An Event-based Platform for Collaborative Threats Detection and Monitoring

Giorgia Lodi, Leonardo Aniello, Giuseppe A. Di Luna, Roberto Baldoni

Cyber Intelligence and Information Security Research Centre and
Dipartimento di Ingegneria Informatica, Automatica e Gestionale "Antonio Ruberti",
Sapienza University of Rome, Via Ariosto 25, 00185 Rome
Tel: +39 06 77274057, Fax: +39 06 77274002

Abstract

Organizations must protect their information systems from a variety of threats. Usually they employ isolated defenses such as firewalls, intrusion detection and fraud monitoring systems, without cooperating with the external world. Organizations belonging to the same markets (e.g., financial organizations, telco providers) typically suffer from the same cyber crimes. Sharing and correlating information could help them in early detecting those crimes and mitigating the damages.

The paper discusses the Semantic Room (SR) abstraction which enables the development of collaborative event-based platforms, on the top of Internet, where data from different information systems are shared, in a controlled manner, and correlated to detect and timely react to coordinated Internet-based security threats (e.g., port scans, botnets) and frauds. In order to show the flexibility of the abstraction, the paper proposes the design, implementation and validation of two SRs: an SR that detects inter-domain port scan attacks and an SR that enables an online fraud monitoring over the Italian territory. In both cases, the SRs use real data traces for demonstrating the effectiveness of the proposed approach. In the first SR, high detection accuracy and small detection delays are achieved whereas in the second, new fraud evidences and investigation instruments are provided to law enforcement agencies.

Keywords: Collaborative Information Systems, Information Systems Monitoring, Event Processing, Security Threats, Fraud Monitoring

Email address: `[lodi|aniello|diluna|baldoni]@dis.uniroma1.it` (Giorgia Lodi, Leonardo Aniello, Giuseppe A. Di Luna, Roberto Baldoni)

1. Introduction

Threats such as frauds and cyber attacks can have serious and measurable consequences for any organization: lost revenue, downtime, damage to the reputation, damage to information systems, theft of proprietary data or customer sensitive information [1, 2]. Increasingly complex threats are extremely difficult to detect by single organizations in isolation, since their evidence is deliberately scattered across different organizations and administrative domains. This leads many political and technical contexts to strongly believe that "information sharing" among groups of organizations is the correct answer to timely detect and react to such threats with a consequent damage mitigation [3, 4]. The need for information sharing is particularly important for information systems of critical infrastructures such as financial, air traffic control, power grid systems that are undergoing profound technological and usage changes. Globalization, new technological trends, increasingly powerful customers, and intensified competition have brought about a shift in the internal organization of the industry's infrastructure from being confined within the organizational boundaries to a truly global ecosystem characterized by many cross domain interactions and heterogeneous information systems and data.

If we consider the context of cyber security for critical infrastructures, the complex proprietary infrastructures where "no hackers knew" are now replaced by an increasing usage of both the Internet as network infrastructure and off-the-shelf hardware and software with documented and well-known vulnerabilities. Although the sophistication of cyber attacks has increased over time, the technical knowledge required to exploit existing vulnerabilities is decreasing [5]. Attacking tools are often fully automated and the technology employed in many attacks is simple to use, inexpensive and widely available. Because of the increased sophistication of computer attack tools, a higher number of actors are capable of launching effective attacks against IT critical infrastructures. From a sophistication viewpoint, today's attacks are becoming widely distributed in space and time. Distributed in space as attackers are able to launch any types of viruses, trojans, worms in a coordinated fashion involving a large amount of hosts possibly geographically dispersed and belonging to different organizations and administrative domains. They are also distributed in time often consisting of a preparation phase spanning over several days or weeks, and involving multiple preparatory steps [6, 7, 8]. In order to cope with such attacks distribution, information sharing is mandatory.

In the context of fraud monitoring, information sharing is also mandatory to quickly detect and react to such threats. Nowadays, frauds can be so sophisticated and intersected with the cyber space that cooperation among stakeholders and law enforcement agencies is required in order to have an effective, wide and timely global picture of what is going on in the information systems of the collaborating parties. From this constantly changing picture, frauds can be identified in the stakeholders information systems, new intelligence operations can be executed and repression measures can be timely identified and deployed by law enforcement agencies. Needless to say, the delay in the detection of fraud activities is an important parameter for mitigating their damages.

As remarked by some security studies [2], information sharing should be preferably carried out by (potentially competing) organizations belonging to the same markets (e.g., financial organizations, telco providers, power grid providers) as they typically suffer from the same vulnerabilities, cyber crimes and frauds. In the context of financial organizations FS/ISAC in USA [9], Presidio Internet in Italy and OLAF in EU [10] are examples of information sharing bodies for cyber crimes and fraud detection. All of them facilitate sharing of information pertaining to cyber threats, vulnerabilities, incidents, frauds, potential protective measures and practices. This is usually achieved through sending information containing the potential threat to both analysts for reviewing and scoring and financial services sector experts. Once a vulnerability or incident is analyzed, it can be categorized as Normal, Urgent, or Crisis, depending on the risk to the financial services sector. After the analysis, alerts are delivered to participants. Alerts typically contain not only the details of the threat but also information about how to mitigate the threat. Needless to underline that all this human-based procedure (i) takes a period of time to be executed that is usually much larger than the time required to deploy the attack and (ii) can work only on an amount of data manageable by humans.

There is then an urgent need to enhance these bodies with suitable collaborative software platforms that are able to combine and correlate in a timely manner a large volume of data coming from multiple information systems that collectively can show an evidence of a threat (e.g., attack, fraud, incident, vulnerability) and then categorize the threat in an automatic way. This paper targets precisely this objective.

A number of research works have focused on the study of collaborative systems for detecting massive large scale security threats [11, 12]. However, possibly competitor distrusting organizations can be reluctant to fully adopt such collaborative approach as this may imply sharing potentially sensitive information (e.g., financial transactions, users identities). In these competitive contexts, a controllable platform should be developed, which is capable of meeting the necessary guarantees (e.g., reliability, availability, privacy) for data and resource management. Contracts established among the parties should be set up which specify the guarantees to be met. Additionally, the platform has to be flexible enough to be easily customized for different contexts, from detection of cyber attacks to fraud monitoring. With these guarantees, it is likely that even distrusting organizations can be motivated in making available their data for collaborative detection of massive threats.

This paper discusses the design of a flexible programming abstraction named Semantic Room (SR)[1]. The SR enables the construction of collaborative and contractually regulated event-based platforms where aggregation and correlation of data coming from the different information systems of the organizations participating in the SR can be carried out with the aim of providing early detection of attacks and frauds. Events correlation can be also enriched by using external information repositories. Each SR has a specific strategic objective to meet (e.g., botnet detection, fraud monitoring) and is associated with both a contract, which specifies the set of rights and obligations for governing the SR membership, and different software technologies that are used to carry out the data processing and sharing within the SR.

The principal contributions of this paper can be then summarized as follows.

- We describe the design, implementation and experimental evaluation of an SR whose aim is to detect attackers performing port scanning activities against organizations participating in the SR itself. In particular, in order to show the flexibility offered by the SR abstraction, we present two different implementations of this SR. The first implementation uses the Complex Event Processing (CEP) engine Esper [13] for realizing its processing and correlation capabilities. Esper is properly instrumented through a set of queries written in Esper's Event Processing Language (EPL). The second implementation deploys a distributed CEP engine called Storm [44] we use to implement the

[1]A preliminary discussion of the SR abstraction and its lifecycle management have been presented in [25]. This paper introduces its successive evolutions, and examples of SRs applicable to real systems and applications, in a more comprehensive framework.

earlier mentioned correlation algorithm. This implementation allows us to assess scalability results by varying the number of participants and to compare the achieved performances to those obtained with the same algorithm implemented in Esper's EPL.

- We propose the design, implementation and a user-level evaluation of an SR devoted to the monitoring of counterfeit euros, tampered ATMs and unauthorized Points Of Sale (POSs). The SR uses Esper and correlates information coming from Italian banks and other financial institutions; it mash-up this information with a location service that is external to the SR so as to provide new evidences to law enforcement agencies about crime hotspots and propagation of frauds in space and time. The SR also provides privacy-preserving mechanisms in order to protect sensitive data.

The remaining of this paper is structured as follows. The next section discusses relevant related work. Section 3 describes the SR abstraction. Section 4 describes the design, implementation and experimental evaluation of an event processing system to be deployed on the top of the SR for inter-domain stealthy port scan detection. An alternative implementation using the distributed event processing engine Storm is also described. Section 5 proposes the design, implementation and a user-level evaluation of an online location intelligence SR for fraud monitoring. Finally, Section 6 concludes this paper and outlines possible future extensions.

2. Related Work

As overviewed in [14], detecting event patterns, sometime referred to as situations, and reacting to them are in the core of Complex Event Processing (CEP) and Stream Processing (SP), both of which play an important role in the IT technologies employed by the financial sector. In particular, IBM System S [15] has been used by market makers in processing high-volume market data and obtaining low latency results as reported in [16]. System S, as other CEP/SP systems such as [17], [18], is based on event detection across distributed event sources. Our solution employs open source CEP systems such as JBoss Drools [19] and Esper [13]. JBoss Drools is a Business Logic Integration Platform which provides a unified and integrated platform for Rules, Workflow and Event Processing. Esper is a CEP engine technology that processes events and discovers complex patterns among multiple streams of event data. As our solution did not require an entire Business

Logic Platform as provided by JBoss Drools, we chose the lightweight Esper engine for our research.

Coping with many event sources that generate heavy event load can become critical when using centralized CEP solutions such as Esper. In the literature, there exist software solutions that are able to distribute the computation itself among more resources and to scale out as the input load increases. One of these solutions is provided by IBM SPADE, the System S declarative stream processing engine [43] which allows programmers to define the processing logic using an intermediate language for flexible composition of parallel and distributed data-flow graphs. More recent solutions such as Storm [44] are available. Storm is an open source distributed CEP engine implemented in Java and Clojure that allows programmers to model the required computation as a graph where nodes represent processing blocks and edges represent event streams connecting processing blocks. In contrast to System S, Storm does not provide any language to define the processing logic. In our SR for Intrusion detection we preferred to use a distributed CEP engine that is open source, as in the case of Storm, in order to cope with SR scalability issues.

The issue of using massive CEP among heterogeneous organizations for detecting network anomalies and failures has been suggested and evaluated in [20]. Also the usefulness of collaboration and sharing information for telco operators with respect to discovering specific network attacks has been pointed out in [11, 12]. In these works, it has been clearly highlighted that the main limitation of the collaborative approach concerns the confidentiality requirements. These requirements may be specified by the organizations that share data and can make the collaboration itself hardly possible as the organizations are typically not willing to disclose any private and sensitive information. In our platform, the SR abstraction can be effectively used in order to build a secure and trusted social business environment, which can be enriched with the degree of privacy needed by the participants where a massive and collaborative event processing computation can occur.

In addition, collaborative approaches addressing the specific problem of Intrusion Detection Systems (IDSs) have been proposed in a variety of works [21, 24, 26, 27]. Differently from singleton IDSs, collaborative IDSs significantly improve time and efficiency of misuse detections by sharing information on attacks among distributed IDSs from one or more organizations [28]. The main principle of these approaches is that there exist local IDSs that detect suspects by analyzing their own data. These suspects are then disseminated using possibly peer to peer links.

This approach however exhibits two main limitations: (i) it requires suspect data to be freely exchanged among the peers and (ii) it does not fully exploit the information seen at every site. The former limitation can be very strong due to the data confidentiality requirements that are to be met. The latter limitation can affect the detection accuracy as emerges from an assessment of commercial solutions such as distributed Snort-based [22] or Bro-based IDSs [23]. These systems propose the correlation of alerts produced by peripheral sensors. Such alerts are generated using local data, only. The information that is cut away by the peripheral computations could bring them to miss crucial details necessary for gaining the global knowledge required to detect inter-domain malicious behaviors.

Our solution addresses precisely this issue through the usage of a general-purpose CEP that offers great flexibility to the management of the detection logic. The need for ease of deployment also drove us to choose among Java-based CEP systems. In addition, no privacy, security and performance requirements are considered in those solutions when gathering and processing the data. In contrast, the SR abstraction we describe is built in order to associate with it a contract where specific requirements on security, performance and privacy can be defined. The SR aims at processing data injected into it so as suspects are detected through the exploitation of all the data made available by every participating organization. The main advantage of this approach is that the detection accuracy can be sharpened, as also shown by the experimental evaluation we report in this paper. Moreover, we show how the SR abstraction can be flexibly designed and implemented in order to meet performance and privacy requirements by presenting an SR where a distributed processing is employed in order to face scalability issues, and an SR where privacy-preserving mechanisms are enabled in order to protect sensitive data.

3. The Semantic Room abstraction

The SR abstraction enables the construction of private and trusted collaborative event-based platforms through which organizations (e.g., financial institutions) can federate for the sake of distributed data aggregation and near real time data correlation. The organizations participating in an SR can exploit it in order to effectively monitor distributed IT infrastructures and timely detect frauds and threats. The participating organizations are referred to as the *members* of the SR.

A Semantic Room is defined by the following three elements:

- *objective*: each SR has a specific strategic objective to meet. For instance, there can exist SRs created for large-scale stealthy port scans detection, or SRs created for detecting web-based attacks such as SQL injection or cross site scripting;

- *contract*: each SR is regulated by a contract that defines the set of processing and data sharing services provided by the SR along with the data protection, privacy, isolation, trust, security, dependability, and performance requirements. The contract also contains the hardware and software requirements a member has to provision in order to be admitted into the SR. According to state of the art on contracts definition [30], an SR contract includes four main parts; namely, the details of the parties involved in the SR contract, a number of contractual statements, a number of so-called Service Level Specifications (SLS) and finally, the signatures of the involved parties;

- *deployments*: The SR abstraction is highly flexible to accommodate the use of different technologies for the implementation of the SR logic. In particular, the SR abstraction can support different types of system approaches to the processing and sharing; namely, a centralized approach that employs a single server (e.g., Esper [13]), a decentralized approach where the processing load is spread over all the SR members (e.g., a MapReduce-based processing [29]), or a hierarchical approach where a pre-processing is carried out by the SR members and a selected processed information is then passed to the next layer of the hierarchy for further computations.

Figure 1 illustrates the high level design of the Semantic Room. As shown in this Figure, the SR abstraction supports the deployment of two principal building blocks; namely, Complex Event Processing and Applications, and Data Dissemination. These components can vary from SR to SR depending on the software technologies used to implement the SR processing and sharing logic. In addition, SR management building blocks are to be employed in order to manage the SR lifecycle and monitor the adherence to the SR contract by its members.

SR members can inject raw data into the SR by means of SR Gateways components (see below) deployed at the administrative boundaries of each SR member. Raw data may include real time data, inputs from human beings, stored data (e.g., historical data), queries, and other types of dynamic and/or static content that are processed in order to produce required output. Raw data are properly pre-processed by SR Gateways in order to normalize them and satisfy privacy requirements as prescribed by the SR contract. Processed data can be used for internal consumption within the

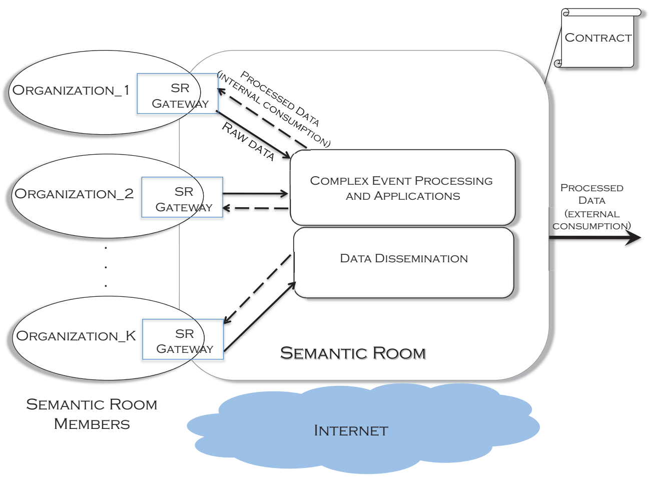


Figure 1: The Semantic Room Abstraction

SR: in this case, derived events, models, profiles, blacklists, alerts and query results can be fed back into the SR (the dotted arrows in Figure 1) so that the members can take advantage of the intelligence provided by the processing. The processed data are made available to the SR members via their SR Gateways; SR members can then use these data to properly instruct their local security protection mechanisms in order to trigger informed and timely reactions independently of the SR management. In addition, a (possibly post-processed) subset of data can be offered for external consumption. SRs in fact can be willing to make available for external use part of their output data. In this case, in addition to the SR members, there can exist clients of the SR that cannot contribute raw data directly but can simply consume a portion of the SR output data.

SR members have full access to both the raw data members agreed to contribute to by contract, and the data being processed and thus the output produced by the SR. Data processing and results dissemination are carried out by SR members based on obligations and restrictions specified in the above mentioned contract. In [67] we describe the overall architecture and a thorough description of its software components.

The rest of this paper mainly focuses on computational and communication aspects within an SR; we do not investigate other practical aspects such as SR fault tolerance. Classical techniques based on replication could be employed to ensure service continuity.

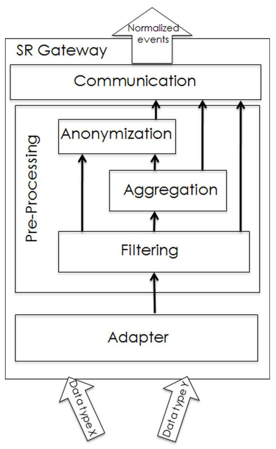


Figure 2: The SR Gateway

3.1. SR Gateway

An important component of the SR abstraction is represented by the SR Gateway. It is typically hosted within the partners' IT boundaries and it acts as an access point for SR members' local management systems. It is responsible for adapting the incoming raw data to the formats consumable by the SR CEP, and sanitizing the data according to privacy constraints. Both the formats and privacy requirements are specified in the SR contract, as previously described. Thus, the SR Gateway component, along with the processing and data dissemination components, can vary from SR to SR.

Figure 2 shows the principal modules that constitutes the SR Gateway. An adapter is used to interface SR members' local systems management. The adapter is designed so as to interface possibly heterogeneous data types (e.g., data from databases, data from networks, data included into files) and to convert the data in the specific SR's format. The adapter then dispatches the normalized data to a pre-processing module of the SR Gateway which performs the following three main functions: (i) filtering, which selects only the data of interest for the SR objective to meet; (ii) aggregation, which can reduce the amount of data being sent to the processing building block of the SR (iii) anonymization, which sanitizes sensitive data before they leave the internal network of the SR members. Data format conversion and data aggregation typically constitute the so-called data unification process, as defined in [52].

Note that, some of the pre-processing functions (and the pre-processing module itself) can be activated in the Gateway only if required; that is, according to specific SR contract clauses. For instance, if the contract states that some sensitive data must be anonymized before being injected into the SR, the anonymization sub-module is properly enabled in the Gateway. Section 5 describes an example of SR in which an anonymization process is performed by the SR Gateway.

The final output of the pre-processing (normalized events) is then sent to a communication module (or proxy) of the Gateway which is responsible for injecting it into the SR (Figure 2). The communication module can be properly configured to embody specific communication protocols when injecting the data to the SRs. Section 5 describes an example of SR where the communication module has been designed to employ a privacy-preserving communication protocol.

3.2. Enforcing Privacy Requirements

As explained in Section 2, one of the main obstacles to the adoption of collaborative environments is the difficulty to meet confidentiality requirements. Such requirements consist in avoiding that sensitive information are inferred by inspecting computation input/output or by analyzing the traffic within the SR. Coping with these issues strictly depends on the level of trust among SR members.

In case a Trusted Third Party (TTP) is agreed upon by all the members and such TTP provides the computational infrastructure, input data can be directly sent to the TTP, which carries out the required elaboration and then disseminates the results to SR members. What is missing in this solution concerns how to prevent information leakage from the output diffused to SR members. Even though addressing this problem is very application-specific, several general techniques are described in the literature based on controlled perturbation or partial hiding of the output of a computation (association rule hiding [55, 56], downgrading classifier effectiveness [57, 58], query auditing and inference control [59, 60]. The side effect of these approaches is the worsening of the accuracy of the result, which is usually dealt with by focussing and leveraging the existing tradeoffs between the required level of privacy and the necessary accuracy of the output; a thorough analysis of such tradeoffs is reported in some of the earlier cited works. An additional solution is the so-called Conditional Release Privacy-preserving Data Aggregation (CR-PDA), presented in [48], which consists in disclosing only the items of the output that satisfy specific conditions (i.e., the item is the IP address of a malicious host). This class of scenarios characterized by the presence of a TTP in charge of executing the computation can be suitable for the SR abstraction, since the TTP can become an SR member and the SR Gateways can be configured so that all the other members send input data to the TTP and then receive back the results, which are properly anonymized by the TTP itself on the basis of the techniques just cited.

If no TTP can be employed, some additional mechanisms are required to anonymize both input data and the identity of the member which provided some specific datum. As for the anonymization of input data, while the fields of the single input datum that do not contribute to the final result can be filtered out before being fed into the SR (thanks to the Filtering module of the SR Gateway), the other fields that contain sensitive information have to be adequately modified. Several techniques exist for dealing with this issue: adding noise to input data (randomization method [61, 62]) and reducing the granularity of data representation so as to make any item indistinguishable from a certain number of other items (k-anonymity [63] and l-diversity [64]). The cited papers also describe what kinds of computations can be performed on an input perturbed in this way and analyze the tradeoff between the provided privacy level and the correctness of the obtained results. These techniques can be integrated within the SR model by conveniently implementing the anonymization module of the SR Gateway so that the required perturbation is applied to input data before being injected into the SR. Techniques from the field of secure Multi-Party Computation (MPC) can be also employed, which allow for the anonymization of provided inputs by leveraging encryption and routing mechanisms to be enforced by the participants of the computation [50, 49, 51, 48]. When there is only the need to make anonymous the physical host that provides an event, lighter and faster techniques could be used [45, 51]. Specific routing algorithms can be implemented directly into the Communication module of the SR Gateway. For what concerns the anonymization of the output, the same observations and solutions reported for the scenario with a TTP hold.

In essence, the choice of the solution to adopt strongly depends on the specific application and on the privacy requirements. In the most general case, input data has to be filtered (Filtering module), encrypted and/or perturbed (Anonymization module) and a proper routing on some specific layout has to be enforced (Communication module).

4. A Semantic Room for Intrusion Detection

We present a specific SR, which we refer to as *ID-SR*, whose objective is to prevent potential intrusion attempts (Intrusion Detection) by detecting malicious inter-domain stealthy SYN port scan activities. Our goal here is to show the flexibility of the SR abstraction to accommodate a variety of different implementations and we do not concentrate on other aspects of the SR such as privacy. Thus, in this specific case, we assume that SR members trust each others and we propose two different implementations of the ID-SR: one based on Esper and a second implementation that uses Storm (interested readers can also find another SR implementation based on the Map-Reduce paradigm in [66]).

4.1. Inter-domain stealthy SYN port scan

The goal of the attack is to identify open TCP ports at the attacked SR members. The ports that are detected as opened can be used as intrusion vectors at a later time. In this paper we focus on an attack carried out by a single host, attacks where multiple hosts are used in a coordinated way are addressed in [53].

The attack is carried out by initiating a series of low volume TCP connections to ranges of ports at each of the targeted SR members. Specifically, the attack we consider is named TCP SYN (half-open) port scan, which span different administrative domains (the different domains of the SR members). We call this attack *inter-domain SYN port scan*.
It is characterized by probe connections that are never completed; that is, the three-way TCP handshake is never accomplished. Let us consider a single scanner S, a target T and a port P to scan. S sends a SYN packet to $T : P$ and waits for a response. If a SYN-ACK packet is received, S can conclude that $T : P$ is open and can optionally reply with an RST packet to reset the connection. In contrast, if an RST-ACK packet is received, S can consider P as closed. If no packet is received at all and S has some knowledge that T is reachable, then S can conclude that P is filtered. Otherwise, if S does not have any clue on the reachability status of T, it cannot assume anything about the state of P.

Note that not all the scans can be considered as malicious. For instance, there exist search engines that carry out port scanning activities in order to discover Web servers to index [31]. It becomes then crucial to distinguish accurately between actual malicious port scanning activities and benign scans, thus minimizing so-called *false positives*; i.e., normal port scanning activities which have been erroneously considered as malicious. In the next subsection we describe one possible algorithm, R-SYN, that aims at improving the detection accuracy of the ID-SR. We implemented this algorithm into two different systems. Other algorithms for SYN port scan detection are possible, e.g., see [36] for an algorithm based on line-fitting and its comparison with R-SYN.

4.2. Rank-based SYN port scan detection algorithm

The R-SYN algorithm adapts and combines in a novel fashion three different port scan detection techniques; namely, *Half open connections*, *Horizontal and Vertical port scans*, and *Entropy-based failed connections* so as to augment the chances of detecting real malicious scan activities. The three techniques are described below in isolation.

Half open connections. This technique analyzes the sequence of SYN, ACK, RST packets in the three-way TCP handshake. Specifically, in normal activities the following sequence is verified (i) SYN, (ii) SYN-ACK, (iii) ACK. In the presence of a SYN port scan, the connection looks like the following: (i) SYN, (ii) SYN-ACK, (iii) RST (or nothing) and we refer to it as an *incomplete connection.* For a given IP address, if the number of incomplete connections is higher than a certain threshold T_{HO} (see below), we can conclude that the IP address is likely carrying out malicious port scanning activities.

DEF: for a given IP address x, let $count_{HO}(x)$ be the number of incomplete connections issued by x; we define $HO(x)$ as follows:

$$HO(x) = \begin{cases} 1 & \text{if } count_{HO}(x) > T_{HO} \\ 0 & \text{otherwise} \end{cases}$$

Horizontal and vertical port scans. In a horizontal port scan the attackers are interested in a port across all IP addresses within a range. In a vertical port scan, attackers scan some or all the ports of a single destination host [32].

In our R-SYN algorithm we adapt the Threshold Random Walk (TRW) technique described in [31]. TRW classifies a host as malicious observing the sequence of its requests. Looking at the pattern of successful and failed requests of a certain source IP, it attempts to infer whether the host is behaving as a scanner. The basic idea is modeling accesses to IP addresses as a random walk on one of two distinct stochastic processes, which correspond to the access patterns of benign source hosts and malicious ones. The detection problem then boils down to observing a specific trajectory and inferring from it the most likely classification for the source host. While the original technique considers as a failure a connection attempt to either an unreachable host or to a closed port on a reachable host, we adapt the TRW technique in order to distinguish between them, since the former concern horizontal port scans whereas the latter to vertical port scans. We then designed two modified versions of the original TRW algorithm in order to take into account these aspects.

Specifically, in order to detect horizontal port scans, we identify connections to both unreachable and reachable hosts. Hosts are considered unreachable if a sender, after a time interval from the sending of a SYN packet, does not receive neither SYN-ACK nor RST-ACK packet, or if it receives an ICMP packet of type 3 that indicates that the host is unreachable. In contrast, hosts are reachable if a sender receives SYN-ACK or RST-ACK packets. For each source IP address we then count the number of connections to unreachable and reachable hosts and apply TRW algorithm. Let $TRW_{HS}(\text{x})$ be the boolean output of our TRW algorithm version for horizontal port scans computed for a certain IP address x. "True" output indicates that x is considered a scanner. Otherwise x is considered a honest host.

DEF: for a given IP address x, we define $HS(x)$ as follows:

$$HS(x) = \begin{cases} 1 & \text{if } TRW_{HS}(\text{x}) == \text{true} \\ 0 & \text{otherwise} \end{cases}$$

In order to detect vertical port scans, we first identify connections to open and closed ports. We then count such connections for each source IP address. Let $TRW_{VS}(\text{x})$ be the boolean output of our TRW algorithm version for vertical port scans computed for a certain IP address x.

DEF: for a given IP address x, we define $VS(x)$ as follows:

$$VS(x) = \begin{cases} 1 & \text{if } TRW_{VS}(\text{x}) == \text{true} \\ 0 & \text{otherwise} \end{cases}$$

Entropy-based failed connections. As previously noted, not all suspicious activities are actually performed by attackers. There exist cases in which the connections are simply failures and not deliberate malicious scans.

As in [33], in order to discriminate failures from malicious port scans, we use an entropy-based approach. In contrast to [33], we evaluate the entropy by considering the destination endpoint of a TCP connection; that is, destination IP and destination port. The entropy assumes a value in the range $[0, 1]$ so that if some source IP issues failed connections towards the same endpoint, its entropy is close to 0; otherwise, if the source IP attempts to connect without success to different endpoints, its entropy is close to 1. This evaluation originates by the observation that a scanner source IP does not repeatedly perform the same operation towards specific endpoints: if the attempt fails a scanner likely carries out a malicious port scan towards different targets.

Given a source IP address x, a destination IP address y and a destination port p, we define $failures(x, y, p)$ as the number of failed connection attempts of x towards $y : p$. For a given IP address x, we define $N(x)$ as follows:

$$N(x) = \sum_{y,p} failures(x, y, p)$$

In addition, we need to introduce a statistic about the ratio of failed connection attempts towards a specific endpoint. We define $stat(x, y, p)$ as follows:

$$stat(x, y, p) = \frac{failures(x,y,p)}{N(x)}$$

The normalized entropy can then be evaluated applying the following formula:

DEF: for a given IP address x,

$$EN(x) = -\frac{\sum_{y,p}(stat(x,y,p) * \log_2(stat(x,y,p)))}{\log_2(N(x))}$$

***Ranking**.* All the above statistics are collected and analyzed across the entire set of the ID-SR members, thus improving chances of identifying low volume activities, which would have gone undetected if the individual participants were exclusively relying on their local protection systems (e.g., Snort Intrusion Detection system [34]). In addition, to augment the probability of detection, the suspicious connections are periodically calibrated through a ranking mechanism which minimizes the probability that a scanner cheats by behaving apparently in a good way.

Our ranking mechanism sums up the three values related to half opens, horizontal port scans and vertical port scans and weights the total result using the entropy-based failed connections.

DEF: for a given IP address x, we define $rank(x)$ as follows:

$$rank(x) = (HO(x) + HS(x) + VS(x)) * EN(x)$$

Such ranking is compared ($\geq$) to a fixed threshold in order to mark an IP address as scanner. IP addresses marked as scanners are placed in a blacklist, which is then disseminated among ID-SR members.

4.3. Esper Intrusion Detection Semantic Room

The Esper ID-SR consists of a number of pre-processing and processing elements hosted on a cluster of machines allocated from the ID-SR hardware pool (as prescribed by its contract). Specifically, the pre-processing elements are the SR Gateway components deployed at each SR Member

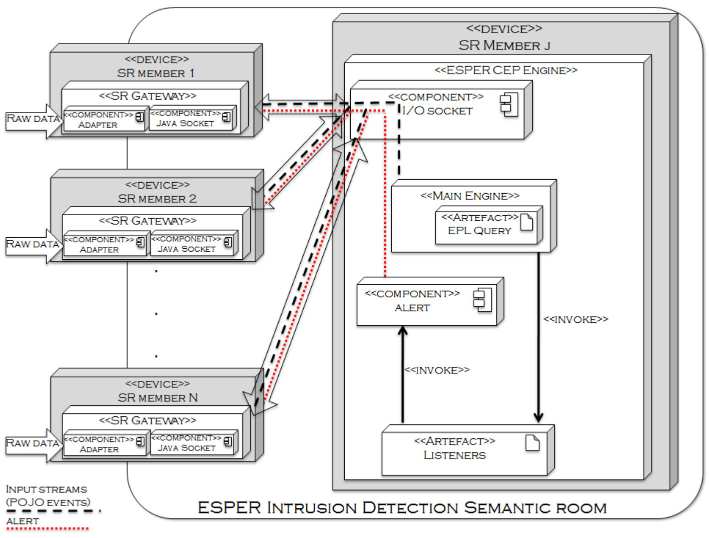


Figure 3: The Esper Intrusion Detection Semantic Room

site; the processing element is represented by a (CEP) system. The individual components of the Esper ID-SR are illustrated in Figure 3 and described in detail below.

SR Gateway. Raw data owned by ID-SR Members can be potentially originated by different sources such as databases, logs, or traffic captured from the internal members' networks. In order to be analyzed by the Esper engine, the data are to be normalized and transformed in Plain Old Java Objects (POJOs). To this end, the SR Gateway has been designed and implemented so as to incorporate an adapter component that (i) takes as input the flows of raw data (see Figure 3), (ii) pre-processes them through filtering operations (for instance, we filter packets related to TCP three-way handshaking only) in order to transform the data in the format prescribed by the SR contract, and, finally (iii) wraps the pre-processed data in POJOs so that they can be analyzed by Esper. We implemented TCPPojo for TCP packets and ICMPPojo for ICMP packets. Each POJO maps every field in the header of the related protocol. POJOs are normalized and sent through Java sockets to Esper. When sending the POJOs our design and implementation maintain the order of the captured packets, which is crucial when evaluating sequence operators in the EPL queries of the Esper engine.

Complex Event Processing. The CEP in the SR is carried out by the well known open source Esper engine [13]. The engine is fully implemented in Java; it receives POJOs that represent the events it has to analyze (input streams). The processing logic is specified in a high level language similar to SQL named Event Processing Language (EPL). In order to detect malicious port scan activities a number of EPL queries are defined and executed by the Esper engine, as shown in Figure 3. EPL queries run over continuous streams of POJOs; when an EPL query finds a match against its clauses in its input streams, its related listener is invoked. A *listener* is a Java object that can be subscribed to a particular stream so that whenever the query outputs a new tuple for that stream, the *update*() method of the listener is invoked. The listeners can execute specific functions (e.g., the entropy computation) or can simply raise an alert that can be further elaborated in order to be disseminated to ID-SR members.

4.3.1. The Esper ID-SR Processing Steps

The processing steps followed by the Esper ID-SR implementation can be summarized as follows. At the fist step, the raw data capturing the networking activity at each of the SR Members are collected. We have implemented the SR Gateway component so that to collect data at real time by running the tcpdump utility and to collect network traces (i.e., log files). Each Gateway at each Esper ID-SR Member normalizes the incoming raw data producing a stream of TCPPojo records of the form ⟨SOURCEIP, DESTINATIONIP, SOURCEPORT, DESTINATIONPORT, FLAGTCP, SEQUENCENUMBER, ACKNUMBER⟩. A Java socket is opened towards the Esper engine through which the TCPPojo records are sent. The incoming TCPPojo records are retrieved by a Java socket component implemented inside Esper (see Figure 3) and passed to the main engine for correlation purposes.

Each detection technique is implemented using a combination of EPL queries and listeners. The EPL queries are in charge of recognizing any packet patterns of interest according to the detection technique. Listeners are invoked whenever such matches are verified. For instance, in order to identify incomplete connections we use the EPL query introduced in Listing 1.

Listing 1: EPL query for half open connections

```
insert into halfopen_connection
  select ...
  from pattern [
    every a = syn_stream --> (
      ( b = syn_ack_stream(...) --> (
        (timer:interval(60 sec) or <c>) and not <d>
        ) where timer:within(61 sec) ) ) ]
```

In this query, we exploit the *pattern* construct of Esper to detect patterns of incomplete connections. In particular, a is the stream of SYN packets, b is the stream of SYN-ACK packets, $< c >$ is the stream of RST packets and $< d >$ is the stream of ACK packets, all obtained through filtering queries. Such pattern matches if the involved packets are within a time window of 61 seconds.

Further queries are then used and bound to Listeners: they filter IP addresses that made more than T_{HO} (in our implementation, we set it to 2) incomplete connections and update the data structure representing the list of half open connections. Interested readers can refer to [35] for the comprehensive implementation in EPL of the R-SYN detection algorithm.

IP addresses marked as scanners are placed in a blacklist (the alert component of Figure 3 performs this task); the blacklist is sent back to the SR Gateway components through Java socket so that the Esper SR-ID members can take advantage of the intelligence produced by the SR.

4.4. Storm Intrusion Detection Semantic Room

The R-SYN algorithm described in Section 4.3 is implemented using a centralized CEP engine. Although Esper is claimed to provide impressive performances in terms of processing latency and event throughput, it also exhibits the typical weaknesses of a centralized system.

A key aspect of a collaborative environment like the SR is the ability to change the membership as the need arises. In case new members join an SR, the SR should reconfigure available resources to adapt to the new situation, in particular it should be able to sustain the event streams provided by new participants without any sensible degradation of the performances. This requires the employment of a CEP engine which can scale in/out to arrange the computational power required to process the input data volume generated by SR members. Once the members altogether produce a traffic higher than a certain threshold, a centralized CEP engine may become a bottleneck, worsening the overall performances.

Another well known issue of centralized architectures is their difficulty of supplying fault tolerance. In the specific case of Esper there is an (non open source) extension called EsperHA (Esper High Availability) which employs checkpointing and hot backup replicas to face possible failures. The required queries and the state of the computation are stored in some stable storage and recovered by replicas as required. Storing such information can have a detrimental effect on the performances, which makes Esper's efficiency even worse.

A viable solution for these two problems is the usage of a distributed CEP engine able to seamlessly scale to adapt to input event rate and efficiently provide robust mechanisms to meet fault tolerance requirements. In this Section we present a software called Storm which provides the building blocks for carrying out a computation in a distributed fashion. We create an SR for intrusion detection that uses Storm to implement R-SYN algorithm.

4.4.1. Storm: a distributed CEP engine

Storm [44] is an open source distributed CEP engine implemented in Java and Clojure that allows programmers to model the required computation as a graph where nodes represent *components* (i.e. processing blocks) and edges represent event streams connecting components. According to Storm's jargon, a component is classified as a *spout* if it has no incoming edge; that is, it is an event source, otherwise it is called a *bolt*. The graph resulting from a certain configuration of components and streams is referred to as a *topology*.

Each component can be replicated in order to sustain the event rate of some critical stream and provide some fault tolerance guarantee. A replica of a component is referred to as a task. The semantic of a stream connecting replicated components can be configured by specifying a so called *stream grouping*, so as to meet the specific requirements and characteristics of the computation to implement. For example, if the replication is aimed at enabling load balancing, then the stream can be setup to partition the events among available tasks. If the destination bolt of a stream computes a stateful operation, then the *field grouping* has to be employed, which allows tuples related to the same entity to be always delivered to the same task of the component. Otherwise, if the operation is stateless, *shuffle grouping* should be used, which evenly distributes tuples to destination tasks in a round-robin fashion. In case the replication of a component is intended to make the topology fault tolerant, the stream can be configured so as to replicate events between tasks (*all grouping*), thus employing required data redundancy. Storm also provides other mechanisms to cope with failures. It is possible to setup a topology so as to track each event originating from a spout together with the whole tree of further events. This permits to detect whether such a processing fails somewhere before all derived events have been fully processed, and possibly to replay the input event from the spout. Storm also monitors all its software processes to detect any failure and restart those that crashed.

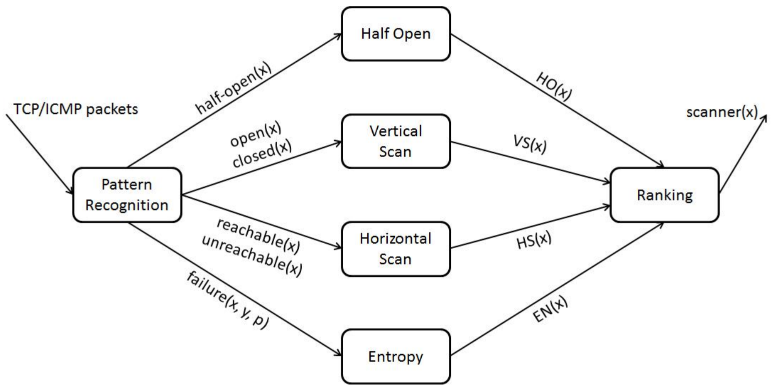


Figure 4: Computation graph of the R-SYN algorithm

The tasks of the components of a topology can be deployed over the available resources to properly harness their computational power. In case any change occurs either in the load to sustain or in the resources at disposal, the deployment can be updated accordingly [54]; this perfectly fits with the requirements of a collaborative environment where both the membership and the input load can vary over time.

4.4.2. R-SYN algorithm in Storm

The first step towards the implementation of an algorithm in Storm is the definition of the related topology; that is, the computation graph which characterizes the algorithm itself. Figure 4 shows such a computation graph for the R-SYN algorithm.

The *Pattern Recognition* component is the unique spout of the topology. It reads TCP/ICMP packets from the network to be monitored and performs pattern recognition tasks for producing its output streams. The component is replicated: each SR member has a Pattern Recognition task and can provide its own input data to the SR. Its implementation is done by embedding in each task an Esper instance which is in charge of recognizing the patterns of interest. Since all the bolts are stateful, all the stream groupings are setup as *field grouping*.

The *Half Open* bolt receives events representing the occurrences of half open connections made by some host x (half-open(x)), and maintains a state for tracking how many half open attempts have been issued by any source host. In case a predefined threshold is exceeded for a certain host x, then an event $HO(x)$ is sent to the *Ranking* bolt.

The *Vertical Scan* (*Horizontal Scan*) bolt receives events modeling con-

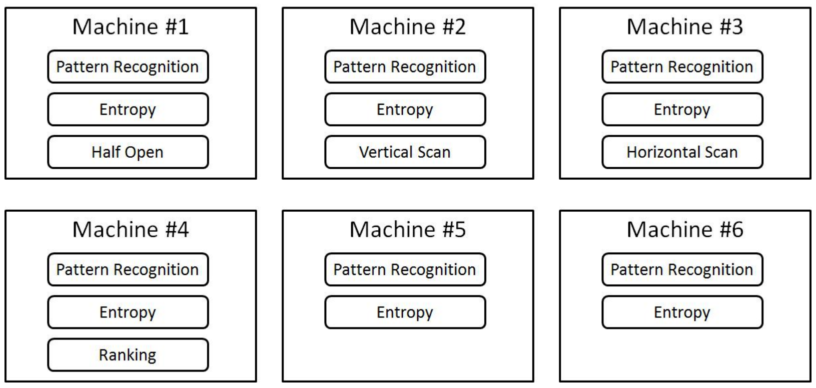


Figure 5: Possible deployment of the R-SYN topology in a six-machine cluster

nection attempts to open/closed ports (reachable/unreachable hosts) and outputs a $VS(x)$ ($HS(x)$) event when for a certain host x a fixed threshold has been exceeded, meaning that a vertical (horizontal) scan has been detected. It has to maintain a state about how many events have been received for each source host.

The *Entropy* bolt gets $failure(x, y, p)$ events as input. Each of these events means that a host x has issued a failed connection attempt (closed port or unreachable host) towards the port p of the destination host y. Upon the reception of such an event, it updates the entropy value for the host x according to the formulas presented in Section 4.2. It then sends to the Ranking bolt an event $EN(x)$ containing a new value.

The *Ranking* bolt is in charge of collecting all the events sent by the other bolts, computing the ranking for each suspicious host and triggering a scanner(x) event if the ranking is over a predefined threshold.

As for the deployment, we configured Storm so as to consider one thread for each task of each component. A possible deployment of the R-SYN topology in a cluster with six machines is shown in Figure 5. The topology has been configured to replicate both the Pattern Recognition and Entropy components in each available machine; the other bolts are not replicated at all. In this case, the replication is aimed at providing load balancing.

4.5. Experimental Evaluation

We carried out many experiments for assessing the effectiveness and the performances of the algorithms described so far using distinct settings. We first evaluate the effectiveness of the collaborative approach with R-SYN in an Esper-based SR, then we analyze how latency and throughput vary between two distinct implementations of R-SYN: one in Esper and the other one in Storm.

Testbed. For our evaluation we used a testbed consisting of a cluster of 10 Linux Virtual Machines (VMs), each of which equipped with 2GB of RAM and 40GB of disk space. The 10 VMs were hosted in a cluster of 4 quad core 2.8 Ghz dual processor physical machines equipped with 24GB of RAM. The physical machines are connected to a LAN of 10Gbit.

Traces. We used five intrusion traces. The first four were used in order to test the effectiveness of our algorithms in detecting malicious port scan activities whereas the latter has been used for computing the detection latency (see next paragraph). All traces include real traffic of networks that have been monitored. The traces are obtained from the ITOC research web site [38], the LBNL/ICSI Enterprise Tracing Project [39] and the MIT DARPA Intrusion detection project [40]. The content of the traces is described in Table 1. In each trace, the first TCP packet of a scanner always corresponded to the first TCP packet of a real port scan activity.

	trace1	trace2	trace3	trace4	trace5
size (MB)	3	5	85	156	287
source IPs	10	15	36	39	23
connections	1429	487	9749	413962	1126949
scanners	7	8	7	10	8
TCP packets	18108	849816	394496	1128729	3462827
3w-h packets	5060	13484	136086	883500	3393087
length (sec.)	5302	601	11760	81577	600
3w-h packet rate (p/s)	0.95	22.44	11.57	10.83	5655

Table 1: Content of the traces

4.5.1. R-SYN in Esper

We have carried out an experimental evaluation of R-SYN to assess two metrics; namely the *detection accuracy* in recognizing inter-domain stealthy SYN port scans and the *detection latency.*

The layout of the components on the 10 machine cluster described in Section 4.5 consisted of one VM dedicated to host the Esper CEP engine. Each of the remaining 9 VMs represented the resources made available by 9 simulated SR members participating in the Esper SR-ID. We emulated a large scale deployment environment so that all the VMs were connected with each other through an open source WAN emulator we have used for such a purpose. The emulator is called WANem [37] and allowed us to set specific physical link bandwidths in the communications among the VMs.

Detection Accuracy. In order to assess the accuracy of R-SYN , we partitioned the traces in order to simulate the presence of 9 SR members participating in the SR. The partitioning was based on the destination IP addresses in order to simulate that distinct SR members can only sniff TCP packets routed to diverse addresses. The resulting sub-traces were injected to the available Gateways of each member in order to observe what the algorithm was able to detect. To this end, we run a number of tests considering four accuracy metrics (following the assessment described in [41]): (i) TP (*True Positive*) which represents the number of suspicious hosts that are detected as scanners and are true scanners; (ii) FP (*False Positive*) which represents an error of the detection; that is, the number of honest source IP addresses considered as scanners; (iii) TN (*True Negative*) which represents the number of honest hosts that are not indeed detected as scanners; (iv) FN (*False Negative*) which represents a number of hosts that are real scanners that the systems do not detect. With these values we computed the *Detection Rate* *DR* and the *False Positive Rate* FPR as follows: $DR = TP/(TP + FN)$, and $FPR = FP/(FP + TN)$.

In all traces, with the exception of trace 4, we observed that the algorithm didn't introduce errors in the detection of port scanners; that is, the FPR was always 0% in our tests. In trace 4 of size 156MB, R-SYN exhibited a FPR equal to 3.4%; that is, R-SYN introduces 1 False Positive scanner.

Figure 6 shows the obtained results for the Detection Rate (DR). In this Figure, it emerges that the collaboration can be beneficial for sharpening the detection of port scanners. Augmenting the number of SR members (i.e., augmenting the volume of data to be correlated) leads to an increase of the detection rate as computed above.

Detection Latency. In the port scan attack scenario, the detection latency should be computed as the time elapsed between the first TCP packet of the port scan activity is sent by a certain IP address and the SR marks that IP address as scanner (i.e., when it includes the address in the blacklist). It is worth noting that we cannot know precisely which TCP packet should be considered the first of a port scan, since that depends on the true aims of who sends such packet. As already said, in our traces the first TCP packet of a scanner corresponds to the first TCP packet of a real port scan activity so that we can compute the detection latency for a certain IP address x as the time elapsed between the sending of the first TCP packet by x and the

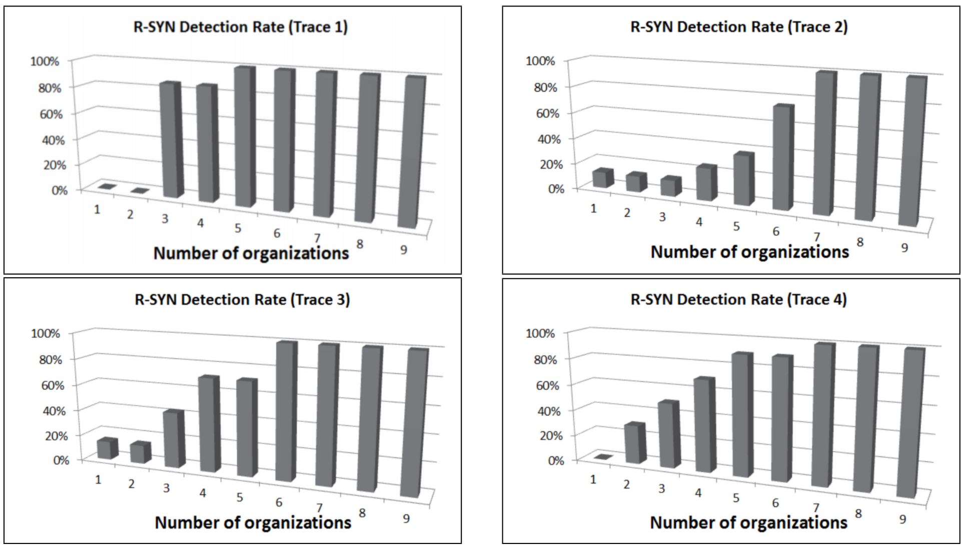


Figure 6: Port scan detection rate vs number of SR members for R-SYN algorithm. Each member contributes to the processing with a number of network packets that is on average 1/9 of the size of the trace.

detection of x as scanner.

In doing so, we need the timestamps of the packets. For such a purpose we developed a simple Java application named `TimerDumping` which (i) takes a trace as input; (ii) sends the packets contained in the trace (according to the original packet rate) to the Gateway using a simple pipe; and (iii) maintains the timestamp of the first packet sent by each source IP address in the trace.

We deployed an instance of `TimerDumping` on each VM hosting the Gateway component. Each `TimerDumping` produces a list of pairs $< ip\_address, ts >$, where ts is the timestamp of the first TCP packet sent by $ip\_address$. The timestamps are then used as beginning events for detection latency computation. Since there are more `TimerDumping` instances, pairs with the same IP address but different timestamps may exist. In those cases, we consider the oldest timestamp.

Timestamps are generated using local clocks of the hosts of the cluster. In order to ensure an acceptable degree of synchronization, we configured all the clustered machines to use the same NTP server which has been installed in a host located at the same LAN. The offset between local clocks is within 10 milliseconds which is accurate for our tests as latency measures are in the order of seconds.

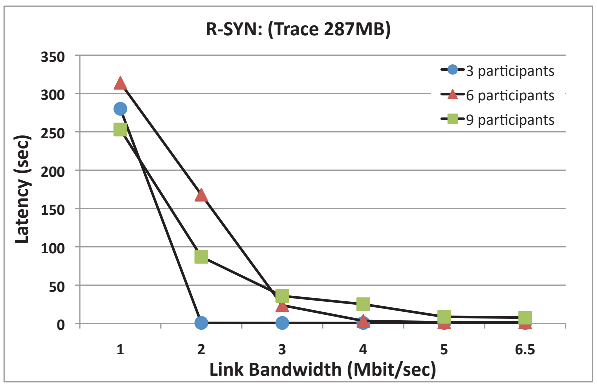


Figure 7: R-SYN detection latencies for different link bandwidths, in the presence of 3, 6, and 9 SR members.

For detection latency tests we used the trace of 287MB and changed the physical link bandwidths to the Esper in order to show how detection latency is affected by available bandwidth. Link bandwidth is controlled by the WANem emulator. We varied the physical link bandwidth using the WANem emulator with values ranging from 1 Mbit/s up to 6.5 Mbit/s. Figure 7 shows the average detection latency in seconds we obtained in different runs of the algorithm.

As illustrated in this Figure, for reasonable link bandwidths of a large scale deployment scenario (between 3 Mbit/s up to 6.5 Mbit/s) R-SYN algorithm shows acceptable detection latencies for the inter-domain port scan application (latencies vary between 0.6 to 35 seconds). In addition, results show that when the collaborative system is formed by a higher number of SR members (e.g., 9), detection latencies are better than those obtained with smaller SRs. This is principally caused by the larger amount of data available when the number of SR members increases: more data allow us to detect the scanners more quickly. In contrast, when 3 or 6 SR members are present, we need to wait more in order to achieve the final result of the computation.

4.5.2. R-SYN in Esper vs R-SYN in Storm

We carried out some experiments to compare the performances of R-SYN implemented in Esper against those obtained from its implementation in Storm. We setup two SRs with six members each (and six machines, one for each member): one SR with Esper as CEP engine and the other SR with Storm.

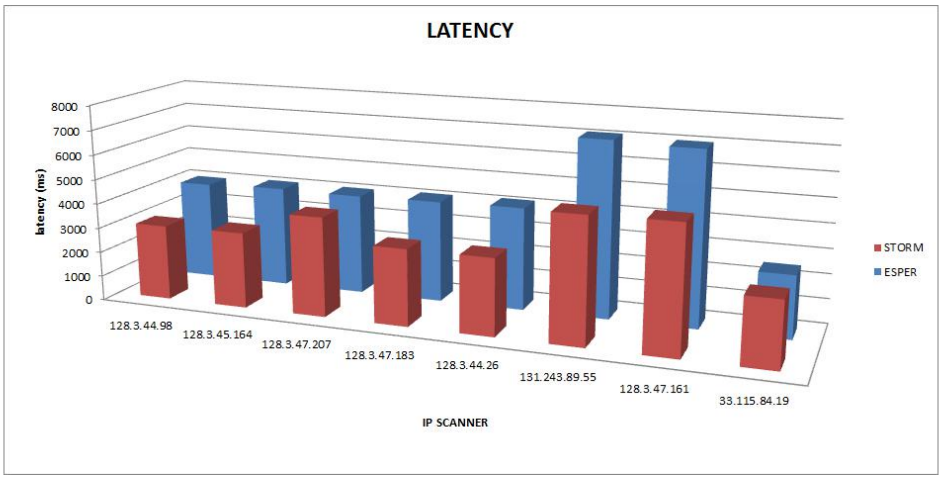


Figure 8: Detection latency comparison between Esper and Storm implementations of R-SYN algorithm

Detection Latency. We first evaluated the detection latency using trace2 (see Table 1); the results we collected show that Storm exhibits on average lower latencies compared to Esper (see Figure 8). Although the processing of a single event in Storm includes message passings between distinct physical machines due to its distributed setting, part of the computation can be performed in parallel, in particular the processing of the bolts *Half Open*, *Vertical Scan*, *Horizontal Scan* and *Entropy*. This explains why we can observe detection latencies lower than those provided by Esper.

Event Throughput. We then evaluated the throughput of the two engines with respect to the input event rate generated by the sources. Varying the rate of the packets that are sniffed by the edge components (i.e., the SR gateways in the Esper-based SR and the spouts in the Storm-based SR), we measured the number of events per second processed by the engine. For the Storm-based SR, we chose the *Entropy* bolt for our measurements since it is the most computation demanding node, and the overall throughput cannot be higher than that provided by it. For the Esper-based SR, we counted the number of updates per second performed by the listener in Esper in charge of computing the entropy.

In order to analyze the scaling capabilities of Storm, we setup distinct configurations of the topology, changing the number of replicas for the *Entropy* bolt. Figure 9 shows the results we obtained. We also included the throughput of the spouts to put an upper bound to the throughput achiev-

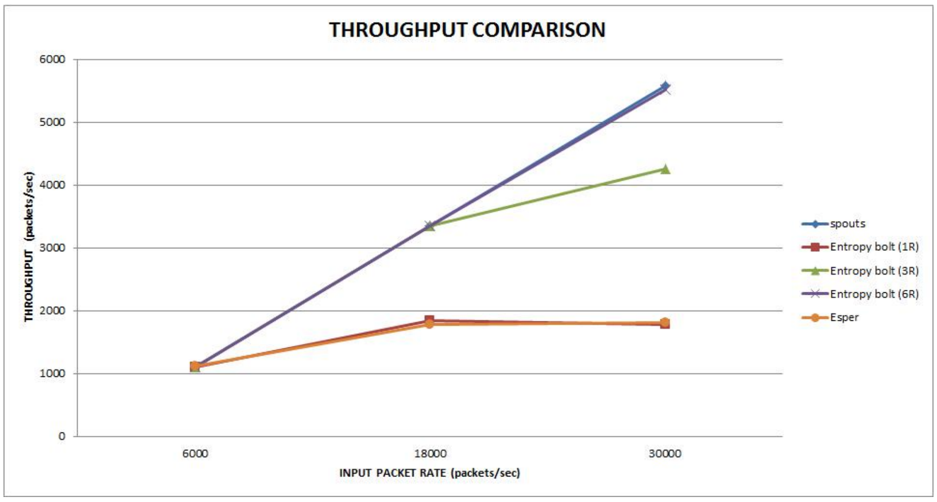


Figure 9: Throughput comparison between Esper and Storm implementations of R-SYN algorithm

able by the two engines. Comparing Esper with the configuration of Storm with a single replica of the Entropy bolt, it emerges that both engines are not able to sustain the input load as this increases, and the exhibited throughputs are almost equal. The configuration with three replicas can sustain higher input loads; however, it fails when the global input packet rate reaches 30 K packet/s. With six bolts, which means one Entropy bolt for each spout, Storm is able to sustain the load generated by the spouts.

These results highlight the ability of Storm to scale out on demand just by increasing the number of replicas of bottleneck bolts, and by also providing the required physical resources where deploying the replicas.

5. An Online Location Intelligence Semantic Room for Fraud Monitoring

In the Department of the Treasury of the Ministry of Economic and Finance (MEF), the V Directorate entitled "Financial crime prevention", and in particular the Central Office for Means of Payment Fraud (UCAMP) [42] is responsible for monitoring counterfeit Euros and preventing fraud committed through the use of payment means other than cash[2]. Within the

[2]In this section, only the following text has been agreed to be disclosed.

Community system set up to protect Euro from being counterfeited, introduced by Regulation 1338/2001, UCAMP serves as the Italian central Office for the collection and exchange of technical-statistical data regarding cases of falsifications detected throughout the national territory. On the basis of a strategic analysis of the information received by financial institutions, UCAMP is constantly able to evaluate the impact of the fraud phenomenon on the economic and financial system. In carrying out its functions, UCAMP performs complex data analysis and collaborative information sharing, acquiring data from banks and others financial institutions. The analysis aims at detecting specific types of frauds such as counterfeit euros and payment card frauds by correlating data that can be apparently fully uncorrelated with each other.

In order to show the high flexibility of the SR abstraction, we used it for enabling information sharing and processing between the earlier mentioned different financial institutions. In this section, we introduce the design and implementation of an SR we refer to as *Esper FM-SR*, we built in collaboration with UCAMP (MEF) so as to deploy an online location intelligence event-based platform for fraud monitoring. The SR exploits the Esper engine for collaboratively correlating fraud information coming from banks and others financial institutions. UCAMP is responsible for the processing of the SR; however, it needs to outsource this task to an application service provider that actually performs the correlation. The results produced by the SR are consumed by UCAMP in order to meet its institutional functions and by banks and other involved financial bodies for protecting themselves from fraudulent episodes (e.g., banks can plan to open a new agency in a geographical area with a low density of frauds). A simpler solution could have been used, which consists in a centralized database where participants share their own data and correlation can be performed; the choice to use a more complex architecture that includes a CEP engine was a specific requirement set by UCAMP: in perspective UCAMP plans to use such technology to accommodate for many others streams with very heterogeneous properties, and for other collaborative scenarios where an online processing can be necessary, as discussed at the end of this section.

The SR allowed us to assess: (i) the applicability of the SR abstraction to the real world of financial context; (ii) the ability of the platform to be easily integrated into "legacy" IT infrastructures of financial stakeholders, without requiring both updates to existing IT solutions and significant efforts; (iii) the possibility of collaboratively fusing data coming from different institutions; (iv) the possibility to instrument the SR gateway in order to anonymize sensitive data that are processed by the SR; and finally (v) the ability to localize and study fraud phenomena in different areas of the Italian territory.

5.1. Fraud monitoring

In order to exploit the SR abstraction capabilities for fraud monitoring purposes, we first carried out an assessment of the data UCAMP obtains from banks and other financial institutions, so as to evaluate which banks' data streams could be used to enable a collaborative correlation.

5.1.1. Data streams

We identified and used two types of data streams: a data stream type related to counterfeit euro notes, and a data stream type related to frauds carried out with electronic payment means. As for the former, data come from reports of banks that register such information as date, ABI and CAB of the bank, serial number and denomination of euros presented at a bank counter service and that appear as counterfeit banknotes. In some cases, the identity of the person who presented the counterfeit euros at the bank is also sent to UCAMP.

As for the latter, we distinguish between different types of data: *unauthorized POSs*, that are points of sale managed by merchants who carry out malicious trade activities, *disregarded credit card transactions* that concern all those transactions that are denied by the owner of the credit card, and *tampered ATMs.*

5.1.2. Types of data correlation

From an assessment of the earlier discussed data, it emerged that the types of correlations we could perform in the SR principally regarded the geographical location of the occurred frauds. Note that the information related to disregarded credit card transactions turned out to be not relevant for our purposes, as the reported data were related only to the city where the denied credit card transactions were carried out (and mostly those cities were located outside the geographical boundaries of Italy).

Owing to this observation, we focused our attention on the remaining three types of data streams (i.e., counterfeit euros, unauthorized POSs and tampered ATMs) and we designed three different correlation algorithms. In all the algorithms, we identified the precise address in the Italian territory where the fraud events happened.

The address was obtained using both ABI/CAB identifiers of bank branches, and the location address of the POS. Each address has been further translated into a point in latitude and longitude necessary for carrying out a spatial correlation among the fraud events. In doing so, we exploited geocoding services offered by Google and Yahoo so that to build a local database (see below) we periodically interrogate in the processing phase in order to execute the correlation algorithms. The correlation algorithms are summarized in the following.

GeoAggregation. The underlying principle of the GeoAggregation algorithm is to divide the Italian territory in small areas of equal size, roughly of 1.1 km^2 each. The areas are obtained by truncating the (latitude, longitude) coordinate. For instance, given a point (41.876883, 12.474309) we truncate the digits up to the second obtaining the SW point of an imaginary rectangle. Within each area, we keep track of the fraud events that are observed. To this end, we assign a weight to the events; the weight depends on the severity and certainty that the events are actually occurred in that area. Hence, in our current implementation of the GeoAggregation algorithm, we assign 8 as weight for tampered ATMs, 4 as weight for unauthorized points of sale and 3 to counterfeit euros. The low value we associated with the counterfeit euros is motivated by the fact that banknotes can be rapidly distributed and it is likely that the point in space in which they appear is not the actual location where they have been produced.

DEF: based on these weights, for a given geographical area x, we define $rank(x)$ as follows:

$$rank(x) = (T\_ATM(x) + UAuth\_POS(x) + C\_Euro(x))$$

The rank is used to identify areas, so-called "hot areas" of the Italian territory with a low, medium and high concentration of fraud episodes.

Entropy. The entropy algorithm draws us inspiration from the entropy computed in the previous SR for intrusion detection. In particular, we use this notion in order to compute how much the crime is "different" within each geographical area (even in this case, the areas are computed as earlier described). With the term "different" we mean that the frauds are related to the three different types of events we focused on, and are carried out by different people. For instance, if the same ATM is tampered twice in a row with the same method (e.g., installing a micro-camera to register the pin code of users) it is likely that the same people maliciously act on it.

DEF: for a given geographical area x, we define $E(x)$ as follows:

$$E(x) = -\frac{\sum_j p_{z_j} log(p_{z_j})}{log(N_z)}$$

where p_{z_j} is the frequency of each criminal event z_j, and N_z is the cardinality of the set of events happened in the area. The entropy assumes a value in the range $[0, 1]$ so that if all the fraud events occurred within an area are different between each other, the entropy of that area is close to 1; otherwise the entropy is close to 0.

Counterfeit Euros. Finally, we carried out a deep analysis of the distribution of counterfeit euros (see Section 5.2.4 for a user-level evaluation of such a distribution). The algorithm we developed evaluates the spreading on the Italian territory over a certain interval of time of counterfeit banknotes; for each note, it computes its cardinality; that is, the number of different banks in which the same serial number of the banknote has been reported. A high cardinality likely means that the counterfeit banknote was made in a large quantity.

In addition, we observed that some banknotes were of the same type and their serial numbers were similar with one another. This observation turned out to be particularly interesting for UCAMP(MEF) people in order to study the phenomenon. Thus, we evaluated the distribution of such kinds of banknotes, that are likely produced by the same counterfeiter, by constructing a similarity graph G(V,E). V is the set of vertices represented by the banknotes and $E : \{e :< v_1, v_2 >\in E| \, if \, and \, only \, if \, h(v_1, v_2) \leq t\}$. In other words, the similarity between banknotes is detected by computing the hamming distance (h) between two serial numbers. If that distance is $<=$ 2 (i.e., t) then we mark those banknotes as similar.

5.1.3. Privacy Requirements

Some of the data to be exchanged, for example the tampered ATMs, contain sensitive information that, if not protected during the processing, can reveal the identity of banks subject to frauds. Since UCAMP entrusts the processing to an application service provider, UCAMP required us to design a processing system capable of preserving the privacy of such sensitive information. In particular, the goal is to avoid simple linkability between an event injected into the SR and the bank that generated such event.

Since all the events are received by the application service provider, the latter should be prevented from inferring which bank produced an event by simply observing the fields of the event itself. To meet this requirement, in our SR we use two strategies: we remove sensitive data fields that do not contribute significantly to the correlation logic, and we modify the content of necessary data fields in such a way to make difficult identifying the bank that originated the data, performing at the same time the correlation.

A weakness of this approach is that the application service provider could discover the bank that originated an event just by tracing the host that sent the event itself, since that host is likely to be hosted exactly by the bank that generated the event. For this reason, another key point is preventing the application service provider, and any other SR member in general, from linking banks to events by analyzing the SR traffic. This can be achieved by employing some combination of the techniques described in Section 3.2. How these privacy requirements are enforced in practice in the FM-SR is described in Section 5.2.2.

5.2. Esper FM-SR

As in case of the previously described SR, the Esper FM-SR consists of a number of pre-processing elements, a processing engine and an advanced Web-based Graphical User Interface. The pre-processing elements are the set of SR Gateways deployed at each SR member site. In the Esper FM-SR, the SR Members are banks and other financial institutions that send data regarding frauds to UCAMP.

The processing element is represented by the Esper engine managed by an application service provider to which UCAMP outsources the data correlation task. The results of the processing are rendered to SR Members by means of an advanced Web-based interface that visualizes geolocation information. The individual components of the Esper FM-SR are illustrated in Figure 10 and described in detail below.

5.2.1. SR Gateway

The SR Gateway takes as input the data provided by the SR members and performs a number of operations: it converts the data into events (the events are represented by POJO objects) for the use by the Esper engine; it reads from a Mysql database local to each SR Gateway (local to each SR member) the information for the conversion of ABI/CAB of banks into addresses, and their related latitudes and longitudes. For this purpose, we use Google and Yahoo cloud services to construct such a database (see Figure 10). This information is then included into the POJO objects.

5.2.2. Enforcing Privacy Requirements

Two different steps are executed in order to anonymize the data. In the first step, the data fields of the events that can uniquely identify the originating bank are obfuscated. For instance, ABI/CAB are filtered out and the latitude and longitude are modified by adding a small noise to both, like in the randomization method [61, 62]. This noise allows us to maintain the

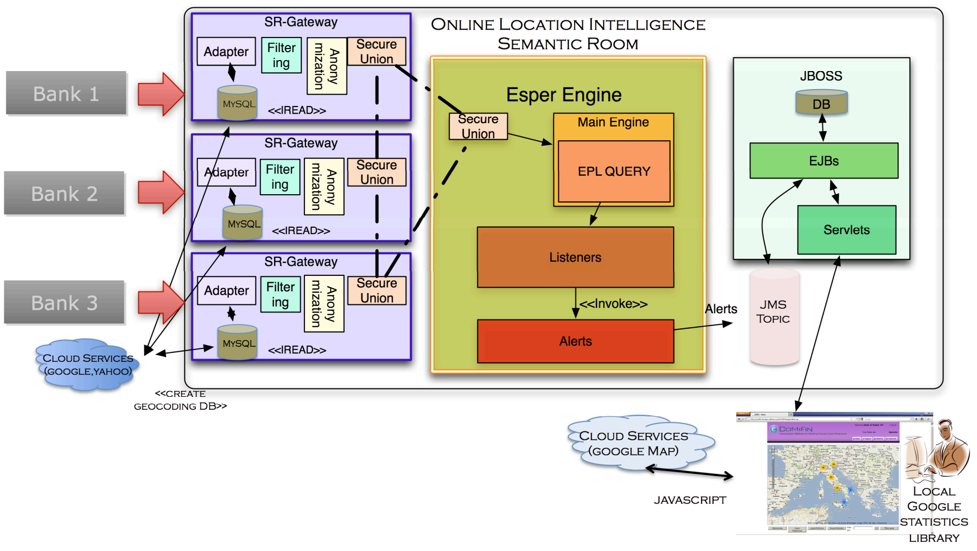


Figure 10: The online location intelligence SR for fraud monitoring

event in its geographical area, avoiding at the same time that the latitude and longitude are the same of the bank that produced the data. Note that without this noise, it would be possible to discover, during the processing, the identity of the financial institution that produced the event. Nevertheless, perturbing coordinates by itself do not guarantee the anonymity of the bank producing the information when the number of banks in a rectangular area is very low. A similar issue has been debated in [65]. The authors proposed to distinguish and tune the anonymization requirements for distinct items. Thus we are working to a solution that adapts the dimension of the rectangular areas so that each area includes, at least, a predefined number of banks.

In the second step, the anonymized events are to be sent to Esper. However, sending them directly to the engine would leak the identity of the banks, thus violating UCAMP's privacy requirements. Therefore, we use a communication proxy located in the SR Gateway capable of anonymizing the communication between banks and Esper. In particular, all the SR Gateways of the SR members periodically execute a secure-union [45] of events; the secure-union is a well known primitive in multi-party computation that allows many participants $\{p_1, p_2, ..., p_n\}$ to compute the union $U : \{e_{p_1}, e_{p_2}, e_{p_3}, ..., e_{p_n}\}$ of their individual input (the input of p_j is e_{p_j}) in such a way that for a specific participant p_x it is not possible to pin-point an element $e_{p_x} \in U$. The secure union primitive has been used by many complex algorithms for Anonymous Intrusion Detection [51] and Oblivious Assignment of m-Slots [46].

In our case, the inputs are the events generated by each SR member, and the resulting set is the set of events that is sent to Esper. We implemented this primitive using a cryptographic scheme that is re-encryptable (such as the scheme described in [47]).

In the scheme, the SR members act on the top of a logic ring; a circulating token is sent by a SR member leader and each SR member adds its encrypted element to the token and re-encrypts all the remaining elements. With this scheme, Esper receives periodically a set of events : $\{e_{p_1}, e_{p_2}, e_{p_3}, ..., e_{p_n}\}$ from the banks where each even e_{p_j} is anonymous in the sense that is not possible to know its original sender. The only role of the leader is to start the union; it does not obtain any information respect to other parties. The election of the leader can be fixed, or in a synchronous faulty environment, can be demanded to standard techniques. In our implementation we used a rotating leader, with a new leader for each round; this approach has the advantage to balance the load in the system: a process has to re-encrypt all the elements added by the processes between it and the round leader.

5.2.3. Esper FM-SR Processing

Listing 2: EPL query for detecting similarity of counterfeit banknotes

```
select count(*) as tuple,
  e1.data as D1, e1.geopoint as G1,
  e1.SERIAL_NUMBER1 as N1, e2.SERIAL_NUMBER2 [....]
from pattern [
  every e1 = bankunique ->
  every e2 = bankunique(Similarity(......) > 0.3 and
  e1.SERIAL_NUMBER1 != e2.SERIAL_NUMBER2)
  where timer:withinmax(30d, 5000) ]
```

We used the Esper CEP engine as in the case of the previous SR for correlating the data coming from the SR Gateways. The engine receives, under the form of POJOs, the data from the banks that are anonymized by means of the Secure Union primitive. For demonstration purposes we used just the data registered during 2010.

The engine is instrumented by a number of EPL queries that implement the types of correlations previously introduced. For the sake of brevity we do not include all the queries we have implemented. However, as example, in Listing 2 we report the query we used to monitor whether for a specific banknote (in Listing 2 the “bankunique” stream) there have been similar banknotes in different places of the Italian territory with similar serial number. *Similarity* is a user defined function developed in order to find out the hamming similarity of two banknotes within thirty days (in Listing 2 the “where” clause)

5.2.4. Presentation Layer: Web-based Graphical User Interface

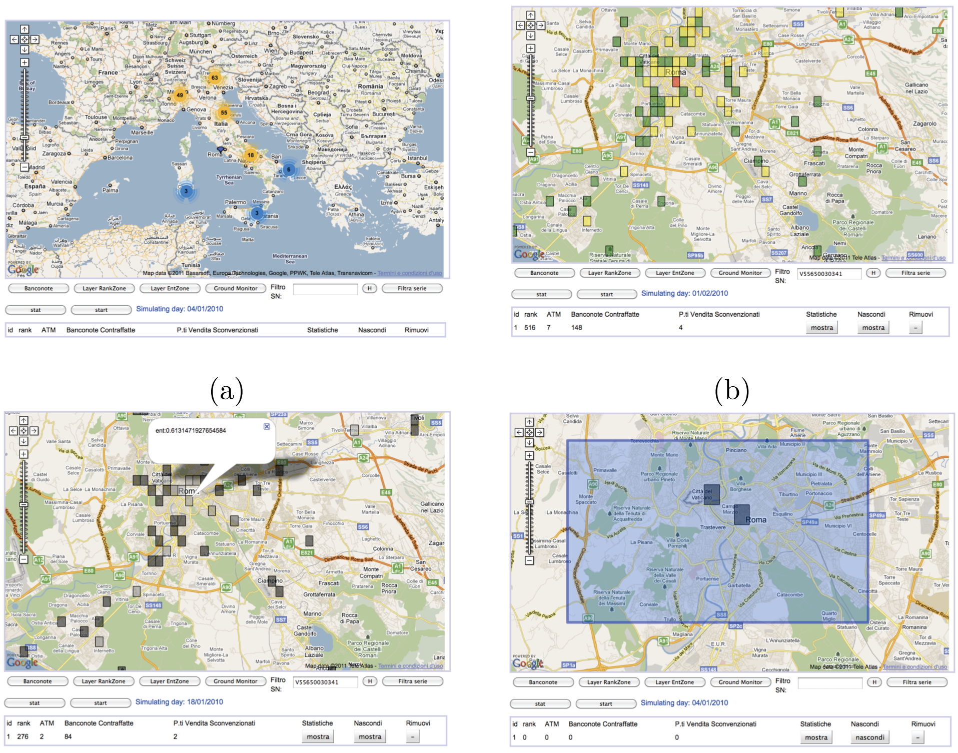
The presentation layer of the Esper FM-SR is represented by a web-based graphical user interface[3] developed in Javascript with the AJAX technology. The web-based GUI calls a number of servlets we implemented that in turn interface the EJB layer in order to obtain the alerts generated by Esper (see Figure 10). These alerts are then visualized on a map. For showing alerts on a map, we used such Google cloud services as the Javascript Maps APIs, without compromising possible privacy requirements required by UCAMP for the sensitive data since the Google services receive only information related to the map size and locations. The web-based GUI (Figure 11) consists of four principal layers; namely the Banknote, GeoAggregation, Entropy, and Ground Monitor layers, each of them briefly described in the following.

The Banknote Layer shows the spreading of the counterfeit euros in the territory at real time (Figure 11 (a), in the screenshot this layer is called “banconote”). The clusters include the number of banknotes and are colored accordingly (red for high concentration, yellow for medium and blue for low). The clusters are dynamically modified at run time as soon as new data are processed by Esper.

The GeoAggregation layer is responsible for visualizing so called “hot areas”; that is, areas that show a high (red color), medium (yellow color) or low (green color) concentration of fraud events (Figure 11 (b), in the screenshot, this layer is called “Layer RankZone”).

The Entropy layer allows a SR member to visualize the fraud entropy in a zone (Figure 11 (c), in the screenshot, this layer is called “Layer EntZone”). Specifically, rectangles are blank if the entropy is low and hence there are similar types of frauds in that area, grey if the entropy is medium, and white if the entropy is close to 1; that is, if there exist many different types of frauds in that area.

[3] In the screenshots of Figure 11, some parts are in Italian as the final prototype was meant to be shown to Italian law enforcement agencies.



(a)

id	rank	ATM	Banconote Contraffatte	P.ti Vendita Sconvenzionati	Statistiche	Nascondi	Rimuovi

(b)

id	rank	ATM	Banconote Contraffatte	P.ti Vendita Sconvenzionati	Statistiche	Nascondi	Rimuovi
1	516	7	148	4	mostra	mostra	-

(c)

id	rank	ATM	Banconote Contraffatte	P.ti Vendita Sconvenzionati	Statistiche	Nascondi	Rimuovi
1	276	2	84	2	mostra	mostra	-

(d)

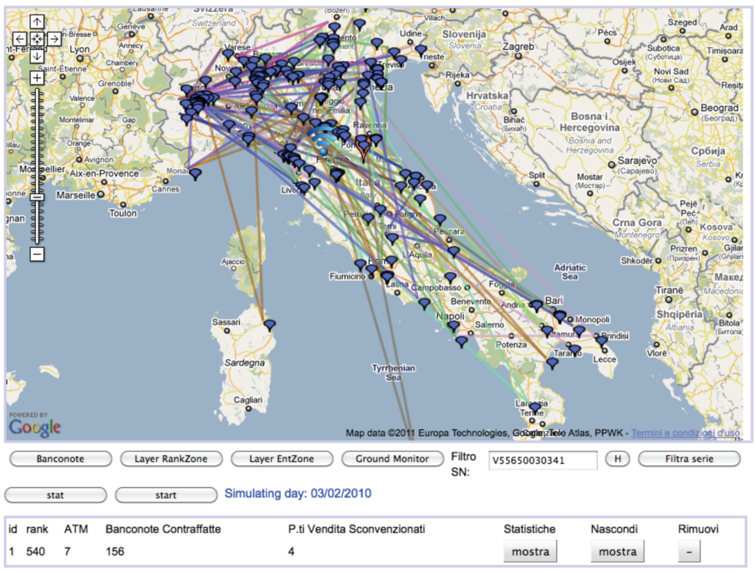
id	rank	ATM	Banconote Contraffatte	P.ti Vendita Sconvenzionati	Statistiche	Nascondi	Rimuovi
1	0	0	0	0	mostra	nascondi	-

Figure 11: Web-based Graphical User Interface: (a) Banknote Layer; (b) GeoAggregation Layer; (c) Entropy Layer; (d) Ground Monitor Layer

The Ground Monitor Layer (see Figure 11 (d)) provides a sort of *on demand fraud monitoring*. In other words, a SR member can select a specific geographical area of interest by drawing a rectangle on the map, and monitor all the results of the previous layers in that rectangle, only.

Italy has unfortunately the sad primacy of being the European country with the largest production of counterfeit euros. These banknotes are distributed not only on the Italian territory but also in all European Union. A deep analysis of this phenomenon can then be crucial in the field of fraud monitoring and detection.

Using the Esper FM-SR prototype we were able to evaluate the distribution in space and time of the counterfeit banknotes. Figure 12 illustrates how counterfeit banknotes events are connected with each other: events occurred on the same day are connected with dots of the same color.



Simulating day: 03/02/2010

id	rank	ATM	Banconote Contraffatte	P.ti Vendita Sconvenzionati	Statistiche	Nascondi	Rimuovi
1	540	7	156	4	mostra	mostra	-

Figure 12: Connection among counterfeit banknotes events

In conclusion, the Esper FM-SR allowed us to build a set of advanced tools to support intelligence operations carried out by UCAMP. These tools can be also conveniently exploited by banks and financial institutions so as to monitor geographical areas with a high density of frauds, and by police institutions and officers of the "Guardia di Finanza" in order to (i) identify the geographic distribution of frauds over the Italian territory and within specific areas of interest selected on demand by the users handling the tools, (ii) monitoring the spreading of counterfeit banknotes and of suspicious banknotes' serial numbers that can be potentially produced by the same counterfeiters, and, finally, (iii) analyze the trend of the fraud phenomenon.

6. Concluding Remarks

Enabling information sharing is one of the most important option that a community has at disposal to defend itself from cyber threats and frauds. Even though information sharing bodies exist, they mainly operate in an off line fashion by mining traces left by attackers and then disseminating the extracted information to the information sharing participants. There is still a gap in providing collaborative platform where participants can inject information that is then correlated in a timely manner. Additionally, the platform should be able to realize an environment where also competing organizations can safely contribute their data.

The paper has focused on these issues by leveraging a programming abstraction named Semantic Room that, once instantiated, acts as an enabler of information sharing. For event correlation purposes, the SR embodies two main components: a SR Gateway that gathers information from participants' information systems, and a CEP engine that correlates the information. It is important to remark that in order to reduce the number of events to be correlated, the SR defines an objective (e.g., portscan detection, botnet detection etc) to achieve. Thus, the SR Gateway and event processing engine are properly instrumented to meet that objective. This makes the overall computation tractable.

In order to demonstrate the effectiveness and flexibility of the Semantic Room programming abstraction, we described the design, implementation and evaluation of two SRs: one for port scan activities detection and one for fraud monitoring. The first SR has been developed using a centralized processing engine named Esper, and a distributed engine called Storm that allowed us to show the SR ability to cope with possible scalability issues. In either cases, the SRs have been instrumented with real network traces including port scan operations. A comparison between the two have been introduced and discussed. The second SR has been developed using Esper, provided with privacy-preserving mechanisms, and instrumented with real data coming from banks and other financial institutions related to fraudulent episodes registered in Italy in 2010.

In the context of port scan detection, accuracy of the detection has been very high and detection delays have been quite small. In the context of fraud monitoring, instantiating an SR has been instrumental to find new types of investigations for the police and central offices like UCAMP, and to timely react to fraud propagation.

There are still open issues we are currently considering in new releases of SRs. One important problem concerns the trustworthiness of the SR platform and consists in providing mechanisms to continuously assess participants' behavior and the quality of the data they provide. This evaluation can then be used in order to discriminate the participation of some SR members in the computation. Another relevant point is the improvement of the privacy guarantees provided by the SR. The privacy preserving mechanisms used in the FM-SR allow us to filter out sensitive data and mask the identity of data provider. An interesting step ahead would be the integration of techniques to also encode the data to process, so that no participant can trace back such data, and still the computation can be executed. Finally, another interesting direction we are exploring is the development of SRs capable of facing other cyber threats such as Distributed Denial of Service and Man in the Middle.

7. Acknowledgements

The work has been carried out first in the context of the European project CoMiFin and then within the Italian PRIN Project TENACE. We would like to thank all the CoMiFin and TENACE partners. Special thanks to Luca Nicoletti, Andrea Baghini (Consip), Francesco Castanó (MEF) and UCAMP people that supported us in the design and development of the Semantic Room for Fraud monitoring. Authors are indebted with the Subject Area Editor and the anonymous reviewers for their comments and suggestions that greatly improved content and presentation of the paper.

References

[1] Global Fraud Report - Annual Edition 2011-2012, Kroll, `http://www.krollconsulting.com/fraud-report/2011-12/press-only/`, 2011.

[2] In the Crossfire: Critical Infrastructure in the Age of Cyber War, `http://www.mcafee.com/us/resources/reports/rp-in-crossfire-critical-infrastructure-cyber-war.pdf`, 2010.

[3] Information Sharing Strategy - National Intelligence, `http://www.dni.gov/reports/IC_Information_Sharing_Strategy.pdf`, 2008.

[4] NATO Network Enabled Capability, `http://nnec.act.nato.int/`, 2011.

[5] H. F. Lipson, Tracking and Tracing Cyber-Attacks: Technical Challenges and Global Policy, University of Pennsylvania Law Review, Vol. 154, p. 477, 2006.

[6] National Australia Bank it by DDoS attack, `http://www.zdnet.com.au/news/security/soa/National-Australia-Bank-hit-by-DDoS-attack/0,130061744,339271790,00.htm`, 2009.

[7] Update: Credit card firm hit by DDoS attack, `http://www.computerworld.com/securitytopics/security/story/0,10801,96099,00.html`, 2009.

[8] FBI investigates 9 Million ATM scam, `http://www.myfoxny.com/dpp/news/090202\_FBI\_Investigates\_9\_Million\_ATM\_Scam`, 2010.

[9] Financial Services - Information Sharing and Analysis Center, `http://www.fsisac.com/`, 2011.

[10] European anti fraud office, `http://ec.europa.eu/anti_fraud/index_it.html`, 2011.

[11] Y. Xie, V. Sekar, M. K. Reiter, H. Zhang, Forensic analysis for epidemic attacks in federated networks, Proceedings of the 14th International Conference on Network Protocols, pp. 43–53, IEEE, 2006.

[12] G. Zhang, M. Parashar, Cooperative detection and protection against network attacks using decentralized information sharing, Cluster Computing, Vol. 13, pp. 67–86, Kluwer Academic Publishers, 2010.

[13] Where Complex Event Processing meets Open Source: Esper and NEsper, `http://esper.codehaus.org/`, 2009.

[14] A. Adi, O. Etzion, Amit - the situation manager, The VLDB Journal, Vol. 13, pp. 177–203, Springer-Verlag, 2004.

[15] L. Amini, N. Jain, A. Sehgal, J. Silber, O. Verscheure, Adaptive control of extreme-scale stream processing systems, Proceedings of the 26th International Conference on Distributed Computing Systems, p. 71, IEEE, 2006.

[16] X. J. Zhang, H. Andrade, B. Gedik, R. King, J. Morar, S. Nathan, Y. Park, R. Pavuluri, E. Pring, R. Schnier, P. Selo, M. Spicer, V. Uhlig, C. Venkatramani, Implementing a high-volume, low-latency market data processing system on commodity hardware using ibm middleware, Proceedings of the 2nd Workshop on High Performance Computational Finance, pp. 7:1–7:8, ACM, 2009,

[17] M. Akdere, U. Çetintemel, N. Tatbul, Plan-based complex event detection across distributed sources, Proceedings of the VLDB Endowment, Vol. 1, pp. 66–77, VLDB Endowment, 2008.

[18] P. R. Pietzuch, Hermes: A Scalable Event-Based Middleware, Ph.D. Thesis, University of Cambridge, 2004.

[19] JBoss Drools Fusion, `http://www.jboss.org/drools/drools-fusion.html`, 2010.

[20] Y. Huang, N. feamser, A. L. and J. J. Xu, Diagnosing network disruptions with network-wide analysis, Proceedings of the International Conference on Measurement and Modeling of Computer Systems, pp. 61–72, ACM, 2007.

[21] DShield: Cooperative Network Security Community - Internet Security, `http://www.dshield.org/indexd.html/`, 2009.

[22] Complete Snort-based IDS Architecture, `http://cybervlad.net/ids/index.html`, 2002.

[23] C.Kreibich, R. Sommer, Policy-controlled Event Management for Distributed Intrusion Detection, Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops, pp. 385–391, IEEE, 2005.

[24] M. E. Locasto, J. J. Parekh, A. D. Keromytis, S. J. Stolfo, Towards collaborative security and p2p intrusion detection, Proceedings of the Information Assurance Workshop, pp. 30–36, IEEE, 2005.

[25] R. Baldoni, G. Lodi, G. Chockler, E. Dekel, B. P. Mulcahy, G. Martufi, A contract-based Event Driven Model for Collaborative Security in Financial Information Systems, Proceedings of the 12th International Conference on Enterprise Information Systems, Vol. 4, pp. 147–152, SciTePress, 2010.

[26] C. V. Zhou, S. Karunasekera, C. Leckie, A peer-to-peer collaborative intrusion detection system, Proceedings of the 13th International Conference on Networks, Vol. 1, pp. 118–123, IEEE, 2005.

[27] N. Verma, F. Trousset, P. Poncelet, F. Masseglia, Intrusion Detections in Collaborative Organizations by Preserving Privacy, Advances in Knowledge Discovery and Management, pp. 235-247, Springer, 2009.

[28] C. V. Zhou, C. Leckie, S. Karunasekera, A survey of coordinated attacks and collaborative intrusion detection, Computers & Security, Vol. 29, pp. 124–140, Elsevier, 2010.

[29] D. Jeffrey, S. Ghemawat, MapReduce: simplified data processing on large clusters, Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation, Vol.6, pp. 137–149, USENIX Association, 2004.

[30] D. Lamanna, J. Skene, W. Emmerich, SLAng: A Language for Defining Service Level Agreements, Proceedings of the The 9th IEEE Workshop on Future Trends of Distributed Computing Systems, p. 100, IEEE, 2003.

[31] J. Jung, V. Paxson, A. W. Berger, H. Balakrishnan, Fast portscan detection using sequential hypothesis testing, In Proceedings of the Symposium on Security and Privacy, pp. 211-225, IEEE, 2004.

[32] H. Singh, R. Chun, Distributed Port Scan, Handbook of Information and Communication Security - Part B, pp. 221–234, Springer, 2010.

[33] W. G. Hai Zhang, Xuyang Zhu, Tcp portscan detection based on single packet flows and entropy, Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, pp. 1056–1060, ACM, 2009.

[34] Snort: an open source network intrusion prevention and detection system (IDS/IPS), `http://www.snort.org/`, 2010.

[35] L. Aniello, G. Lodi, R. Baldoni, Inter-Domain Stealthy Port Scan Detection through Complex Event Processing, Proceedings of the 13th European Workshop on Dependable Computing, pp. 67–72, ACM, 2011.

[36] L. Aniello, G. A. Di Luna, R. Baldoni, A Collaborative Event Processing System for Protection of Critical Infrastructures From Cyber Attacks, Proceedings of the 30th Conference on System Safety, Reliability and Security, pp. 310–323, Springer-Verlag, 2011.

[37] WANem The Wide Area Network emulator, `http://wanem.sourceforge.net/`, 2011.

[38] ITOC Research: CDX Datasets, `http://www.itoc.usma.edu/research/dataset/index.html`, 2010.

[39] LBNL/ICSI Enterprise Tracing Project, `http://www.icir.org/enterprise-tracing/`, 2010.

[40] 2000 DARPA Intrusion Detection Scenario Specific Data Sets, `http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/2000data.html`, 2000.

[41] C. Zhou, S. Karunasekera, C. Leckie, Evaluation of a Decentralized Architecture for Large Scale Collaborative Intrusion Detection, Proceedings of the 10th International Symposium on Integrated Network Management, pp. 80–89, IEEE, 2007.

[42] Dipartimento del tesoro - anti fraud system, `http://www.dt.tesoro.it/en/antifrode_mezzi_pagamento/`, 2011.

[43] Bugra Gedik, Henrique Andrade, Kun-Lung Wu, Philip S. Yu, Myungcheol Doo, SPADE: the System S declarative stream processing engine, Proceedings of the International Conference on Management of Data, pp. 1123–1134, ACM, 2008.

[44] Storm, distributed and fault-tolerant realtime computation, `http://storm-project.net/`, 2011.

[45] L. Kissner, D. Song, Privacy-preserving set operations, Proceedings of the 25th Annual International Cryptology Conference, pp. 241–257, Springer, 2005.

[46] G. Ateniese, R. Baldoni, S. Bonomi, G. Di Luna, Oblivious assignment with m-slot, Proceedings of the 14th International Symposium on Stabilization, Safety, and Security of Distributed Systems, pp. 187–201, Springer-Verlag, 2012.

[47] P. Golle, M. Jakobsson, A. Juels, P. Syverson, Universal re-encryption for mixnets, Proceedings of the RSA Conference, Cryptographers track, pp. 163–178, Springer-Verlag, 2002.

[48] B. Applebaum, H. Ringberg, M. J. Freedman, M. Caesar, J. Rexford, Collaborative, privacy-preserving data aggregation at scale, Proceedings of the 10th International Conference on Privacy Enhancing Technologies, pp. 56–74, Springer-Verlag, 2010.

[49] D. Bogdanov, S. Laur, J. Willemson, Sharemind: A framework for fast privacy-preserving computations, Proceedings of the 13th European Symposium on Research in Computer Security, pp. 192–206, Springer, 2008.

[50] N. Nisan, B. Pinkas, Y. Sella, Fairplay: A secure two-party computation system, Proceedings of the 13th conference on USENIX Security Symposium, Vol. 13 p. 20, ACM, 2004.

[51] H. Kargupta, Multi-agent, distributed, privacy-preserving data management and data mining, US Patent 0017870 A1, 2008.

[52] N. Avourdiadis, A. Blyth, Data Unification and Data Fusion of Intrusion Detection Logs in a Network Centric Environment, Proceedings of the 4th European Conference on Information Warfare and Security, pp. 9–20, 2005.

[53] R. Baldoni, G. A. Di Luna L. Querzoni, Collaborative Detection of Coordinated Port Scans, Proceedings of the 14th International Conference on Distributed Computing and Networking, pp. 102–117, Springer, 2013.

[54] L. Aniello, R. Baldoni, L. Querzoni, Adaptive Online Scheduling in Storm, Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems, pp. 207–218, ACM, 2013.

[55] S. R. M. Oliveira, O. R. Zaiane, Y. Saygin, Secure Association Rule Sharing, Proceedings of the 8th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, pp. 74–85, Springer Berlin Heidelberg, 2004.

[56] Y. Saygin, V. S. Verykios, C. Clifton, Using unknowns to prevent discovery of association rules, SIGMOD Record, pp. 45–54, ACM, 2001.

[57] L. Chang, I. S. Moskowitz, Parsimonious downgrading and decision trees applied to the inference problem, Proceedings of the Workshop on New Security Paradigms, Vol. 30, pp. 82–89, ACM, 1998.

[58] I. S. Moskowitz, L. Chang, A decision theoretical based system for information downgrading, Proceedings of the 5th Joint Conference on Information Sciences, 2000.

[59] D. Dobkin, A. K. Jones, R. J. Lipton, Secure databases: protection against user influence, Transactions on Database Systems, pp. 97–106, ACM, 1979.

[60] K. Kenthapadi, N. Mishra, K. Nissim, Simulatable auditing, Proceedings of the 24th SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 118–127, ACM, 2005.

[61] R. Agrawal, R. Srikant, Privacy-preserving data mining, Proceedings of the SIGMOD International Conference on Management of Data, pp. 439–450, ACM, 2000.

[62] D. Dakshi, C. C. Aggarwal, On the design and quantification of privacy preserving data mining algorithms, Proceedings of the 20th SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 247–255, ACM, 2001.

[63] P. Samarati, Protecting Respondents' Identities in Microdata Release, Transactions on Knowledge and Data Engineering, pp. 1010–1027, IEEE, 2001.

[64] A. Machanavajjhala, D. Kifer,J. Gehrke, M. Venkitasubramaniam, L-diversity: Privacy beyond k-anonymity, Transactions on Knowledge Discovery from Data, p. 24, ACM, 2007.

[65] C. C. Aggarwal, P. S. Yu, On Variable Constraints in Privacy Preserving Data Mining, Proceedings of the 5th International Conference on Data Mining, pp. 115-125, SIAM, 2005.

[66] L. Aniello, R. Baldoni, G. Chockler, G. Laventman, G. Lodi, Y. Vigfusson, Agilis: An Internet-Scale Distributed Event Processing System for Collaborative Detection of Cyber Attacks, MidLab Technical Report 04/2011, 2011.

[67] R. Baldoni, G. Chockler, Collaborative Financial Infrastructure Protection: Tools, Abstractions, and Middleware, Springer, 2012.