Closing the loop of SIEM analysis to Secure Critical Infrastructures

(Article begins on next page)

19 April 2024

Closing the loop of SIEM analysis to Secure Critical Infrastructures

Alessia Garofalo*, Cesario Di Sarno*, Ilaria Matteucci†, Marco Vallini‡, Valerio Formicola*

*University of Naples "Parthenope", Department of Engineering, Naples, Italy
email: {alessia.garofalo, cesario.disarno, valerio.formicola}@uniparthenope.it
†IIT-CNR, Pisa, Italy, email: ilaria.matteucci@iit.cnr.it
‡Politecnico di Torino, Dip. di Automatica ed Informatica, Torino, Italy, email: marco.vallini@polito.it

Abstract—Critical Infrastructure Protection is one of the main challenges of last years. Security Information and Event Management (SIEM) systems are widely used for coping with this challenge. However, they currently present several limitations that have to be overcome. In this paper we propose an enhanced SIEM system in which we have introduced novel components to i) enable multiple layer data analysis; ii) resolve conflicts among security policies, and discover unauthorized data paths in such a way to be able to reconfigure network devices. Furthermore, the system is enriched by a Resilient Event Storage that ensures integrity and unforgeability of events stored.

Keywords—*Security Information and Event Management, Decision Support System, Hydroelectric Dam.*

I. INTRODUCTION

Department of Homeland Security (DHS) defines a Critical Infrastructure in the following way: *"Critical infrastructure are the assets, systems, and networks, whether physical or virtual, so vital to the United States that their incapacitation or destruction would have a debilitating effect on security, national economic security, national public health or safety, or any combination thereof"* [1]. Therefore, the protection of these infrastructures must be carefully considered to avoid disasters. For this purpose, DHS has identified sixteen different Critical Infrastructures that need to be monitored and protected [2]. The study performed by "Industrial Control Systems Cyber Emergency Response Team" (ICS-CERT) [3] highlighted that energy sectors - dams included - are the most attractive targets for cyber-attacks. Recently, the U.S. intelligence agency [4] traced back a cyber intrusion performed by China government into a database containing sensitive information of the USA government. Specifically, the database stored vulnerabilities of major dams in the United States that can be exploited to perform a future cyber attack against the US electrical power grid.

Nowadays, as described in McAfee's report [5], Security Information and Event Management (SIEM) systems are widely used to perform real-time monitoring and control of a Critical Infrastructure. SIEM [6] solutions are a combination of the formerly heterogeneous product categories of Security Information Management (SIM) and Security Event Management (SEM). In particular, SEM systems are focused on the aggregation of data into a manageable amount of information

with the help of which security incidents can be dealt with immediately, while SIM primarily focuses on the analysis of historical data in order to improve the long term effectiveness and efficiency of information security infrastructures [7]. SIEM technology aggregates event data produced by security devices, network infrastructures, systems and applications. The primary data source is log data, but SIEM technology can also process other forms of data. Event data is combined with contextual information about users, assets, threats and vulnerabilities. The data is normalized, so that events and contextual information from heterogeneous sources can be correlated and analyzed for specific purposes, such as network security event monitoring, user activity monitoring and compliance reporting. SIEM technology provides real-time security monitoring, historical analysis and other support for incident investigation and compliance reporting.

A weakness of SIEM systems is that they lack several features for Critical Infrastructure protection. In particular, four limits have been identified: 1) SIEMs processing capabilities include collection, aggregation and cross-correlation of heterogeneous sources, typically characterized by different syntax and semantics. However, they cannot process multiple layer data and take into account business process view, service view and physical domain view at the same time. Also, SIEM collectors cannot process data in proximity of the monitored domain in order to limit the amount of information disclosed out of the collection boundaries, *i.e.,* typically the part under legislative control of a company. 2) Gartner report [8] highlighted as the SIEM lacks context policies that are needed to identify exceptions. Critical Infrastructure monitoring requires instead that many context policies are defined in order to avoid misbehaviour. Also the need for several policies raises additional issues: what happens if some policies generate a conflict? How is it possible to take a decision when two policies are in conflict? 3) Critical Infrastructure monitoring is performed by deploying communication networks that enable the exchange of information between the monitored facilities and the control systems. In order to deny specific connections from external networks towards the internal ones, security policies pose strong limitations to data flows. For instance, an operation of sensor firmware re-writing can only be done from specific hosts in a permitted LAN, where privileged accounts exist and limited access to domain expert profiles is allowed. SIEM systems lack a methodology to identify and control all possible data paths existing in the monitored infrastructures. 4) SIEMs generate alarms when attack signatures are detected. Alarms are stored along with related events in storage media,

1

e.g., databases. Information contained in the alarms can be used for forensic purposes in order to discover the attacker's identity and get details on the attack execution. Thus, alarms' integrity and unforgeability are two requirements that must be ensured to consider them as valid evidence. Today, only few commercial SIEMs ensure these requirements through modules that sign the alarms with classic cryptography algorithms, such as RSA or DES. This approach is not resilient to attacks against the module that generates signed records.

In this paper we proposed an enhanced SIEM system that overcomes the limits described above. In particular, the proposed SIEM is designed by integrating: an advanced security information and event collector enabling multiple layer data analysis, namely the Generic Event Translation (GET) framework; the Decision Support System that allows both to resolve the conflict between security policies when it is raised and to analyze/control IT networks. IT networks monitoring and control allow to discover unauthorized data paths and perform automatic re-configuration of network devices; a Resilient Event Storage system that ensures integrity and unforgeability of alarms even in the case of attacks against its components. Finally we analyzed the Hydroelectric Dam Critical Infrastructure. In particular we propose a misuse case that mimics the attack that can be performed by the China government as described above.

II. ENHANCED SIEM ARCHITECTURE

In this section we discuss the conceptual architecture of the enhanced version of the SIEM system we propose. We present it through the block diagram in Fig. 1 where an Hydroelectric Dam is the Critical Infrastructure we want to protect from cyber-attacks. These facilities host the information sources, such as, *e.g.,* sensors, logical and physical access events, network system, that we monitor through the SIEM. Hence, in
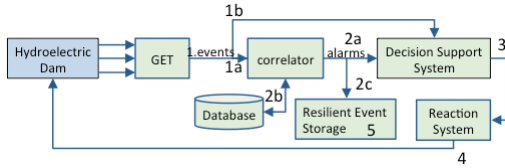
Fig. 1. Enhanced SIEM Architecture

the following, we firstly describe how an Hydroelectric Dam works and we point out possible security threats. Then we detail the workflow of the proposed architecture.

A. An Hydroelectric Dam

The purpose of a hydroelectric dam is to feed the power grid as any common generator. The simplified process that allows the generation of hydropower is the following: - the reservoir is fed by a river; - a certain amount of water contained in the reservoir flows through more than one penstocks and move a turbine; - the turbine is linked to an alternator that converts the mechanical energy into electrical energy; - the electrical energy is injected in a transmission line of the power grid. The power generated by hydroelectric dam mainly depends on two quantities: the water flow rate Q provided to the turbine through the penstocks and the difference between
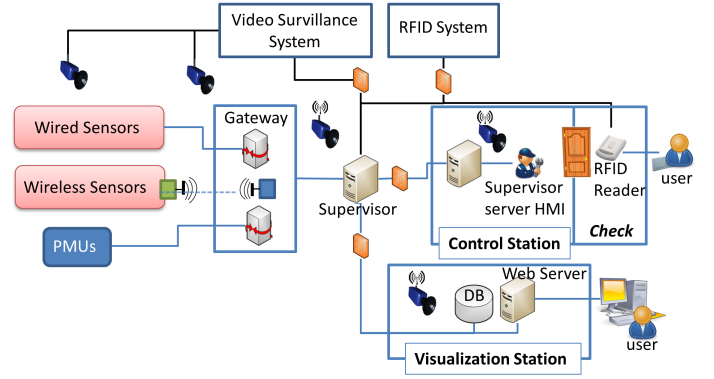
Fig. 2. IT infrastructure to support dam monitoring and control

water level in the reservoir and water level in the turbine Δh. The complete expression used to calculate the power generated by turbine rotation is described from the expression 1

$$P = \rho * \eta * g * \Delta h * Q \tag{1}$$

where: ρ is the water density ($10^3 \frac{Kg}{m^3}$), η is the efficiency of the turbine and g is the gravitational constant ($10 \frac{m}{s^2}$). If we suppose that Δh is a constant in expression 1, then we obtain that the power is only a function of the water flow rate Q. With this assumption it is possible to increase the power generated P by increasing Q. Hence, if the water flow rate increases without control, a state of emergency is generated because this implies an increase of power generated that can lead to a damage of the turbine. Indeed, higher power implies higher number of the turbine rotations. If the number of rotations per minute overcome a fixed threshold, then the turbine is destroyed and/or the electric power generated overcomes the security threshold. In Fig. 2 we show a simplified view of IT systems that are used to monitor and control the dam process. In particular, the 'visualization station' shows data gathered to generate statistics or to monitor a specific process. The 'control station' allows to control the process, *e.g.,* it is possible to send commands to sensors and actuators. Wireless or wired sensors are used to monitor different physical dam parameters in order to assess the safety of the global dam and foresee possible failures or anomalies [9]. Both stations and sensors can be attacked by a malicious user. Indeed, visualization and control stations can be subjected to attacks. For example, due to a misconfiguration the firewall allows an user in the 'visualization station' to re-write the sensor firmware; an unsatisfied employee discovers such wrong configuration in network devices, and so he/she exploits this vulnerability to perform a serious attack to the hydroelectric dam. Also, wireless sensors can be attacked from insider attackers as well as from outsider ones that try to violate or spoof the communication among sensors in order to obtain some key information useful to violate the dam architecture. In all these cases, control stations and sensors represent possible points of failure of the architecture and they have to be monitored in order to rapidly identify and solve the occurred security issue.

B. The proposed Architecture

As depicted in Fig. 1, the workflow is made of the following steps:

1. The events generated by the Hydroelectric Dam are monitored by the GET module that is the security collector software. The purpose of the GET is to generate security events by observing multiple layer data from the sources in the monitored infrastructure. The GET translates such events into a common format which is suitable for both the central Correlator engine (1a) and for the Decision Support System (1b). **2.** The Correlator analyzes the GET events to discover known attack signatures, *i.e.,* signatures encoded through schematic rules and stored in the rule database of the SIEM (2b). If one of the patterns of security breach is found, then the Correlator generates an alarm and sends it to the Decision Support System (2a) and to the Resilient Event Storage (2c). Generated alarms contain information about the security breach, so they are useful for forensic purposes. **3.** The Decision Support System (based on XACML engine) ensures that the security policies established are not violated and it implements a resolution strategy when two policies are in conflict. Also, the Decision Support System uses a novel modelling approach based on the matricial representation of network device configurations (*i.e.,* policies) which allows the computation of the reachability analysis. The reachability analysis is an approach useful to discover unauthorized data paths between hosts due to a misconfiguration of network infrastructure. **4.** If the Decision Support System discovers a misconfiguration, it activates the Reaction System that performs a control action on the scenario, *e.g.,* a data path discovered and unauthorized is closed by modifying firewalls rules. **5.** In order to use the alarms generated and stored by the SIEM as evidence of cyber-crime, integrity and unforgeability must be guaranteed. The Resilient Event Storage is an intrusion and fault tolerant facility designed to ensure these two requirements even if some components are compromised by an attack.

III. DESCRIPTION OF NEW SIEM COMPONENTS

In this section we provide more details about GET module, the Decision Support Systems functionalities, and the Resilient Event Storage system, respectively.

A. Probes and Correlation Process

The GET framework gathers data from probes deployed within the monitored infrastructure in order to collect security information on specific processes, and generates events when suspicious activities are detected. The event format generated by the probes as output of the GET is the same in order to enable processing of the *Correlator* engine. In the GET framework the process of excerpting semantically richer information from the logs is realized in a number of sub-tasks including: gathering of raw and heterogeneous data, correlation and abstraction of security relevant facts based on complex analysis [10] [11]. All processes are performed at the edge of the SIEM - *i.e.,* in the network of field systems - and are assigned to different modules organized as in Fig. 3. Key components of the GET framework are the *Adaptable Parsers* and the *Security Probes*. Adaptable Parsers (APs) are high performance data parsers relying on grammar-based compilers. APs are hardcoded parsers obtained by a formal description (grammar) of the data structures and they extract tokens from the data streams given as input to the GET framework. Security Probes (SPs) are event pattern detectors based on complex

State Machine models. SPs exploit flexibility of statecharts and minimize the effort of implementing all transitions to define security event patterns. In Fig. 1, *Correlator* engine
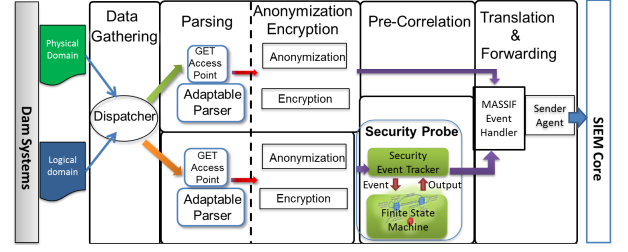

Fig. 3. Generic Event Translation (GET) framework: architecture

is a software component that allows to detect specific attacks signatures within events flow received by GET. When an attack signature is matched, the Correlator generates an alarm. The alarm generated contains also information about the events that generated it. Alarm generation through Correlator is performed in order to improve the accuracy of incident diagnosis and allow better response procedures. The Correlator shows few, semantically richer alarms in the face of the huge number of events coming from single sensors. The well-known attacks signatures are defined through the *correlation rules*. In particular, a correlation rule describes a relation between some information contained in the fields of events gathered in order to identify an attack. An example of correlation rule that can be used, for example, to discover a brute-force attack is shown in Fig. 4. In particular the rule establishes that a brute-force
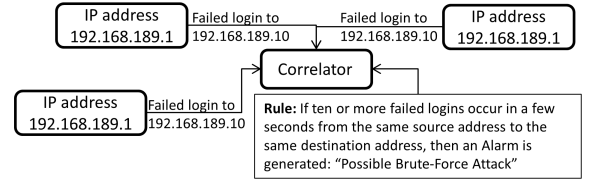

Fig. 4. Correlation rule example to discover a brute-force attack

attack occurs when many failed logins are performed by same source IP address to the same destination IP address. Thus, if this signature is found within events flow analyzed an alarm is generated and then a reaction can be activated.

B. Decision Support System

The Decision Support System exposes two main functionalities: i) a policies conflict resolution strategy that allows to solve conflicts occurring among different XACML-based policies that can be applied at the same time but that allow conflicting actions; ii) a reachability analysis that is able to discover unauthorized network access and allows to re-define network configurations.

1) Policies Conflict Resolution: It is based on the Analytical Hierarchy Process (AHP) [12], [13], a well-know multi-criteria decision system. The AHP approach requires to subdivide a complex problem (*i.e.,* ranking conflicting policies) into a set of sub-problems, equal in number to the chosen *criteria*, and then computes the solution (*i.e.,* choose the applicable policy) by properly merging all the local solutions
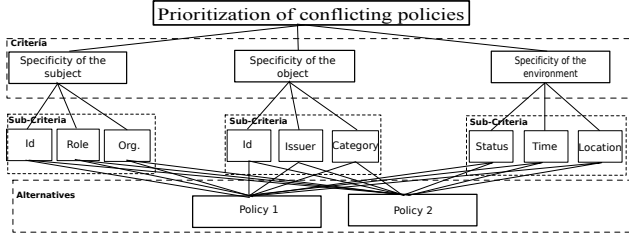
Fig. 5. AHP Hierarchy for Policy Conflict Resolution [14], [15].

for each sub-problem. In Fig.5 we show a possible instantiation of the AHP hierarchy for policy conflict resolution showed in [14], [15]. The goal (the box on top of the hierarchy in Fig.5) is "select the policy" to among conflicting ones, *e.g.,* Policy 1 and Policy 2 in the boxes at the bottom of the hierarchy. To solve conflicts we consider as *criteria* (second group of boxes starting from the top of the hierarchy) the *specificity* of the elements that constitute a policy: i) *Specificity of the subject*, in which we evaluate the attributes exploited in the two policies to identify the subject, to determine which of the policies define a more specific set of subjects. ii) *Specificity of the object* in which we evaluate the attributes exploited in the two policies to identify the object. iii) *Specificity of the environment* in which we evaluate the attributes to identify the environment.

Furthermore, AHP features the capability to further refine each criterion in sub-criteria, by considering the attributes that identify each element, *e.g.,* for the subject, the Identification Number (ID, the subject Role, the organization the subject belongs to). It is worth noticing that the set of considered attributes depends on the chose scenario. Once the hierarchy is built, the method performs pairwise comparison, from bottom to top, in order to compute the relevance, hereafter called *local priority*: i) of each alternative with respect to each sub-criteria, ii) of each sub-criterion with respect to the relative criterion, and finally, iii) of each criterion with respect to the goal. Note that, in case of a criterion without sub-criteria, the local priority of each alternative is computed with respect to the criterion. Comparisons are made through a scale of numbers typical to AHP (see Table I) that indicates how many times an alternative is *more relevant* than another.

Intensity	Definition	Explanation
1	Equal	Two elements contribute equally to the objective
3	Moderate	One element is slightly more relevant than another
5	Strong	One element is strongly more relevant over another
7	Very strong	One element is very strongly more relevant over another
9	Extreme	One element is extremely more relevant over another

TABLE I. FUNDAMENTAL SCALE FOR AHP

We define an ordering among the attributes related to the same policy element. This ordering expresses how including in a policy a condition on a given attribute contributes to make the policy more specific. Roughly, attribute *a1* of element *e* is more *specific* than attribute $a2$ of the same element if a condition on this attribute is likely to identify a more homogeneous and/or a smaller set of entities within *e*. For example, in Fig. 5 the subject ID is more specific than the Role.

To calculate local priorities, we perform k 2x2 pairwise comparison matrices, where k is the number of subcriteria (in our case, k=9). Matrices are built according to the presence of the

attributes in the policies. Given that a_{ij} is the generic element of one of these matrices: i) Policy1 and Policy2 contain (or do not contain) attribute A: then $a_{12} = a_{21} = 1$. ii) If only Policy1 contains A, than $a_{12} = 9$, and $a_{21} = \frac{1}{9}$. iii) If only Policy2 contains A, than $a_{12} = \frac{1}{9}$, and $a_{12} = 9$. Once a comparison matrix has been defined, the local priorities can be computed as the normalized eigenvector associated with the largest eigenvalue of such matrix [16].

Then, moving up in the hierarchy, we quantify how sub-criteria are relevant with respect to the correspondent criterion. Hence, we evaluate how the attributes are relevant to identify the subject, the object and the environment: *e.g.,* referring to the subject, ID is more relevant than Role and Organization. Role and Organization have the same relevance.

Finally, we quantify how the three criteria are relevant for achieving the goal of solving conflicts. Without loss of generality, we hypothesize that all the criteria equally contribute to meet the goal. The global priority is calculated according to the following general formula:

$$P_g^{a_i} = \sum_{j=1}^{n_1} \sum_{k=1}^{q(w)} p_g^{c_w} \cdot p_{c_w}^{sc_k^w} \cdot p_{sc_k^w}^{a_i} + \sum_{j=1}^{n2} p_g^{c_j} \cdot p_{c_j}^{a_i} \qquad (2)$$

where we have in mind a hierarchy tree where the leftmost $n1$ criteria have a set of sub-criteria each, while the rightmost $n2$ criteria have no sub-criteria below them, and $n1 + n2 = n$ is the number of total criteria; $q(w)$ is the number of sub-criteria for criterion c_w, $p_g^{c_w}$ is the local priority of criterion c_w with respect to the goal g, $p_{c_w}^{sc_k^w}$ is the local priority of sub-criterion k with respect to criterion c_w, and $p_{sc_k^w}^{a_i}$ is the local priority of alternative a_i with respect to sub-criterion k of criterion c_w. $p_{c_w}^{sc_k^w}$ and $p_{sc_k^w}^{a_i}$ are computed by following the same procedure of the pairwise comparisons matrices illustrated above. In this straightforward case, the pairwise comparison matrix is a 4x4 matrix with all the elements equal to 1, and the local priorities of the criteria with respect to the goal are simply 0.25 each.

It is worth noticing that, in our approach, we do not consider as a decisional criterion the specificity of the action. This is because we evaluate the action only according to its ID, always present in a policy.

2) Reachability analysis: The objective of DSS for reachability analysis (depicted in Fig. 6) is to discover unauthorized network access. An unauthorized network access occurs when firewall rules are modified by non authorized personnel (*e.g.,* by an attacker), by authorized personnel (*e.g.,* configuration mistakes) or when a policy is not enforceable (*e.g.,* no available firewall to enforce the policy). The main idea to detect these situations is the adoption of network reachability to identify which services are reachable from a set of hosts by traversing devices of a network. The proposed approach adopts the reachability analysis of the filtering rules comparing the ones derived from policies with the firewall configurations. However policies are defined by using an abstract language (*e.g.,* subject, action, object) and are topology independent (*i.e.,* the network topology is not considered during policy authoring). To simplify the policy definition we introduce the action *reach* that specifies which network interactions (between subject and object) are authorized. This approach simplifies the policy management, *e.g.,* the undefined interactions are prohibited as default and policy conflicts are avoided (only

some anomalies must be addressed, *e.g.,* equivalent rules). On the contrary, firewall rules are represented by using a common format (source IP address, source port, destination IP address, destination port, protocol, action) and depend on network topology, *i.e.,* where the firewall is placed in the network and which set of hosts protects. Therefore policies must be transformed into a concrete format (this operation must be executed for each filtering device of the network) before performing reachability analysis. The DSS, by using a set of rules and an inference engine, drives the process to detect unauthorized network access. At the beginning, or when policies are modified, DSS starts the refinement process to generate the set of rules for filtering devices. This process is organized by a set of *Policy refinement tasks* as depicted in Fig. 6. First of all, policies and system description are analysed and a graph-based network topology representation is generated. In particular, during the policy analysis, the anomalies (*e.g.,* redundancy) are detected and addressed. System description (represented as XML) contains hosts (and related information, *e.g.,* IP addresses), capabilities (*e.g.,* filtering), services and network topology. By using *Network analysis* the process identifies the set of firewalls to enforce each policy. This task discovers, on the graph-based representation, the network paths (that include at least a firewall) between the subject and the object of a policy. Since the default action of a firewall is to deny all traffic, each firewall contained in a path must be configured to permit the policy traffic. Hence, for each firewall a set of filtering rules is generated to enforce the policies. When it does not exist at least a firewall to implement the policy, it is not enforceable. This typically occurs when subject and object belong to the same subnet and their traffic does not traverse any firewall. Therefore any type of traffic between subject and object is permitted, potentially creating a security breach. This situation is managed by the DSS, that logs the *security issue* and saves it into the *internal models* repository. Once the refinement process is completed, the DSS performs the reachability analysis evaluating the filtering rules, the ones generated by the previous process (*generated rules*) and the ones deployed into firewalls (*deployed rules*). This activity is organized in the following phases:

I. **Expansion**: For each *generated* and *deployed* rule, the fields that contain ranges (*e.g.,* IP addresses, ports, *etc...*) are expanded by considering network description (*i.e.,* the hosts and services defined into the system description) and creating new rules. Let's consider the rule $r1$ as *srcIP:192.168.0.1*, *srcPort:**, *dstIP:192.168.10.10*, *dstPort:80,443*, *proto:TCP*, where the destination IP address refers to a host that offers a web service on ports 80 and 443. The expansion phase transforms $r1$ into rules $r1, 1$ and $r1, 2$: the former to match the port 80 and the latter to match the port 443. The same approach is followed for the IP addresses expressed as subnets. Before applying expansion operation, *deployed* rules are analysed to detect and address anomalies;

II. **Composition**: the objective of this phase is to create the reachability matrices. Each firewall i has two rule sets, one for *generated rules* ($R_{g,i}$) and another for *deployed rules* ($R_{d,i}$). We introduce the equivalent rule set for the firewall i ($R_{e,i}$) that contains both generated and deployed rules, *i.e.,* $R_{e,i} = R_{g,i} \cup R_{d,i}$. Considering the $R_{e,i}$ we create two partitions: the former contains source IP address and port fields ($S_{IP,port}$) and the latter the destination IP address, port and protocol ($D_{IP,port,proto}$). A two-dimensional reachability matrix for a firewall i (M_i) has $S_{IP,port}$ as row and $D_{IP,port,proto}$ as column. The composition phase, for each firewall i: 1. creates two matrices: $M_{g,i}$ for *generated* rules and $M_{d,i}$ for *deployed* rules. Each matrix contains the $S_{IP,port}$ individuals as rows and $D_{IP,port,proto}$ as columns; 2. computes $M_{g,i}$: for each rule r of $R_{e,i}$, *i.e.,* $r \in R_{e,i}$: if r is part of $R_{g,i}$ rules, *i.e.,* $r \in R_{g,i}$, sets 1 for corresponding row and column, otherwise 0; 3. computes $M_{d,i}$: for each rule r of $R_{e,i}$, *i.e.,* $r \in R_{e,i}$: if r is part of $R_{d,i}$ rules, *i.e.,* $r \in R_{d,i}$, sets 1 for corresponding row and column, otherwise 0;

III. **Analysis**: this phase compares reachability property of *generated* with *deployed* rules. For each firewall i, we compute $M_{\rho,i} = M_{g,i} - M_{d,i}$. If $M_{\rho,i} = 0$ (when all the elements are equal to 0) the reachability for *deployed* and *generated* rules is the same, *i.e.,* no security issues are identified. Otherwise ($M_{\rho,i} \neq 0$) at least an element is equal to 1 or -1. In the first case the corresponding rule is not deployed into the firewall. Therefore, the firewall configuration drops a packet that must be permitted by the policy. This situation is reported as an *anomaly*. Otherwise (equal to -1) the corresponding rule is enforced by the firewall configuration but it is prohibited by the policy. In this situation, the firewall contains a misconfiguration and the DSS logs it as *security issue*. Finally, the DSS evaluating reachability analysis reports detected issues (*i.e.,* *anomalies*, *security issues*) and proposes a *remediation*, *i.e.,* a list of suggestions on how to modify the firewall rules or where to install a filtering device (*e.g.,* personal firewall) to enforce the policy.
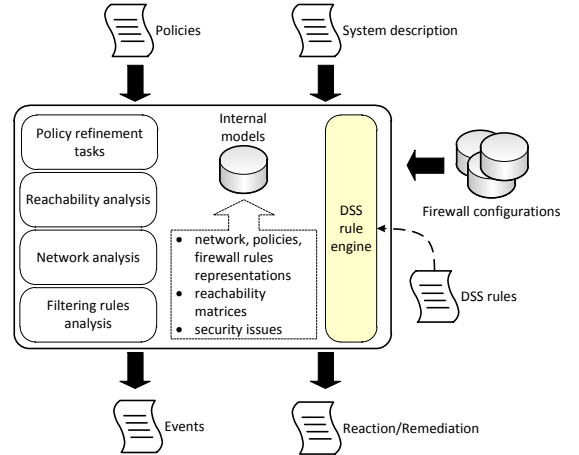

Fig. 6.   Architecture of DSS for reachability analysis

C. Resilient Event Storage

Resilient Event Storage (RES) system is an infrastructure designed: to be tolerant to faults and intrusions; to generate signed records containing alarms/events related to security breaches; to ensure the integrity and unforgeability of alarms/events stored. In particular, the RES fault and intrusion tolerant capability makes it able to correctly create secure signed records even when some components of the architecture are compromised. The RES conceptual architecture is shown in Fig. 7. The basic principle is to use more than one secret key. Specifically, the main secret key is one but it is divided in n parts, namely shares, and each share is stored by a different
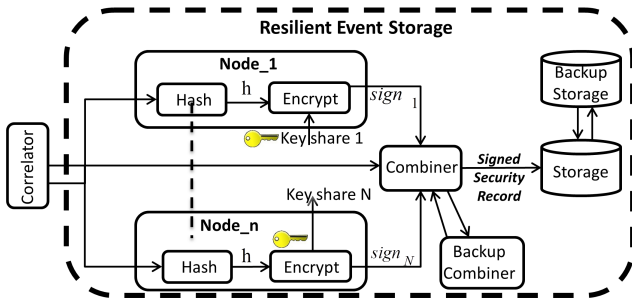
Fig. 7. Resilient Event Storage Architecture

node. This approach can be realized by Shoup threshold cryptography algorithm [17]. The most important characteristic of a threshold cryptography algorithm is that the attacker has zero knowledge about the secret key, if less than $(k-1)$ secret key shares are compromised. Threshold cryptography algorithm is characterized by two parameters: n *i.e.,* the number of nodes and k *i.e.,* the security threshold. The output of a cryptography algorithm and its threshold version are equivalent. In the RES architecture, a component called Dealer is responsible to generate n secret key shares, n verification key shares and one verification key from a main secret key. This component is not shown in Fig. 7 because it is used only in the initialization phase. After Dealer generated the keys, it sends each secret key share to each node whereas n verification key shares and the verification key are sent to the Combiner. Each verification key share is used to check the correctness of each signature share generated by each node with its own secret key share. The verification key is used to check the correctness of complete signature generated when the Combiner puts together the signature shares provided by nodes. Input data to the RES architecture are provided by Correlator (Fig. 1) because the alarms contain information about a security breach and they need to be stored in secure way. The incoming alarm is sent to all nodes and to the Combiner component. Then, each node computes a hash function of the received alarm. This function returns a digest for this alarm, represented by h in Fig. 7. The next step is to encrypt the digest with the secret key share in order to produce a signature share and send it to the Combiner. When the Combiner receives from nodes at least k signature shares for the same alarm it can assemble all partial signatures in order to generate a complete signature. Then the Combiner verifies the complete signature through the verification key. If the verification process fails, the Combiner verifies the correctness of each signature share using the corresponding verification key share. When the node that sent the wrong signature share is discovered it is flagged as corrupted. Next time if new signature shares are available for the same alarm, the Combiner uses the already validated signature shares and the new signature shares to create a new set of k signature shares. Then the Combiner generates a new complete signature and repeats the verification process. If the verification process is successful this time, then the complete signature, the original alert and the identifiers of corrupted nodes are stored in a storage system. Further details about RES implementation are described in [18]. In order to improve the intrusion and fault tolerance of RES, replication and diversity are employed in the media storage and the Combiner component.

IV. CONCLUSION AND FUTURE DIRECTIONS

In this paper we provided an enhanced SIEM architecture able to cope with cyber security problems that may occur in a Critical Infrastructure. Indeed, starting from a discussion about the limitations of the existing SIEM systems, we proposed a new enhanced SIEM architecture in which we have introduced functionalities for enabling multiple layer data analysis, resolving conflicts among security policies, and discovering unauthorized data paths in such a way to be able to reconfigure network devices. Also, we provided a sketch of a possible usage of our architecture when a misuse case affects hydroelectric dam. As described in [3], this Critical Infrastructure is highly exposed to cyber attacks today. We are currently working to perform an extensive experimental campaign with the purpose of 1) setup the system we proposed in order to integrate all the components and 2) validate the enhanced SIEM proposed in Hydroelectric dam scenario.

REFERENCES

[1] *What Is Critical Infrastructure?*, Department of Homeland Security Std.

[2] *Presidential Policy Directive – Critical Infrastructure Security and Resilience*, Department of Homeland Security Std., 2013.

[3] *Industrial Control Systems Cyber Emergency Response Team (ICS-CERT).*, Std., 2013.

[4] *Sensitive Army database of U.S. dams compromised; Chinese hackers suspected*, 2013.

[5] *Protect critical infrastructure*, McAfee Std., 2012.

[6] *Carr, D.F.: Security Information and Event Management. Baseline, No. 47, p. 83*, Std., 2005.

[7] A. Williams, *Security Information and Event Management Technologies, Siliconindia, Vol. 10, No. 1, pp. 34-35*, Std., 2006.

[8] *SIEM and IAM Technology Integration*, Gartner Std.

[9] J. Yang, T.-d. Bao, D.-s. Liang, Y.-f. Mi, and L. Yang, "Management information system for dam safety monitoring based on b/s structure," in *Proceedings of ICISE '09*. IEEE Computer Society, 2009, pp. 2332–2335.

[10] F. Campanile, A. Cilardo, L. Coppolino, and L. Romano, "Adaptable parsing of real-time data streams," in *Parallel, Distributed and Network-Based Processing, 2007. PDP '07. 15th EUROMICRO International Conference on*, Feb 2007, pp. 412–418.

[11] L. Coppolino, S. D'antonio, M. Esposito, and L. Romano, "Exploiting diversity and correlation to improve the performance of intrusion detection systems," in *N2S '09*, June 2009, pp. 1–5.

[12] T. L. Saaty, "Decision-making with the ahp: Why is the principal eigenvector necessary," *European Journal of Operational Research*, vol. 145, no. 1, pp. 85–91, 2003.

[13] ——, "Decision making with the analytic hierarchy process," *International Journal of Services Sciences*, vol. 1, no. 1, 2008.

[14] I. Matteucci, P. Mori, and M. Petrocchi, "Prioritized Execution of Privacy Policies," in *DPM/SETOP*, 2012, pp. 133–145.

[15] A. Lunardelli, I. Matteucci, P. Mori, and M. Petrocchi, "A Prototype for Solving Conflicts in XACML-based e-Health Policies," in *CBMS*. IEEE, 2013, pp. 449–452.

[16] T. L. Saaty, "A scaling method for priorities in hierarchical structures," *Journal of Mathematical Psychology*, vol. 15, no. 3, pp. 234–281, 1977.

[17] V. Shoup, "Practical threshold signatures," in *Proceedings of EURO-CRYPT'00*, ser. LNCS, 2000, pp. 207–220.

[18] M. Afzaal, C. Di Sarno, L. Coppolino, S. D'Antonio, and L. Romano, "A resilient architecture for forensic storage of events in critical infrastructures," in *HASE 2012 IEEE*, 2012, pp. 48–55.