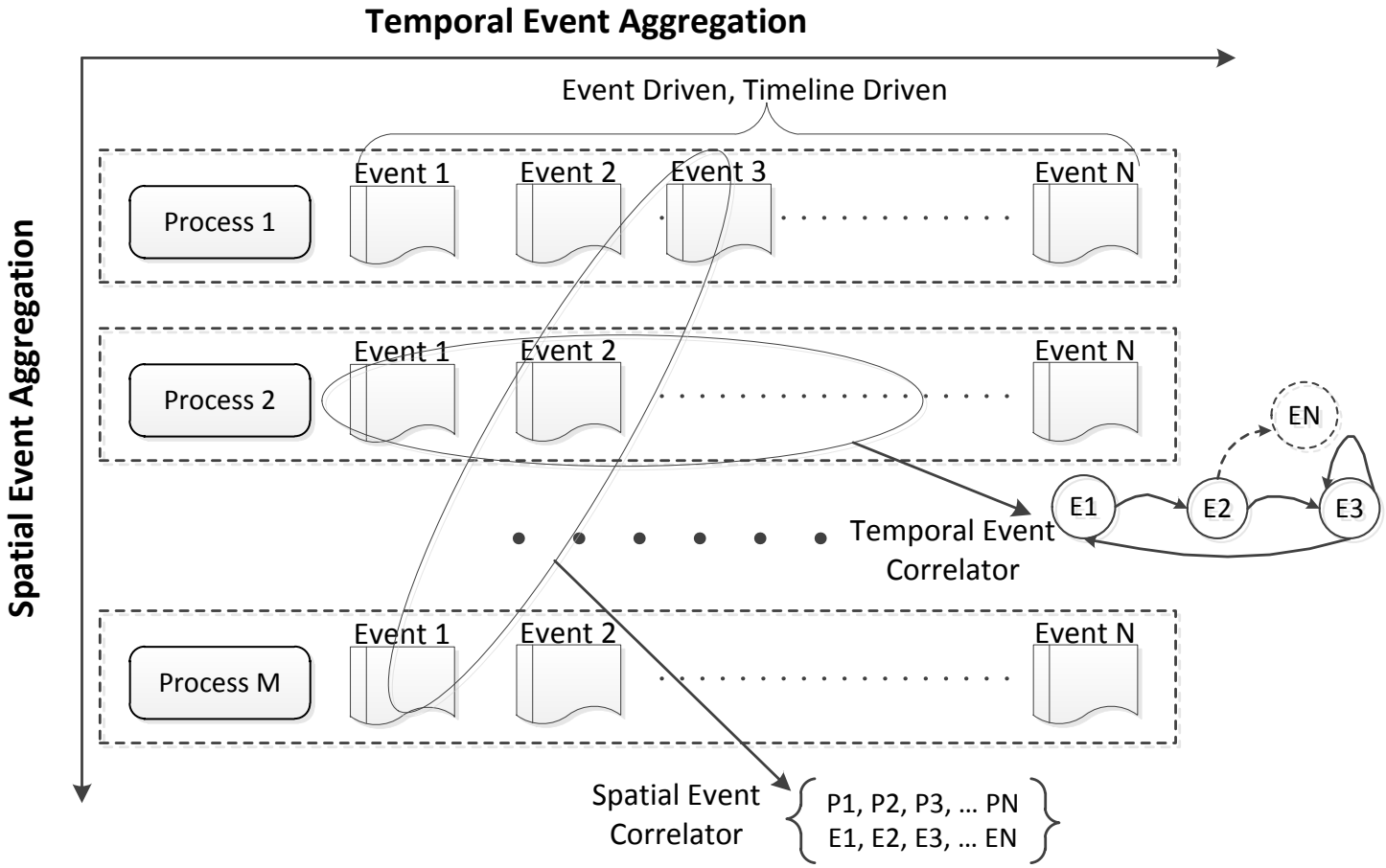
Examples

- Temporal
  - 3 unsuccessful login attempts within a pre-specified time duration –3Faillogin
  - Each unsuccessful login – primitive event
  - 3Faillogin – 1st order event
  - More than n 3Faillogin over a period of 12 hours – 2nd order event
- Spatial
  - 5 hosts indicate a change in memory usage
  - Each host's sudden increased memory usage – primitive event
  - 5 hosts reporting sudden increased memory usage – 1st order event

Examples (2)

- Spatio-temporal
  - All 3Faillogin's are from 2 distinct IP addresses
  - The 2 distinct IP addresses are within the same subnet
  - 2nd order spatio-temporal
    - Collection of 3Faillogins – each 1st order
    - Location information associated – spatial information

Spatio-temporal Correlation



Expression Event Correlation Relations

- How to express first order events in terms of primitive events
- How to express nth order events in terms of ith order events where $i \leq (n-1)$
- The intensional expression to extensional monitors
  - Automated Monitor Synthesis
- What is the best choice?
  - Regular Expressions?
  - Temporal Logic ?
  - Separation Logic ?
  - A combination?
- The choice must render itself to automated monitor synthesis

ECL – Event Correlation Logic

- We define a new modal logic based on temporal logic, Separation logic and regular expressions
  - Examples:
    - G([T] #(p) < n)
      - G – Always
      - p – primitive event of interest
      - #(p) – number of occurrences of p
      - T – a time interval
      - [T] – over that time interval
    - *If number of events p is n or more over a time period T, then we need to construct a first order event*

Pseudo-code for synthesised Monitor for G([L](#(p) < n))

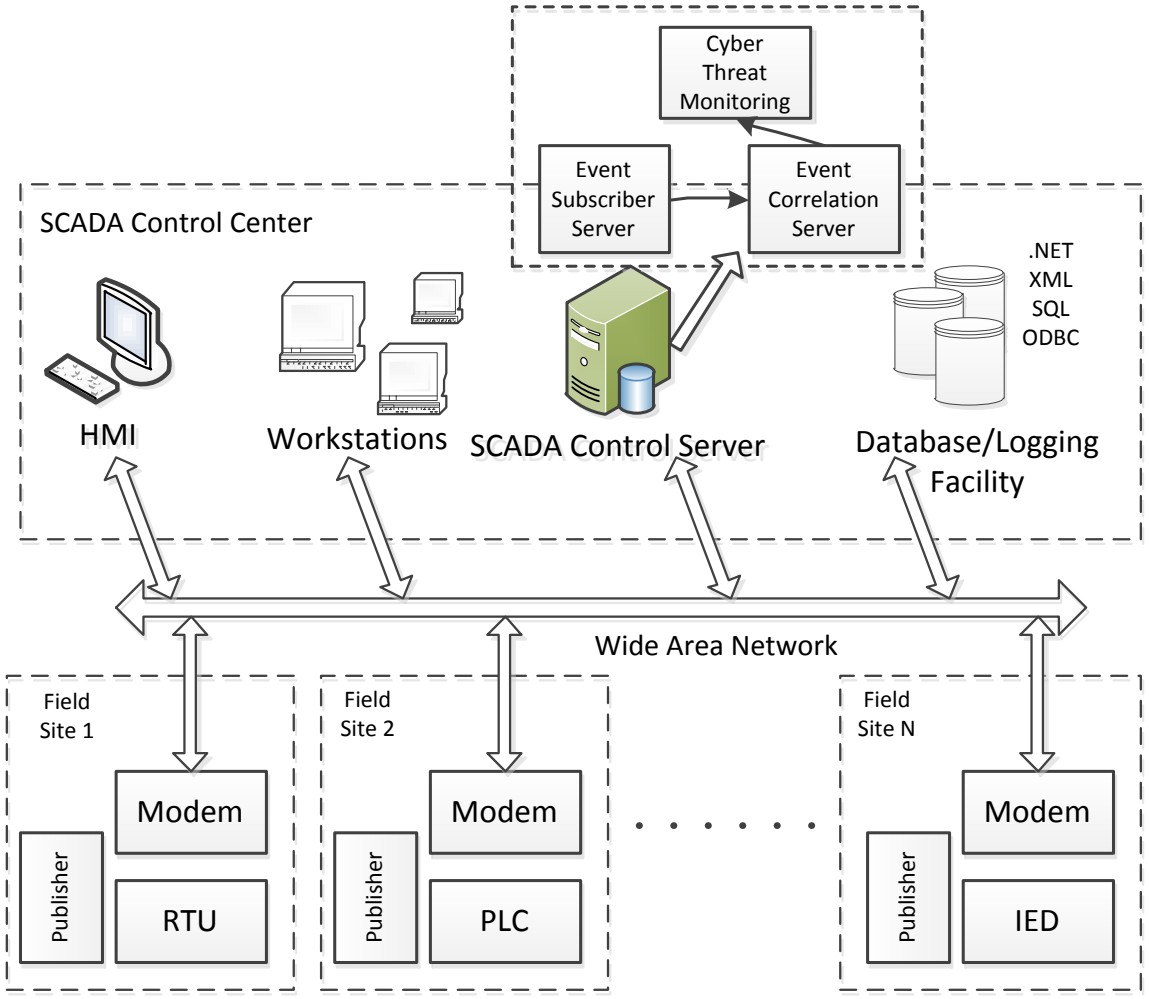
```
for every p
        Start { count = 0; monitor (p,L, count);}

monitor (p,L,count) {
        startTime = 0;
                for every p {
                        if (CurrentTime – startTime < L) {
                                count = count +1;
                                }
        If (count > n) flag event;
                else abort;
                        }
        }
```

A Spatial Correlation Example

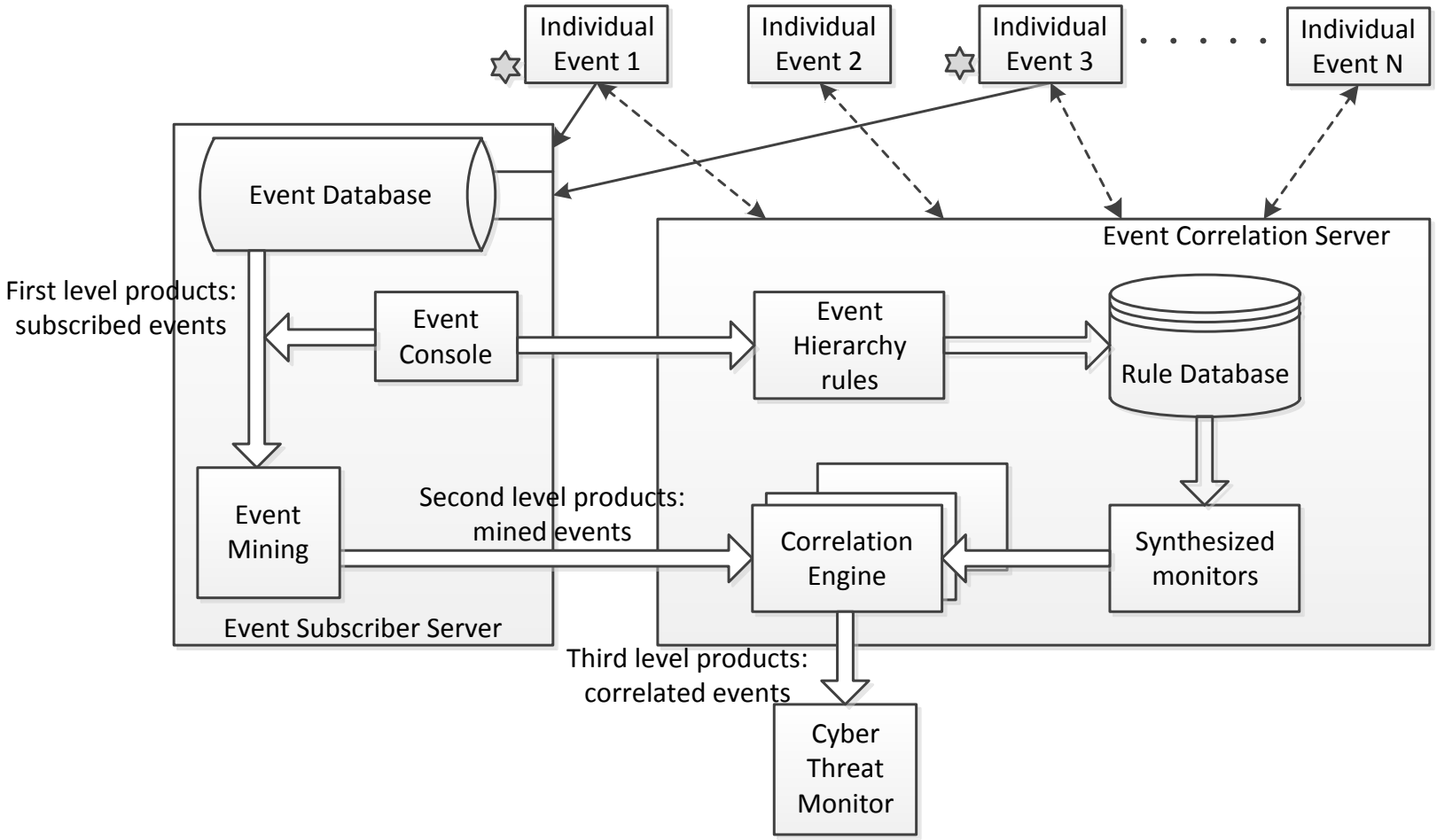
- G (login@l1 & login@(l2) & l1 ≠ l2) -> MultiLogin(l1,l2));
  - G – Always
  - login@l1 – a login event at IP address l1
  - login@l2 – a login event at IP address l2
  - l1 ≠ l2 → MultiLogin(l1,l2)

  - Whenever same account login happens from two distinct IP addresses raise a first order event called MultiLogin

Proposed Software Architecture for Correlation Monitors



Proposed Correlation System Architecture



Goals of this work

Assess Major Event Correlation Engines and Technologies

Define Event Hierarchy Model for Specific Domains

Define ECL and rules of inference in ECL

Design templates for customizing domain specific ECLS

Develop monitor Synthesis Algorithms and prototype implementation

Create a SCADA test bed on our GECO platform

GECO – PSLF and NS-2 Cosimulation platform

Inject security attacks, and check efficacy of monitors

Estimate Latency of communication for time critical event correlation

Integrate all the developed algorithms and techniques into a set of tools dedicated for intelligent analysis.

Meta Grammar for Policies

For the grammar for the Meta language is as follows:

Policy ::= Access_Right | Access_Right Policy | Access_Right + Policy | (Policy) | Policy*

Access_Right ::= (Actors : Access_Types : Resources)

Actors ::= Actor | {Actor, Actors}

Resources ::= Resource | {Resource, Resources}

Access_Types ::= Access_Type |{Access_Type, Access_Types}

Actual Policy as CFG

Actors ::= processor1 | processor2 | processA | process | device1
Access_Type ::= read |write | execute | erase | lock
Resource ::= memory_address_range(n1, n2) | device2 | shared_object1 | mutex1

Example 1

ar1 = (process1:{read+write}: memory_address_range(x,y)),
ar2 = (process2:read: memory_address_range(u,v))

(ar1 + ar2)*

Example 2

Bell and LaPadula confidentiality model

ar1 = (process1:{read+write}:range1);
ar2 = (process1:read:range2);
ar3 = (process2:{read+write}: range 2);
ar4 = (process2:write:range1);

policy = (ar1 + ar2 + ar3 + ar4)*.

Example 3

policy = switch (mode) {
case preboot: (X:{ r+w, EPROM)*;
case default: (X, r, EPROM) *;
}

Summary

- Run-time Monitoring is a common technique for security
- Monitor Synthesis is important for fast deployment
- Event correlation is important for system level threat detection
- Defining suitable formal language to capture security policies is an important area of research