

Decentralized publish-subscribe system to prevent coordinated attacks via alert correlation

J. Garcia, F. Autrel, J. Borrell,

S. Castillo, F. Cuppens, G. Navarro

`{jgarcia,jborrell,scastillo,gnavarro}@ccd.uab.es,`

`{fabien.autrel,frederic.cuppens}@enst-bretagne.fr`

Main Points

- ▶ Introduction
- ▶ Classical architectures
- ▶ Prevention framework
- ▶ Current Development
- ▶ Conclusions

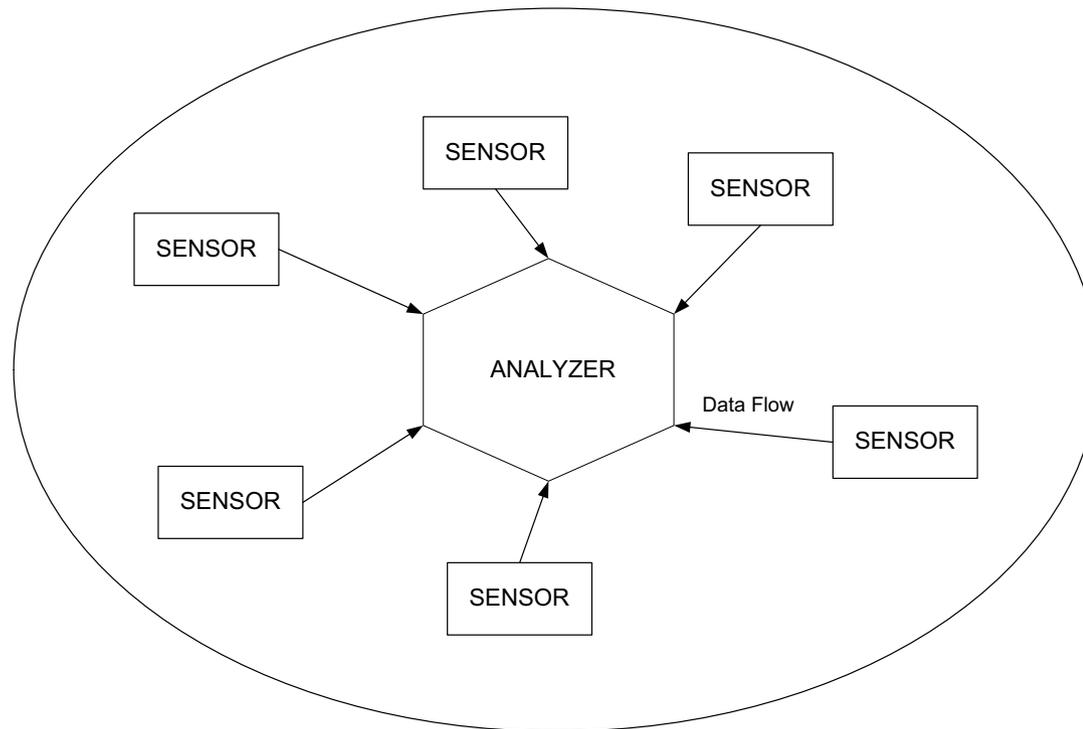
Coordinated Attacks

- ▶ “*Combination of actions performed by a malicious adversary to violate the security policy of a target computer system.*”
- ▶ Networks resources can become an active part of a coordinated attack
- ▶ E.g. An attack might start with an intrusion
 - ⇒ Nodes have to be monitored
- ▶ A global view of the whole system is needed for detection
 - ⇒ Collection and combination of events from different nodes

Components needed to prevent coordinated attacks

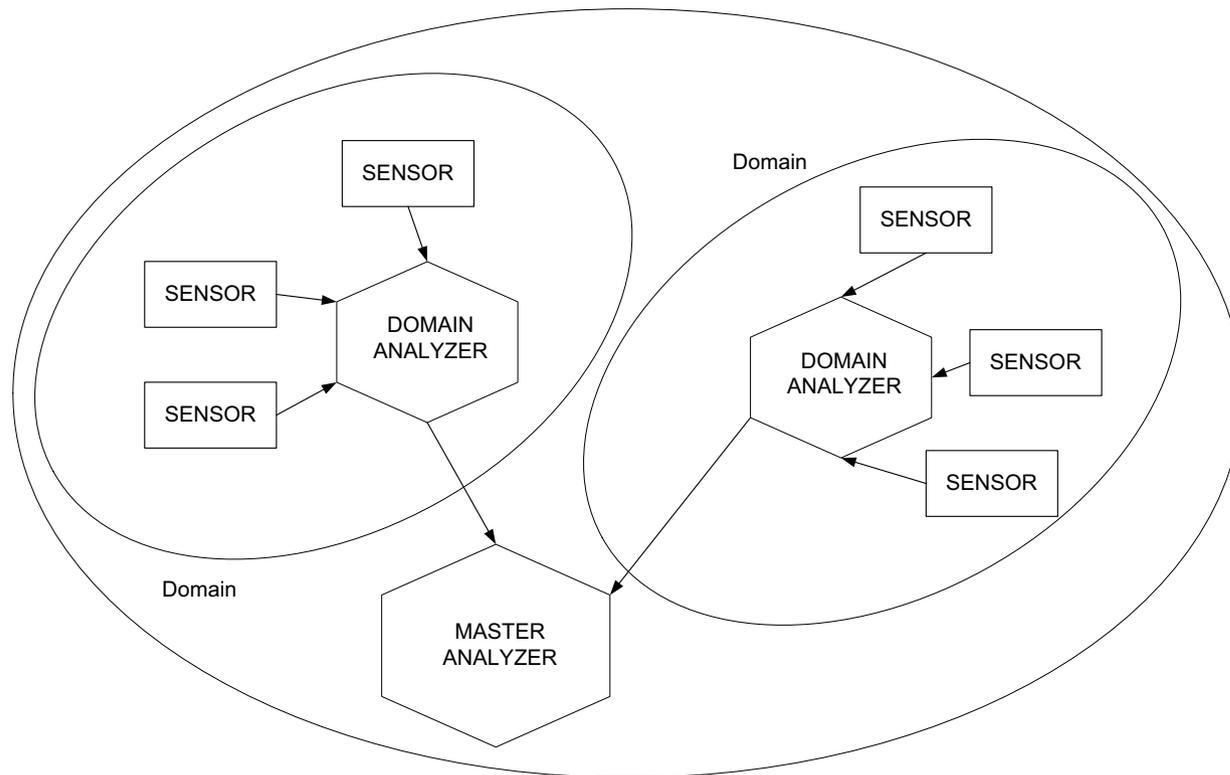
- ▶ Sensors (host, application or network based)
 - ▶ Analyzers (misuse or anomaly based)
 - ▶ Managers (data consolidation and alert correlation)
 - ▶ Response units (active or passive reaction)
-
- ▶ Intrusion Detection Systems use these same components to prevent a node getting compromised by an attacker
- ⇒ We use these components to prevent a compromised node becoming an active part of a coordinated attack.

Centralized event correlation



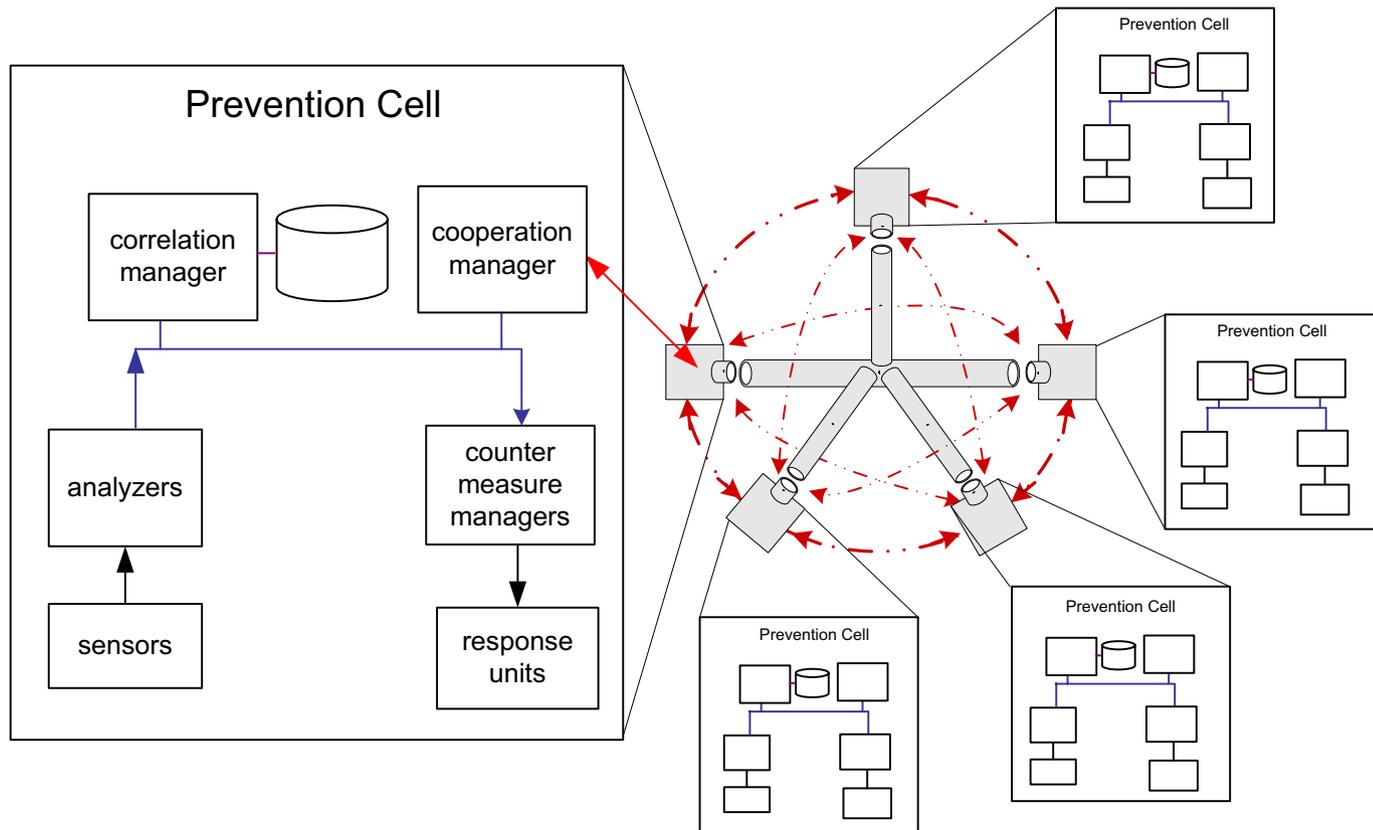
- ▶ DIDS - University of California, Davis (1991)
- ▶ STAT - University of California, Santa Barbara (1992)

Hierarchical event correlation



- ▶ EMERALD - SRI International, California (1997)
- ▶ AAFID - CERIAS, Purdue University (1998)

3. - Prevention Cells System

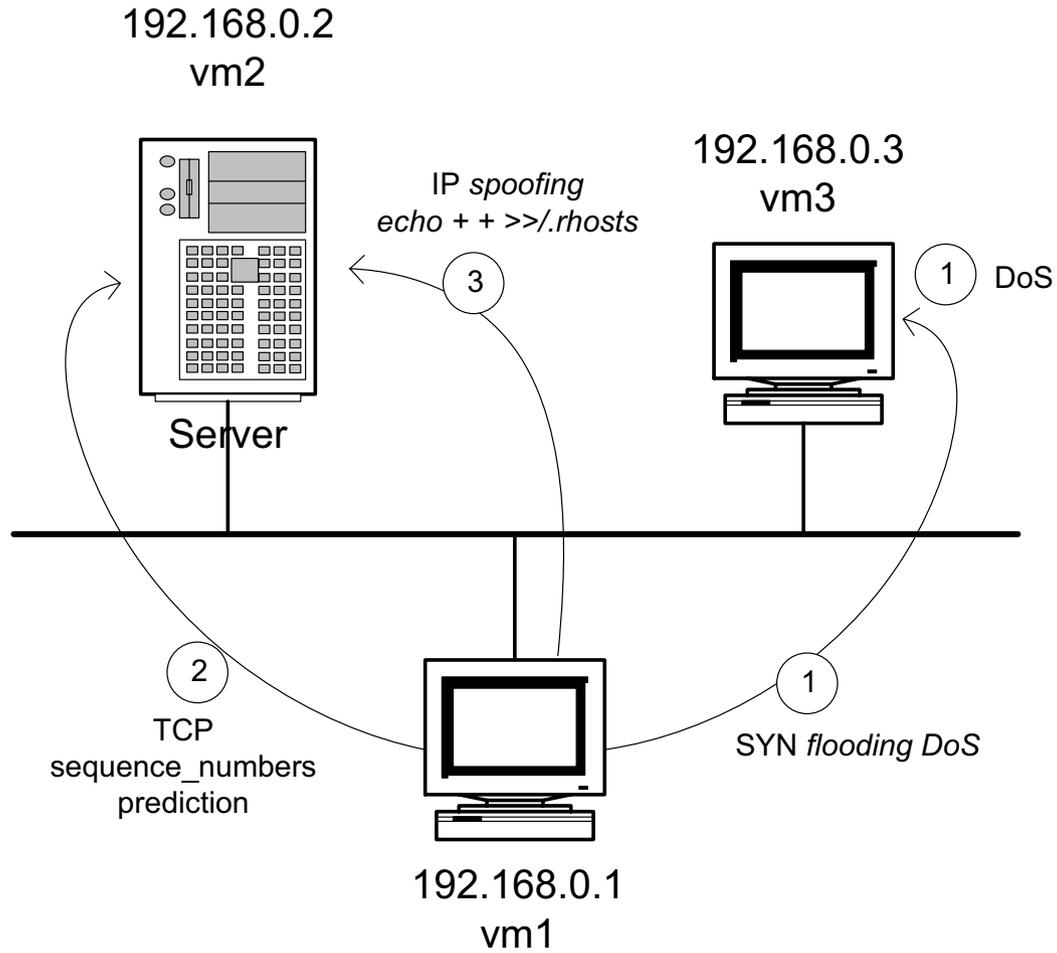


► Message passing architecture

⇒ The detection process can be completely distributed

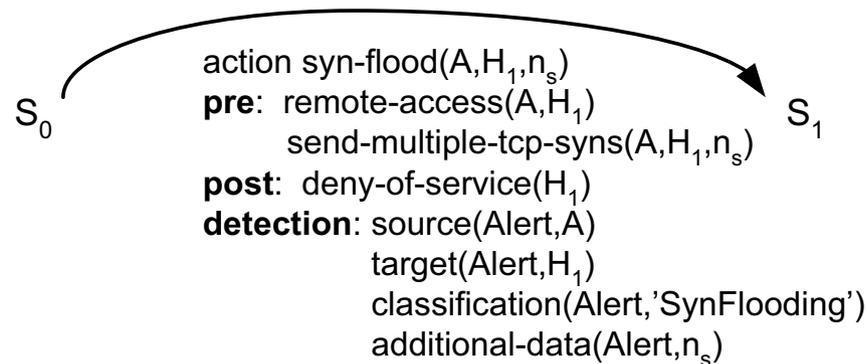
3. - Prevention framework

Sample scenario



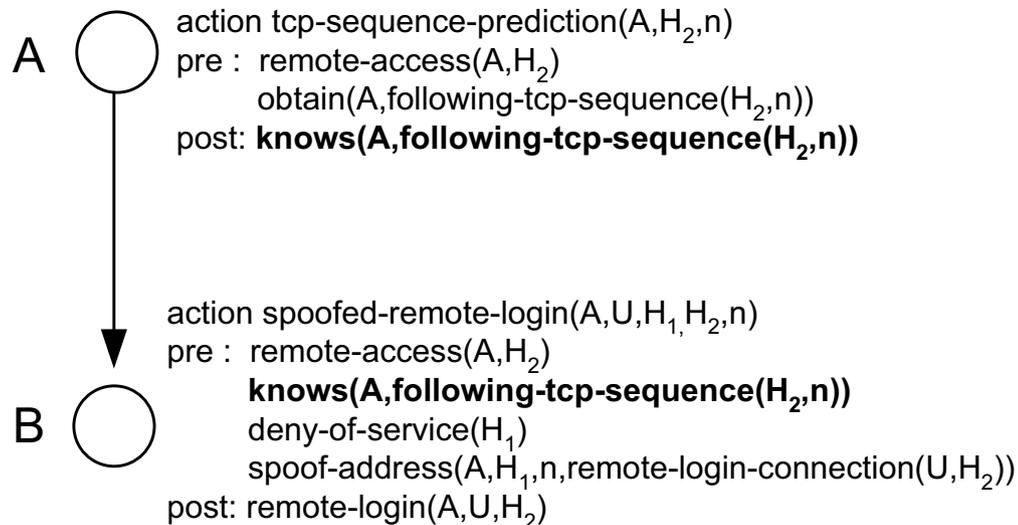
Detection Process

- ▶ Find the set of actions which transforms the system from an initial state S_0 to a final state S_n .



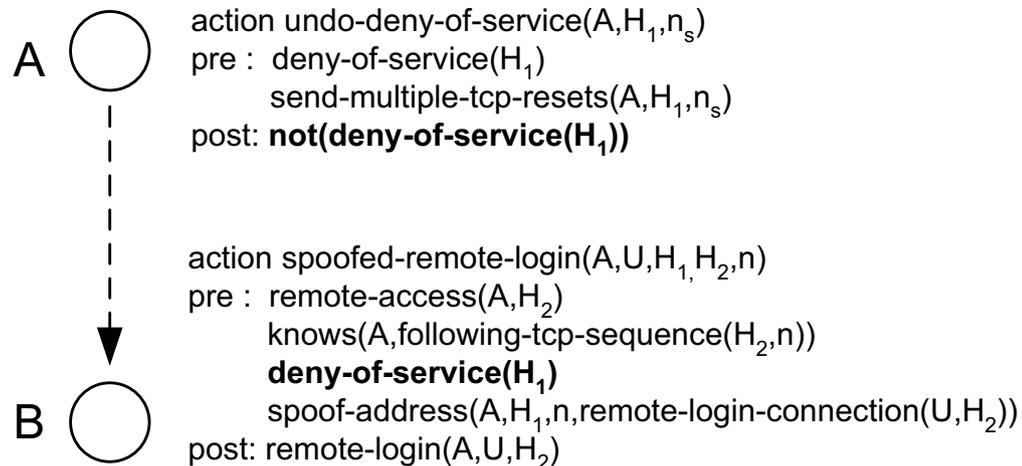
Detection process via alert correlation

- ▶ Two actions A and B can be correlated when the realization of A has a **positive influence** over the realization of B (given that A occurred before B):
 - ▷ $(E_a \in post(A) \wedge E_b \in pre(B)) \vee (not(E_a) \in post(A) \wedge not(E_b) \in pre(B))$
 - ▷ E_a and E_b are unifiable through a unifier θ



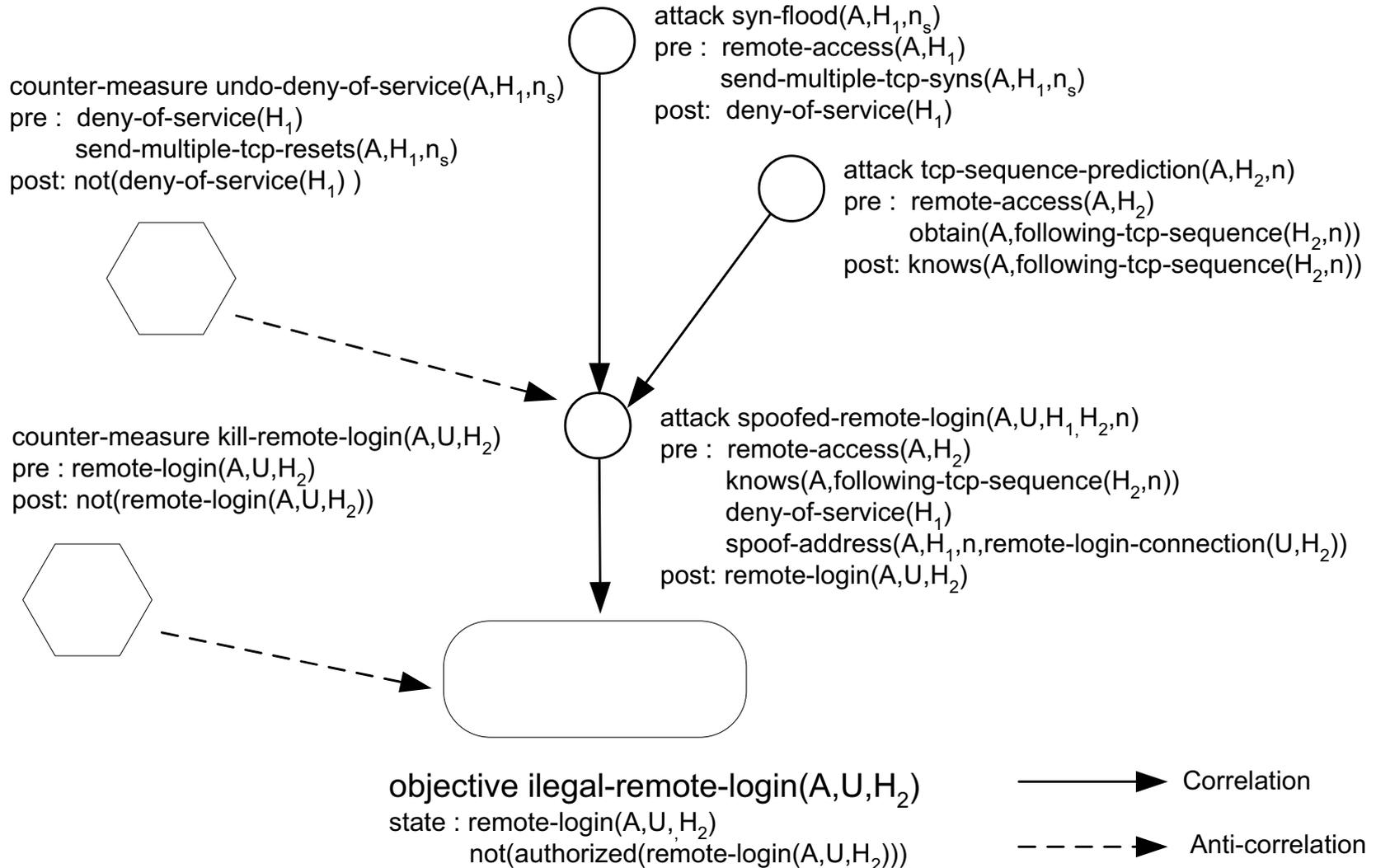
Reaction process via anti-correlation

- ▶ Two actions A and B are anti-correlated when the realization of A has a **negative influence** over the realization of B (given that A occurred before B):
 - ▷ $(\text{not}(E_a) \in \text{post}(A) \wedge E_b \in \text{pre}(B)) \vee (E_a \in \text{post}(A) \wedge \text{not}(E_b) \in \text{pre}(B))$
 - ▷ E_a and E_b are unifiable through a unifier θ

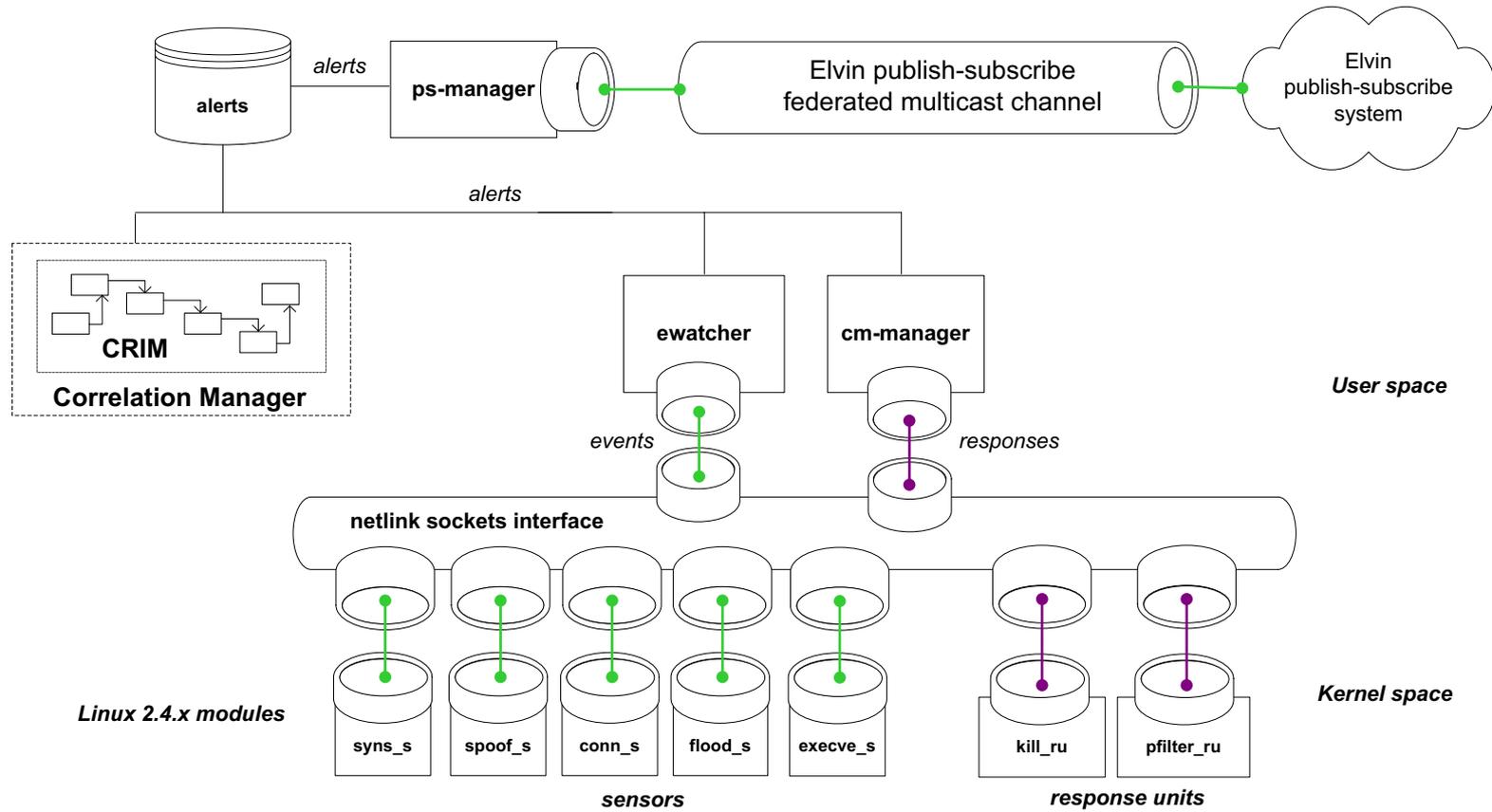


3. - Prevention framework

Detection and reaction graph for the sample scenario



Current Development



4. - Current Development

Name	Loaded Status
SYN/RST SYN/RST establ	Yes On
IP Spoofing	Yes On
SYN Flooding Adap.	Yes On
ICMP Flooding Adap.	No Off
CONNECT Logger	No Off
EXEC Logger	Yes On
KILL Process	Yes On
PACKET Filter	Yes On

```

Jun 15 04:38:22 vm2 kernel: SYN Stealth Scan Sensor Module; Unloaded
Jun 15 04:38:23 vm2 kernel: IPspoof Sensor Module; Unloaded
Jun 15 04:38:23 vm2 kernel: SYN Flooding Sensor Module; Unloaded
Jun 15 04:38:25 vm2 kernel: Execve Logger Module; Unloaded
Jun 15 04:38:26 vm2 elvind[2625]: Timeout waiting for response from ewaf;/tcp,none
Jun 15 04:38:26 vm2 elvind[2625]: Endpoint disconnected without warning.
Jun 15 04:38:26 vm2 last message repeated 5 times
Jun 15 04:38:26 vm2 kernel: Killer process Module; Unloaded successfully!
Jun 15 04:38:26 vm2 kernel: Packet filter Module; Unloaded successfully!
Jun 15 04:38:29 vm2 kernel: Packet filter Module; Loaded successfully!
Jun 15 04:38:30 vm2 kernel: Killer process Module; Loaded successfully!
Jun 15 04:38:31 vm2 kernel: SYN Stealth Scan Sensor Module; Loaded
    
```

classification	associated model name
JNQ-0001	syn-flood
JNQ-0003	IP_spoofing
unknown	unknown
JNQ-0004	spoofed-remote-shell
JNQ-0001	syn-flood
JNQ-0001	syn-flood
JNQ-0002	tcp-sequence-prediction

Scenario step information

Selected action: IP_spoofing (virtual)
 alert file: C:\joaquin\crim\JNQ\virtual_alerts\IP_spoofing_virtual_alert_0.xml
 pre condition correlated actions:
 syn-flood (C:/joaquin/crim/JNQ/processed_alerts/jnq_sflood_s-999882.xml)
 syn-flood (C:/joaquin/crim/JNQ/processed_alerts/jnq_sflood_s-999882.xml)
 syn-flood (C:/joaquin/crim/JNQ/processed_alerts/jnq_sflood_s-999882.xml)
 tcp-sequence-prediction (C:/joaquin/crim/JNQ/processed_alerts/jnq_tcppre_s-999884.xml)
 post condition correlated actions:

Selected scenario graph

```

graph LR
    A["tcp-sequence-prediction (0.00)"] --> B["IP_spoofing (0.50)"]
    B --> C["spoofed-remote-shell (1.00)"]
    C --> D["illegal-remote-shell"]
    B --> E["syn-flood (0.00)"]
    B --> F["syn-flood (0.00)"]
    B --> G["syn-flood (0.00)"]
    E --> H["block-spoofed-connection"]
    F --> H
    G --> H
    
```

Results of our work

- ▶ State of the art about coordinated attack prevention
- ▶ Study about alert correlation mechanisms
- ▶ Development of a generic framework avoiding bottleneck of centralized architectures using a distributed approach
- ▶ Both detection and reaction are performed by using the same formalism

Future work

- ▶ Incorporate fault tolerant mechanisms
- ▶ Make a more in-depth study of the format used for alerts
- ▶ Incorporate other information about the environment

Thank you! Questions?