Autonomous Protection Mechanism for Joint Networks in Coalition Operations

Martin Rehák, Michal Pěchouček, Magdalena Prokopová, Lukáš Foltýn and Jan Tožička

Gerstner Laboratory - Agent Technology Group
Department of Cybernetics, Czech Technical University

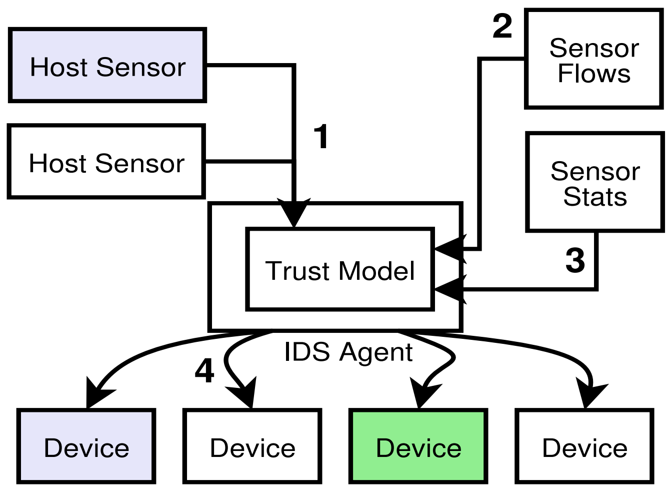
Supported by ARL project N62558-05-C-0028

- Joint network without well defined **perimeter**
- **Dynamic** network without joint oversight team
- Communication & **Interoperability** *vs* **Security**
- Devices belong to different coalition partners, need for cooperation **policies**
- **Constrained Environment**: Hard limitations on reasoning code (**i**) performance, (**ii**) robustness and (**iii**) size - requires low runtime complexity $\longrightarrow$ very good reflection use-case
- **Collaborative Agents in Adversarial Environment**: Limited competitiveness or self-interestedness – most agents/actions are either collaborative or adversarial

Protection Phases

- **Observation**: Observe the behavior of the network
  - Distributed
  - High-Performance
  - Low Overhead
  - Low Maintenance
  - Versatile
- **Detection**: Analyze the observations and discover the attacks
  - Effectiveness - low false positives/negatives
  - High-Performance – near-real time
- **Reaction**: Stage an efficient and effective response to detected attack
  - Effectiveness – low false positives/negatives)
  - Efficiency – limited performance impact
  - Robustness – decentralized, dynamic

IDS Architecture Components



- **Sensors**:
  - **Host Sensors**: Detect suspected attacks on hosts
  - **Network Sensors**: Connection/Flow Statistics [NetFlow like] and flow samples
- **IDS Agents**:
  - **Correlate** alarms from hosts with network flows with **generalized trust** modeling
  - **Generate** filters for attacks
  - **Start** filter deployment
- **Programmable Network Elements**:
  - Collectively **deploy** filters generated by IDS agents – **distributed task allocation**
  - **Delegate** filtering to other devices upon need

Goals and Assumptions

- **Goals**:
  - Reduce the **impact** of attacks based on malicious mobile code
  - Our solution does not prevent attacks, it counters their spread and effects
- **Assumptions**:
  - Host alerts **correlated** in time with attacks
  - **Heterogenous**, protected host population – use diversity for protection
  - **Random** attack spread strategy – all hosts in the system attacked with approximately identical probability
  - Availability of **adaptive** network elements
  - Doctrine change: (**i**) **Humans** are no longer directly in the loop and (**ii**) we counter mobile malicious code with autonomous collective reflection, i.e. **mobile protective** code

Observation

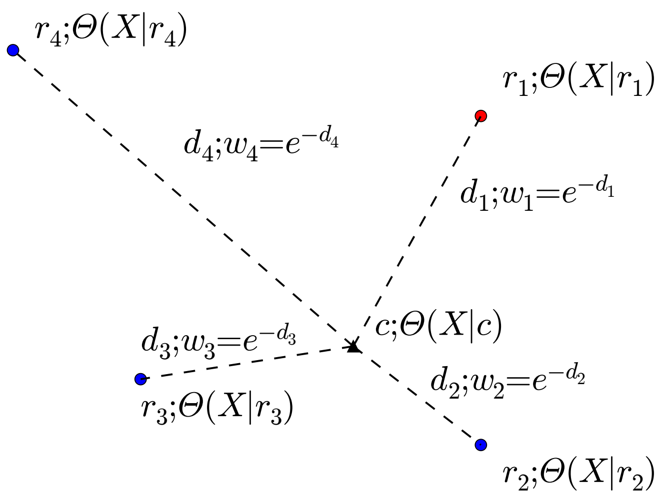
- **Host Sensors**: Detect attacks and suspicious activity on hosts – we only require the ability to provide binary alert information
  - personal firewalls [CA HIPS]
  - Host IDS systems [tripwire]
  - log analyzers
- **NetFlow**: and similar sensors provide statistics about connections on the network
  - provided by commercial network components [Cisco,others] and de-facto standard for research data as well
  - data aggregated by {**srcIP:srcPrt, dstIP,dstPrt, protocol**} over a time period
- **Flow Monitor**:
  - based on the concept of application identification [AT&T(Haffner)2005]
  - identification/separation of applications using the first 256 bytes of flow payload

Flow Modeling

- Characteristics of the flow, using the NetFlow-like identity format and context (adapted from MINDS [Ertoz2004])

Feature	**Description**
Connection Identity	
srcIP	Source IP Address.
destIP	Destination IP
srcPort	Source Port
destPort	Destination Port
Protocol	Protocol (TCP/UDP/ICMP)
Payload Signature	First 256 bytes of the flow content (application headers)
Connection Context	
count-dest	Number of flows to unique destinations from the same source.
count-src	Number of flows from the unique sources toward the same destination.
count-serv-src	Number of flows from the same IP to the same port.
count-serv-dest	Number of flows to the same destination IP using the same source port.

Flow Modeling: Identity and Context (1)

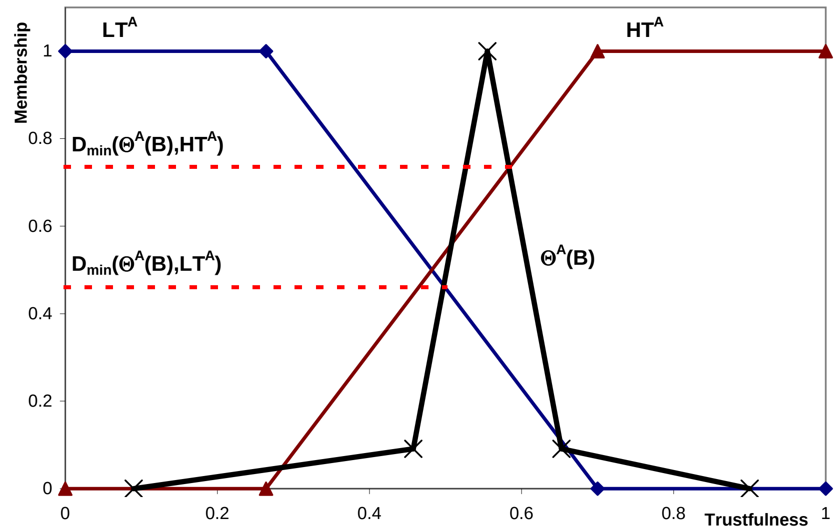


- During **observation**, reference centroids are updated with a weight that decreases with distance.
- During **evaluation**, we aggregate the opinions from nearby centroids with respective weights.

- Trustfulness is not associated with a flow only, but with an **(identity,context)** tuple.
- Identity-Context feature space with appropriate distance function.
- **Identity** is a property of the flow.
- **Context** represents information about other similar flows.
- Centroids are added during the learning process using the **Leader-Follower** algorithm.
- **On-line** process, single parameter required.
- Partially/fully **fixed** centroid positions in our domain.

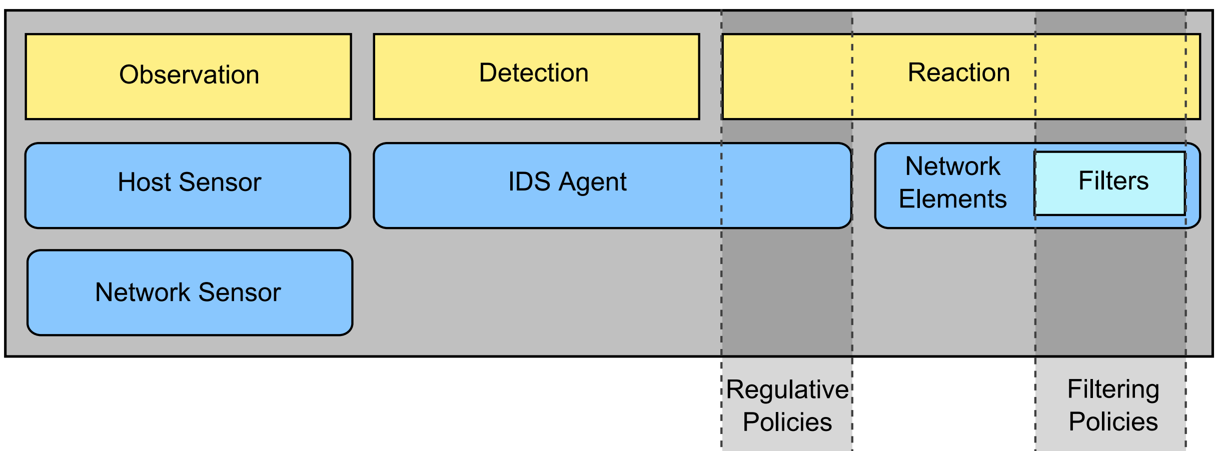
Decision: Trustfulness Evaluation

- Iterative model based on **fuzzy numbers**
- **Outputs**: Score, relative score or binary output.
- **Complexity**: One fuzzy number per each centroid
- **Aggregation** from adjacent centroids in metric space
- **Autonomous adjustment** to natural background alarm level in the system
- Fuzzy trust component based on AFRL project `FA8655-04-1-3044`



Reaction (Reflection)

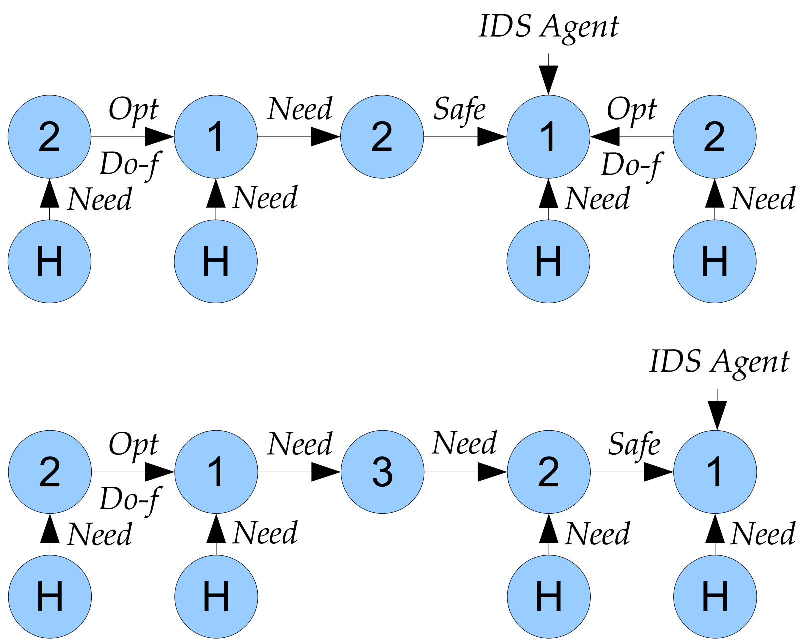
- Filtering Policy Creation (IDS Agents)
- Filter deployment (Network Elements)
  - **collective reflection**
  - **distributed task allocation** to distribute basic assignment of filtering responsibilities
  - **filtering delegation/optimisation** using Extended Contract Net Protocol to optimize allocation of filters between devices



Filter Creation

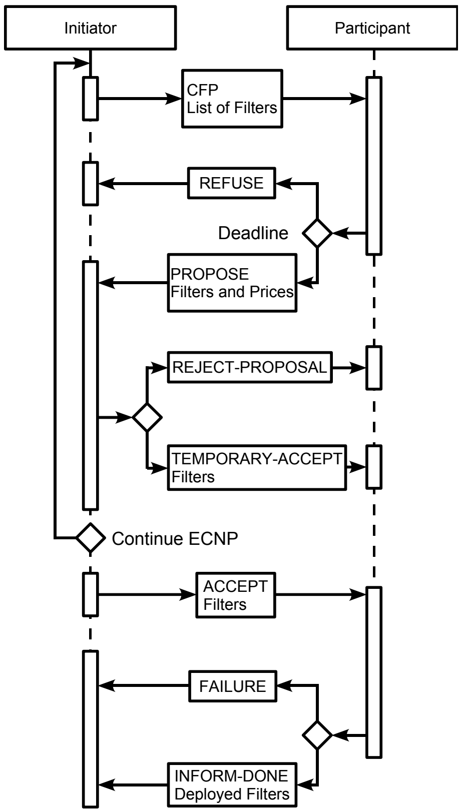
- **Filters** are created for all traffic considered as untrusted (malicious) by the model
- Filters only use the **identity** of one flow – no access to context
- Regulated by **meta-policies**
  - **efficiency** - "*do not create a filter if the centroid is defined by < 20 Flows*"
  - **tradeoffs** - "*local HTTP traffic to server 192.168.2.253 shall always be allowed*"
  - **threat assessment** - "*if the protocol is UDP and number of recent flows in the centroid is high, ban all UDP traffic*"
- Filtering policies are converted into **java code** and compiled; alternative (e.g device specific) bytecodes are feasible
- Filters are conceptually similar to SNORT or other rules/policies: defined by a pattern over packet header and pattern(s) in the application header
- Policy stage can be used to integrate other reaction techniques

Filter Allocation Problem

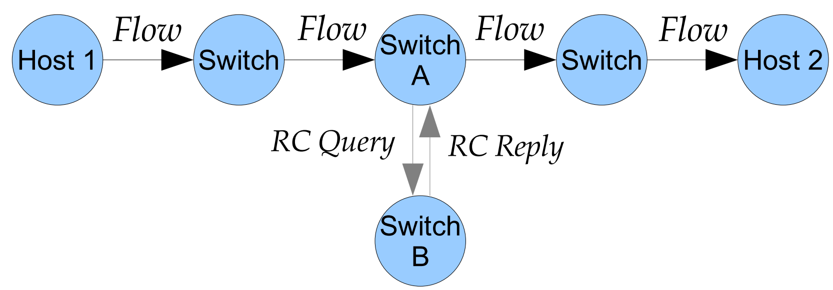


- Assumption: The threat is **already active within** network
- We need to place filter between each pair of vulnerable hosts
- Limitation: device processing power/bandwidth

Delegated Filter Deployment

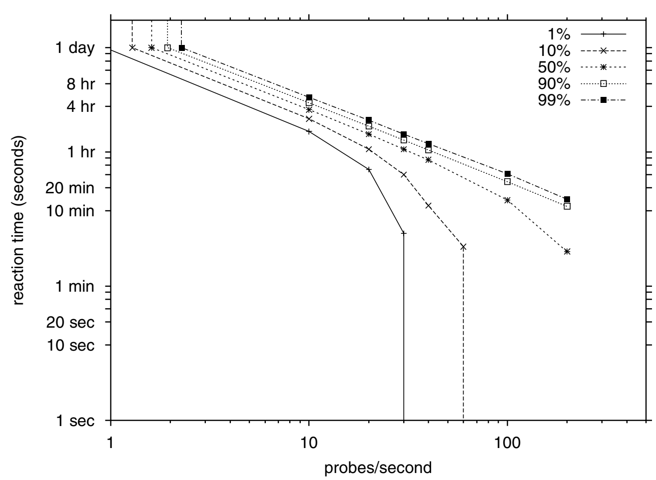


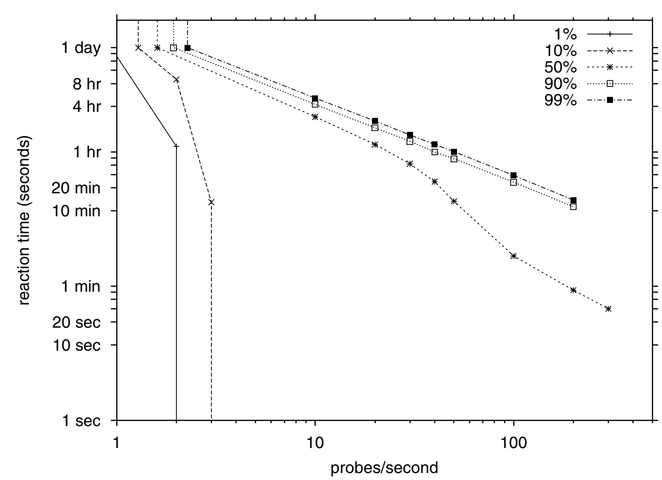
- Delegation of filtering to other network devices
- Requires flow tunnelling for delegated inspection
- We need to coordinate the effort between all agents resolve dependencies - bandwidth
- Use of CNP extension - Extended CNP which allows partial bids, temporary accepts and backtracking



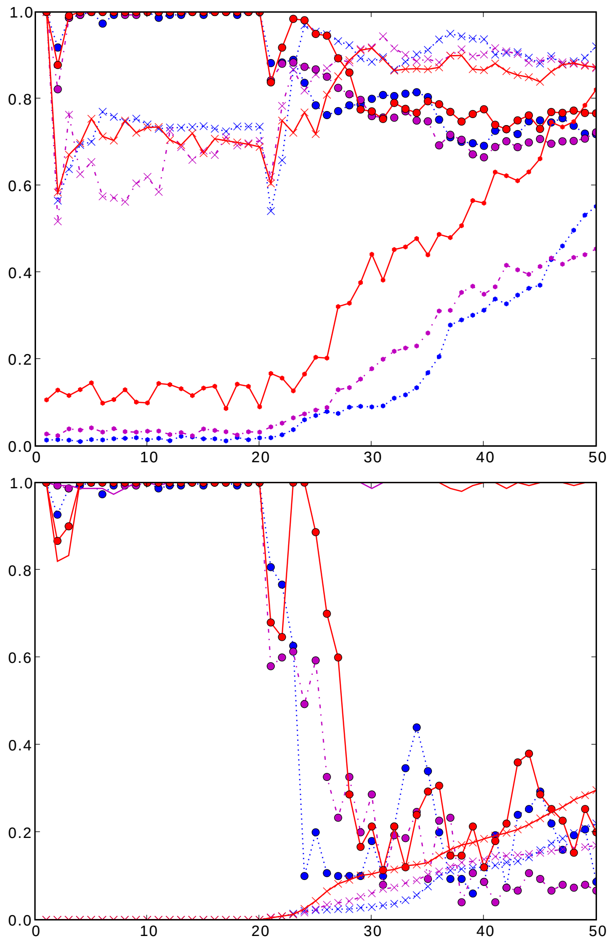
Threat Model: Worm Propagation

- Scanning strategy
- Protocol (TCP/UDP)
- Scanning speed (efficiency vs stealth)
- Requirements from [**Moore03**] (for Internet-Wide infection)





Experimental Results - Detection (Cognition)



- Cognition experiments establish theoretical upper limit on system performance (modulo generalization phenomena)
- Performed on simple mathematical model of worm spread
- Use both Identity and Context information
- Suppose 100 % of flows are filtered by trust model directly
- Results suppose several successive intrusions from the same worm from outside of the network to random addresses inside

Experiment	First worm		Second worm	
Experiment	% Filtered Flows	% Infected Hosts	% Filtered Flows	% Infected Hosts
1	0.86	0.09	0.19	0.93
2	0.93	0.07	0	0.98
3	0.50	0.22	0	0
4	0.65	0.51	0.33	0.54
5	0.88	0.01	0.44	0.86
6	0.93	0.01	0	0.85
7	0.90	0.03	0.93	0.28
8	0.46	0.39	0.11	0.68
9	0.60	0.07	0	0.51
10	0.90	0.13	0.3	0.30
Avg	**0.761**	**0.153**	**0.229**	**0.593**

Table 1: Percentage of infected hosts in experimental runs on identical network. Differences are due to the scanning strategy influence.

Application: Convoy Formation

Conclusions

- Reflective agent techniques allow fast response to novel threats
  - exploit a weak point of worm code: **uninformed spread** and **speed/stealth tradeoff**
  - use feedback from **heterogenous, protected hosts** to improve the results of **anomaly detection** methods
  - evaluated as effective **impact-reduction** technique
- Weaknesses:
  - performance against **stealth** (very slow scanning) threats
  - performance against **multiple threats** launched at once
  - availability of filtering network elements
- Future Work:
  - improve the **detection**
  - further optimize **filter allocation**, combination and deallocation
  - notion of network dynamics
  - study of system autonomy and improved control mechanism