Assessing Incident Severity in a Network and Automatic Defense Mechanisms *

Luis Servin, Till Dörges, Klaus-Peter Kossakowski
{ls,td,kpk}@pre-secure.de
PRESECURE Consulting GmbH

May 27, 2007

Abstract: *Threat sources for computer networks are diverse and increasingly complex. Attackers make use of vulnerabilities or configuration mistakes to pass through the external lines of defense and break into different hosts or pry on what should otherwise be a secure/private communication channel.*

*Unfortunately, the means to prepare for and to detect on-going attacks work mostly isolated. Among them we can count firewalls, Intrusion Detection System (*IDS*), Intrusion Prevention System (*IPS*), and honeypots, as well as the possibility of doing penetration tests from within or from outside the network.*

Combining all these methods at hand, there is a lot of information available. Information that has to be processed into knowledge that will help in assessing the current situation. This assessment can be used in a policy-administered network to change the ruling policies to a deteriorating or improving situation, accordingly. The amount of information to be processed and its correct appraisal to the overall impact on the network is not trivial and could overwhelm a person trying to do it manually.

This paper presents a framework for policy-based network administration that uses the input from different sensor types to assesses the situation and decide on an action to counter a possible attack. Actions range from (semi-)automatically changing the security policies for the whole network, to reconfiguring a service within a host.

In particular, the processing method to make the assessment will be the core of this article.

Keywords: *Network Health Assessment, Multi-sensor Correlation, Policy-based Security, Fuzzy Cognitive Maps*

1 Introduction

Modern communication networks are made up of many different systems – like printers, servers, routers, and computers – providing particular services. The correct configuration and administration of so many heterogeneous systems and services demands a broad and deep knowledge, that might surpass the capacity of a person. Policy-based network administration offers a significant relief for administrators by offering a process that largely simplifies and automates their task [4, 6].

Network, services, and application control and management are defined at a high abstraction layer through policy rules. This switches the focus from the devices and interfaces to users and applications that interact according to certain business rules.

*This work is part of the POSITIF project, funded by the EC under contract no. IST-2002-002314

The Internet Engineering Task Force (IETF) Policy Framework Working Group in conjunction with the Distributed Management Task Force (DMTF) [7] have come up with a policy meta-model [12]. Within this model a policy architecture definition has been created. It is depicted in Figure 1, and it is made up of a Policy Management Point (PMT) to define policies in a centralized location; a Policy Repository to store the policies created at the PMT; a Policy Decision Point (PDP) that interprets the policies and transmits them to a Policy Enforcement Point (PEP). This last element can enforce and apply the policies at the devices it controls. But having policies is not enough if they cannot be monitored for compliance and dynamically adjusted to a changing environment.

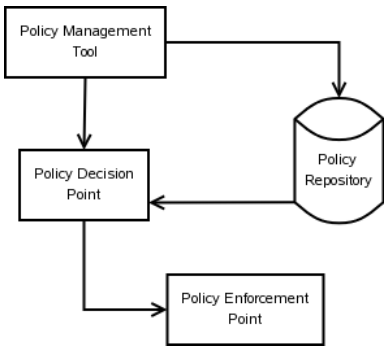


Figure 1: IETF/DMTF Policy Framework

There are multiple ways in which a network can be protected against attacks. Common methods include Firewalls, Intrusion Detection System (IDS), Intrusion Prevention System (IPS), and Honeypots. Penetration tests can also be used, to test for vulnerabilities at various levels. All these can be used together to monitor the policies set by the administrator.

When deciding among so many different solutions and implementing them, one is faced with the daunting task of being attentive to all their different outputs. All these systems tend to keep logs that increment with every event they recognize as malicious and they can also be configured to send emails to the system administrator or to make a summary of events in an administrative console.

Each alert message is typical to the application producing it. Therefore, they tend to be incompatible among each other, thus making it difficult to use quick search methods to sort the information. An intermediate step is required to standardize them. A standardization example can be found in the Incident Data Message Exchange Format (IDMEF) [11]. Then the messages can be handled by a central instance. Without this standardization, the complexity of extracting useful information from the alerts and making an estimation of the danger they pose is too big.

The POSITIF[22] project[1] parts from the IETF-DMTF model. It will provide administrators with a framework and tools for centralized computer network management. Furthermore, it extends the model to include the Proactive Security Monitor (PSM), an element capable of monitoring the network for signs of intrusion and vulnerabilities through security elements. There are different types of security elements available, some active and some passive in nature.

The PSM complements the policy-setting capabilities of the framework with their supervision and a reasoning ability to help in countering intrusions. It assesses the current situation of the supervised network and, in case it were necessary, it initiates changes in the policy set. These changes make it possible to go from more relaxed to more constrictive rules, or vice versa [9]. This article is centered on the method to produce this situational assessment, and

[1] POSITIF is still a work in progress

the defined actions that can be taken to help in countering an intrusion and possibly provide intrusion tolerance to the different hosts in a network.

The rest of this article is organized as follows:

section 2 presents the POSITIF framework as a whole, and the Proactive Security Monitor in particular is detailed

section 3 analyzes some methodologies used in the past for countering attacks and changing security policies, which are used as base for the PSM Assessment (PSM-A)

section 4 details the assessment process done by the PSM-A, finally

section 5 presents the conclusions and future research directions.

2 POSITIF Framework

POSITIF aims at supplying administrators with a framework and tools for centralized management of computer networks – following a policy-based philosophy. It parts from the IETF-DMTF model and extends it in two ways. The first is meant to support the notion of business policies defined in the PMT. The second provides a monitoring element to supervise the observation of the policies and take actions destined to correct deviations.

Figure 2 corresponds to a possible solution of the problem posed by defining policies as business rules, and transforming them in device specific rules. It is based on [6, 21]. In such a generic PMT, it is necessary to complement the business policies, written in a high-level policy language, with extra information, like topology and methodology. All these elements are then turned by a transformation element into low-level, device-specific policies. These are then the policies distributed.

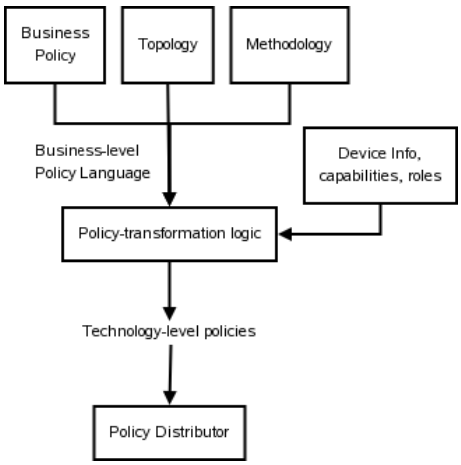


Figure 2: Generic Policy Management Tool

POSITIF has implemented such a PMT, which has as a consequence a workflow like the one illustrated in Figure 3. As expected from Figure 2, a high-level policy description is input, in the Security Policy Language (SPL). Policies include information about devices. This device information is reflected from the topological information, provided in the System Description Language (SDL). Both inputs are created as XML documents, defined in their corresponding schema document (XSD).

The SDL describes, among other things, the network topology, connected hosts with their respective operative systems, and running services. Every defined element receives an ID tag, through which it can later be referred to. Further, it is possible to define the sensitivity of any topological element in terms of its *confidentiality*, *integrity*, or *availability* (CIA triad). This provides an extra layer of knowledge to the deployment of the network, with which the importance a given element presents to the network as a whole can be assessed.

In the SPL it is possible to define five types of policies: authentication, authorization, filtering, channel protection, and operational. They define allowed and denied actions, thus characterizing the behavior inside the network. In order to have the ability to adapt to a changing environment, it is necessary to decide which policies are to be applied at any time with an estimated danger level. For this purpose, the concept of *Security Levels* has been created. At an elevated security level, only the most essential services would be kept on running. This would correspond to a perceived state of high-alert.

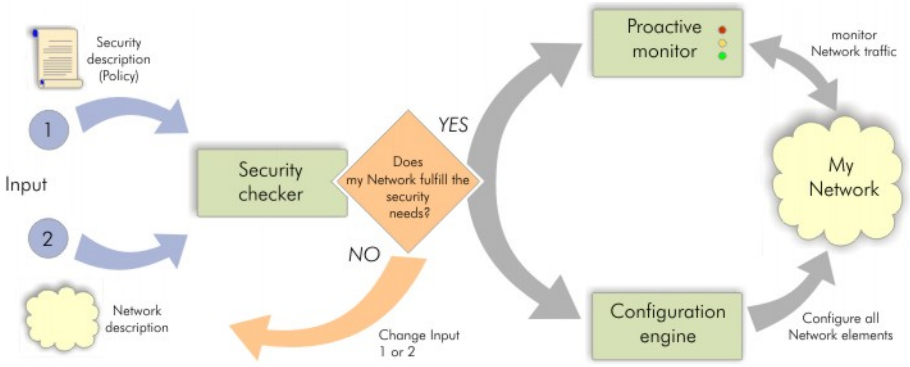


Figure 3: Schematic workflow of the POSITIF project

Both the SPL and the SDL are then be syntactically and semantically verified to see if the desired policies can be applied given the current network architecture. If verification is successful, both are united in a so-called internal format derived from the Common Information Model (CIM) [8]. This step would correspond to the Policy-transformation logic element from Figure 2.

Policies are then distributed and automatically deployed to all networked equipment managed by POSITIF through the Configuration Engine from Figure 3. This element corresponds to the PET. The POSITIF elements corresponding to the PDP and the Policy Repository are not illustrated in the workflow. More information can be found in [22] and [23].

After the deployment phase the Proactive Security Monitor (PSM) steps in because from now on compliance of all activities with the given policies needs to be monitored. Any deviations from allowed behaviors will be reported, and evaluated to determine the danger they present for the network as a whole.

The PSM is made up of a combination of reactive, proactive, and processing elements. They are used to complement each other in the tasks of monitoring the policies and assessing the situation of the network. A *reactive* (or *passive*) element is that which only reacts to a given situation, while a *proactive* (or simply *active*) element acts to prove if e. g. a policy holds or a configuration is correct. A *processing* element filters redundant information and condenses it in a single report. These elements are displayed in figure 4.

The *reactive* elements are:

IDS an Intrusion Detection System, which is observing network traffic to detect suspicious

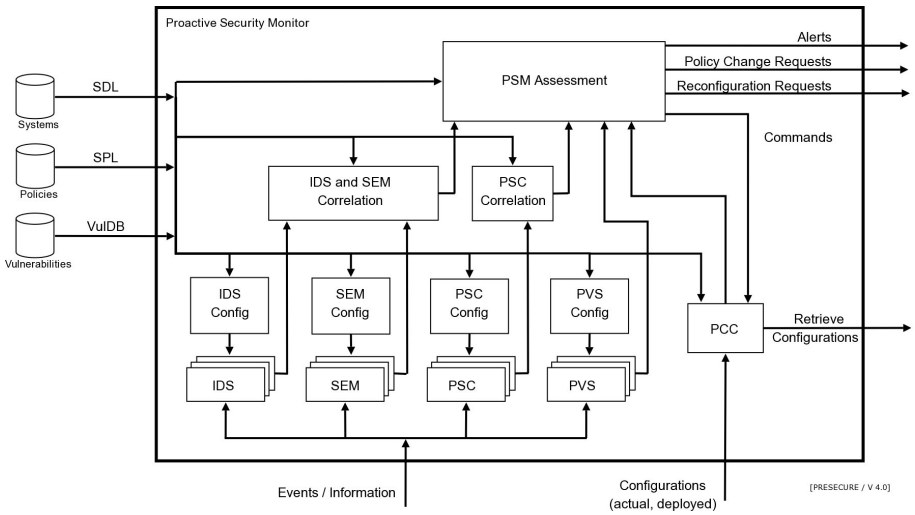


Figure 4: PSM Architecture

events, and reports them to a processing element. The development in this area builds upon SNORT[24], Argus [3], and raprelude [13]. The first one detects signals of intrusions from the network traffic, and the second and third are used to detect violations to the filtering rules defined in the `SPL`.

SEM a lightweight, Small-footprint sEcurity Module is also an `IDS`, but it is suitable for portable devices. It builds upon SNORT, but complements the misuse engine with an anomaly detector that uses Self-Organizing Feature Maps (SOM).

PVS Policy Violation Sensor is another passive sensing component which will recognize policy violations not already covered by the IDS and SEM components. There are violation sensors implemented for the following four protocols: SSL, HTTP, SMTP, and POP3.

The *proactive* elements are:

PSC the Proactive Security Scanner actively tries to retrieve and collect information about systems in order to conduct further analysis on different levels to identify existing vulnerabilities and other risks. It emulates user behavior to test the different policies in place and tries to exploit different vulnerabilities for the scanned host.

PCC the Proactive Configuration Checker continuously verifies that the configurations of the security blocks (applications) haven't changed from a "golden" standard. This was defined for them when configuring each application through the framework.

Finally, the *processing* elements are:

IDS and SEM Correlation a collector of all information from IDS and SEM components, as they are very similar to each other. In addition to data collection, its correlation is also done inside this component. Its design was strongly influenced by [26].

PSC Correlation as much information might actually point or relate to the same root cause, a collection and correlation of retrieved events is necessary.

PSM Assessment based on the information provided from all the elements in the module, a correlation on a higher level takes place. This effectively results in three major kinds of

outputs: *alerts*, *reconfiguration requests* and *policy change requests*. This component provides the ultimate situational assessment (in terms of the PSM). Its output can be handled automatically or manually. This decision will reside within the management responsible for the deployment of the POSITIF framework, while the framework will allow for automation according to the work plan.

Summarizing the information just presented regarding each sensor type, Table 1 is presented.

Sensor Type	Behavior	Recognizes
IDS/SEM	reactive	Attacks / Policy Violations
PVS	reactive	Attacks in form of Policy Violations to a protocol
PCC	proactive	Misconfigurations caused by potential attacks and Potential Policy Violations
PSC	proactive	Vulnerabilities, Potential Attacks and Potential Policy Violations

Table 1: Comparison of the different sensor types found inside the PSM

3 Incident Response

To counter intrusions inside the network, there are two foreseen methods: restoring the configuration of a security block (application) or changing the set of policies that govern allowed and prohibited behaviors inside the network. By using a reconfiguration approach at the application level, it is expected to help in making them intrusion-tolerant [25], by containing and ideally blocking the intruder, while minimizing the impact of the penetration.

3.1 Intrusion Tolerance

A survivable system has the ability to react or adapt in face of active intrusion. Only so it can survive an attack that cannot be repelled. Linger [17] makes a description of the requirements to transform a network into a survivable system. *Essential* from *non-essential* services need to be distinguished. The difference between the two resides in that the first group must be maintained even in face of successful intrusions. This separation can be planed by evaluating a priori the *C.I.A. triad* for all computer systems providing services to the network. This knowledge together with an according set of security policies should be considered as the first step towards survivability.

The requirements for survivability proposed by Linger are:

Resistance is the capability to deter attacks. Mostly provided through router authentication and encryption strategies, as well as restrictive provision for only a limited set of services.

Recognition is the capability to identify that an attack or probe is taking place. This is achieved through sensors, and the best is to use a set of heterogeneous sensor types. Then by correlating their outputs more reliable information can be won.

Recovery refers to the ability to restore services after intrusion occurred. It improves a system's capability to resist or recognize future events.

Stavridou describes in [25] an intrusion tolerant software architecture. She introduces a fault tolerance paradigm. It tries to move the emphasis from detecting an attack in progress to detect, diagnose, and recover from a deviation in expected system behavior. Intrusion tolerance is defined as *the ability to continue providing adequate service after penetration*. Thus, an intrusion tolerant service is inherently survivable. But neither naive replication of systems nor simple reconfiguration fulfill these criteria. Policies are used to distinguish between systems with different intrusion tolerance levels. Externally-observable behavior of systems is also modelled by Finite State Machines. The focus is on security-related failures, i. e. integrity and availability.

Castaldi [19] proposes a reconfiguration system to respond to intrusions. He sustains his point of view by stating that for a comprehensive network security management, all network devices must be correctly configured. Moreover, he ascertains that applications need to be reconfigured since the administrator relies on the security policies there implemented. The same applies to operating systems, in which packets and modules can be activated or deactivated. Furthermore, he specifies reactions at the application, host and network levels. This differs from our approach in that we only be act on the application level.

Sullivan [15] argues that survivability is the minimization of the aggregate cost, produced by the reduction of service within an acceptable bound under normal and adverse circumstances. When a system is disrupted, it must be adjusted to assure the provision of the information service. This adjustment involves system reconfiguration, which can occur on different levels: operating parameters, module implementations, code location, and physical device replacement. He uses a bank network as an example. When nodes corresponding to a branch detect that they are under attack, they report to their controlling node, which in turn might decide to reconfigure them.

Sullivan applies a decentralized hierarchichal and adaptive control model with four operating regimes: *No-attack*, *single-attack*, *regional-attack* and *widespread attack*. These regimes are represented within a Finite State Machine (`FSM`) that transitions between those states, depending on the types of alerts coming in and dictates the activities that take place for a system.

3.2 Security Policy Change

Chen and Yang [5] propose the management of policies for network-based intrusion detection and prevention. They propose an `Attack Response Matrix` (ARM) that maps intrusion types to traffic enforcement actions. They use `Policy Decision Points` (PDP), `Policy Enforcement Points` (PEP) and a policy moderator to set the policies and take action. They consider a key point of their project to perform actions at the reconnaissance or vulnerability scanning phase, before any malicious payload can be delivered. Similarly to [5] some elements within the `PSM-A` could be considered as our PEP and PDP. But we differ in the approach used to change the policies in place.

In [18] a language called *Ponder* is used to describe the policies, very similarly to what the `SPL` in `POSITIF` does. The adaptation of the policies is done by 1) specifying new attribute values, 2) enabling/disabling policies at run-time, and 3) learning most suitable policy from system behavior. This differs from our approach, in which from the beginning the policies are categorized, e. g. green, orange , red. Thus there is no need to adapt the policies. When the assessment determines that it is necessary to "escalate ", the next-level policies will be set in place.

4 Health Assessment

The PSM-A has the task of making an assessment of the current situation of the network as a whole. This will allow it to decide on an action to take. Actions can be alerts for human observation and interpretation, policy change requests that affect the network as a whole and reconfiguration requests that affect single hosts. First we will explore some efforts already done in this area, and then what we propose.

4.1 Related Work

Ambareen et al [2] propose a system to determine what they call the "overall security view" of a network. Their approach differs from the POSITIF approach in that it is based solely on reactive sensing. They propose that a system being attacked presents a staged decrease in its "health" that can be characterized as increasing suspicion level. For this purpose an Intelligent Decision Engine is introduced. This Decision Engine makes use of Fuzzy Cognitive Maps[16] (FCM) for doing causal knowledge inferencing.

The idea of using FCM is attractive since it allows to model the world as concepts and causal relations between them in a structured collection. A concept is an event that originates in a system with a value that changes over time. A causal relation links concepts and gives an indication of how much a value change in one affects the other. Three kinds of interaction can take place: no cause-effect, positive and negative causality. The first indicates independence between concepts. The second indicates that an increases in the first will produce an increase in the second. Finally, the third indicates that an increase in the first will produce a decrease in the second and vice versa.

Further in [1], Ambareen deepens the details with which the process takes place. He calls it the "Dynamic Fusion Model." This model consists of three steps: *prioritization*, *alert association*, and *situational assessment*. The first one discards alerts deemed as unimportant, based on the attack's and resources' criticality level. Alert association takes place in two simultaneous processes: *clustering*, for discovering structural relations between alerts, and *correlation*, for discovering cause-effect relations. Finally, the assessment is produced by adaptively fusing the output of both association processes. The fusion operators change following a measurement of agreement among the clustering and correlation outputs.

The Open Source Security Information Management[20] (OSSIM) provides a set of open-source tools along with a correlation engine inside a framework "for centralizing, organizing, and improving detection and display for monitoring security events within the organization." It is made up of sensors (IDS and anomaly), firewalls, and monitors. So far it is not so different from most current solutions, except that everything is integrated.

What is most interesting about OSSIM is how assessment is performed. OSSIM uses three parameters to determine the importance of an event:

1. the value of the asset involved in the event
2. the threat this event poses
3. the probability the event is occurring

Furthermore, each event will be accumulated by using two state variables that represent the levels of Compromise and Attack for each host. These two variables try to make a description of the situation of a given host in time. When there has been a sensed attack to it, its Attack level will increase. The compromise level of the attacker will increase as well. The compromise

level will, in the end, give a measure of the reliability of the host to work as expected. Finally the levels of both variables will be reduced in time by a constant value given that there are no new alarms.

4.2 The POSITIF Approach

First of all, the PSM Assessment module (PSM-A) maintains a list of hosts that are currently "under observation". This means that we suspect something's wrong with them because there have been alerts coming in from any of the sensors. This list is maintained in a relational database, which allows us to make fast queries to correlate information later on. One other important thing that is known by the PSM-A are the current state of the network and valuable assets therein.

When an alarm comes in, it gets categorized by sender. For each sender there can be more than 1 category, e.g. PSC has *begin* and *end* probe messages, as well as problem reports. To eliminate incompatibility between alerts, all of them must be structured following the IODEF standard.

Two processes take place simultaneously when an alert comes to the assessment module. The first involves a Finite State Machine (FSM) to decide on an action to take for the individual system. The second determines its effect on the involved systems and to the network as a whole, through the Dynamic Fusion Model, which uses FCM and fuzzy logic for the situational assessment. The description of both of them follows.

Systems that have not yet been involved in any incidents, are considered to be in an "idle" state. This is their ideal state. As illustrated in Figure 5, from the inactive state a host changes to another depending on the alert's source. It also plays an important role on the FCM used in the Dynamic Fusion Model when correlating all alerts for a host, as shown in Figure 7.

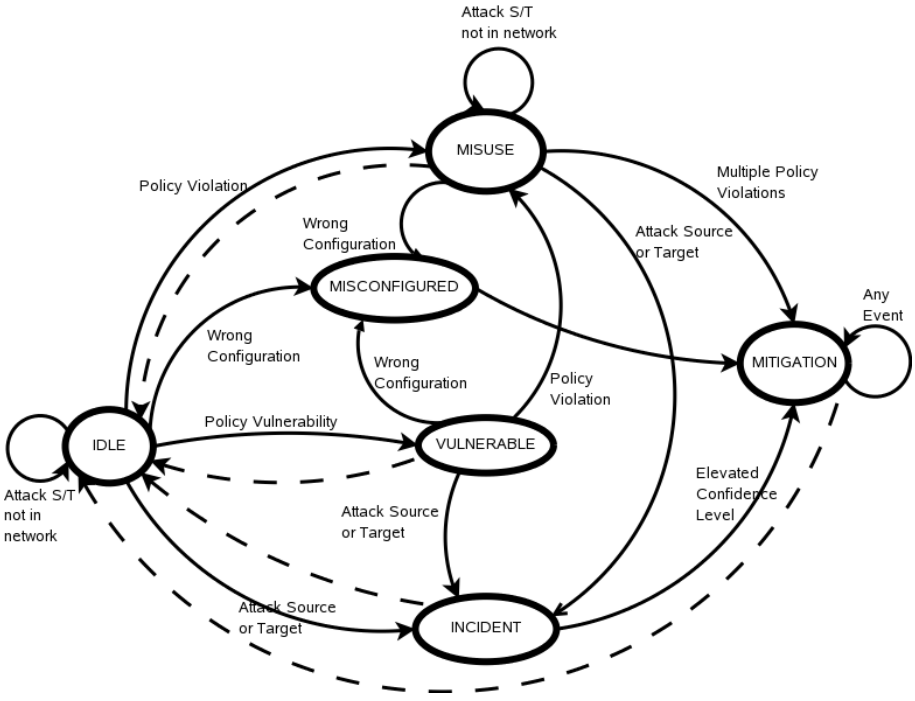


Figure 5: Different States to represent Hosts in PSM-A

Once a host finds itself outside the "idle" state, the assessment module will try to make it return there. After the reception of an alarm sent by one component, another one (an active one)

will be invoked to check on the suspicious host and report back on any found vulnerabilities or configuration problems. This can increase or reduce the suspicion that it has been compromised. It also produces further state changes. The ultimate state to be reached is "mitigation". This necessarily requires human intervation to mitigate and contain the damage.

The Dynamic Fusion Model from [1] has been adapted to the inputs available within POSITIF. The process can be seen in Figure 6. For the prioritization, the criticality of source and target, as indicated in the SDL, are taken together with the severity level of the alert. This value can be changed later on, if the PSC or PCC report the presence or absence any vulnerabilities or configuration problems.

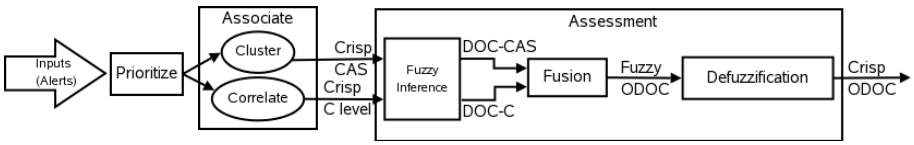


Figure 6: Dynamic Fusion Model

The association stage has to be modified to accommodate for a different methodology when performing the alert correlation. Alerts to be correlated come not only from IDS but also from other sensor types. All alert types must be used together to determine the level of concern for a system. This level of concern is expressed in our model as the "Confidence" (C) we have that the system has been compromised. We achieve this through the FCM shown in Figure 7. Here, it can be seen how each message received by the assessment module affects the "Attack" (A) or "Confidence" levels of a system. The first one is an indication of the attack density that a system is observing. A high density can ultimately lead to a denial of service or allow an intruder to break into a system, thus compromising it. For this reason it also affects its Confidence level.

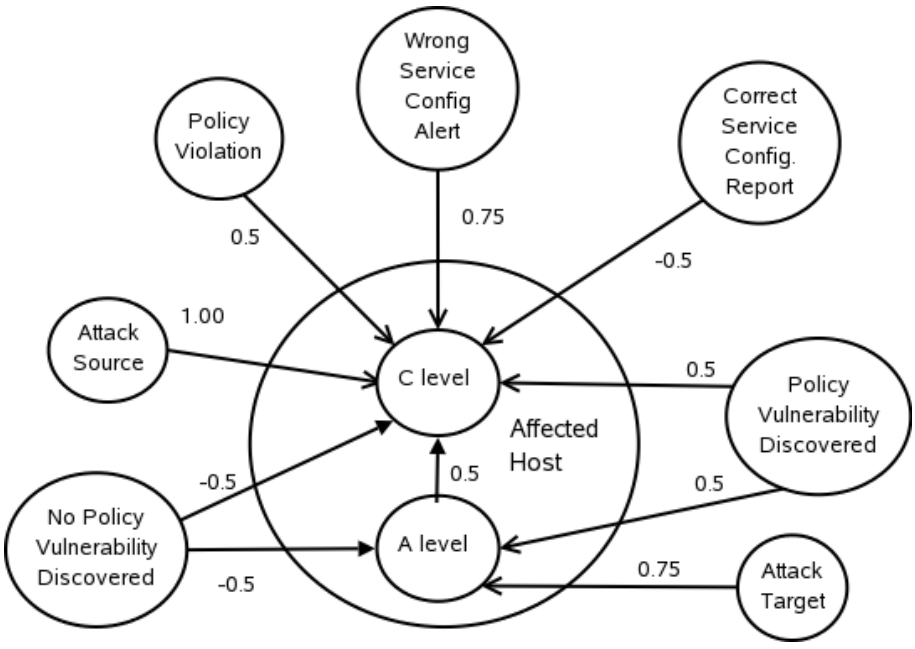


Figure 7: FCM for alert correlation

The logic behind the transitions in the FSM, as well as its effect on the FCM is as follows:

- A PVS alert (policy violation) indicates that the host is possibly compromised and it is being misused. The host will move into a state of *Misuse*. Actions that can get triggered

upon arrival to this state are a vulnerability and a configuration check. If policy violations accumulate, the state will change to mitigation. On the correlation process, each policy violation affects the C level of a host because its misuse can be a signal of compromise.

- An `IDS` alert indicates that there has been an incident. This increases the host's A level if it is the target, its C level if it is the source, and both the target's and source's C level if the attack is reported as successful. Upon arrival, a message is sent to the framework. If the C level is high enough, it will go into the mitigation state.

- A `PSC` alert (vulnerability) indicates that the host has been found to have a vulnerability. This moves the host into a vulnerable state. Upon arrival to this state, a configuration check will take place. If found to be misconfigured, it will transition to that state. In case of any other type of alert, the transition is to that corresponding state. A vulnerability alert increases a host's A and C levels, since its more probable that this vulnerability can be exploited to attack and/or compromise it.

- A `PCC` alert (configuration) indicates that the configuration found for any host's service differs from the expected one. That is, the configuration defined through the framework. This could be an indication of malevolous or accidental activity that potentially allows for vulnerabilities. The action to be taken is to prompt the framework for an immediate reconfiguration of the affected service. Upon reception of this alert, the C level of the host increases, since it is possible that it was changed by an attacker.

The system is capable of auto-stabilizing itself. A host's C level decreases with reports about correct configuration and vulnerability absence. Time also plays an important role in the stabilization. If after a given time frame there have not been any further reports for a host, its C level in the `FCM` will start decreasing in a negative exponential fashion. When this level is low enough, it will transition back to "idle" in the `FSM`.

Continuing with the dynamic fusion process shown in Figure 6, for the association stage the clustering method remains to be explained. Alert clustering will attempt to discover existing associations of each alert with identical alert clusters. Since there is a "similarity" to account for, a *concept* or *generalization hierarchy* is used to organize commonalities in a tree structure. It is considered that the root of any subtree is more abstract than any of its descendants. Therefore the root provides the general properties that all its descendants share.

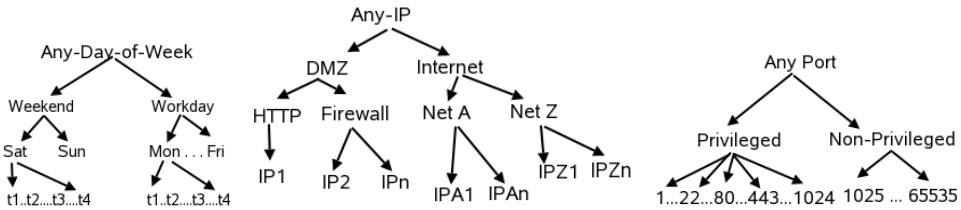
Ambareen [1] makes use of Julisch's [14] method for root cause analysis. The implemented algorithm is a variation of the Attribute Oriented Induction (AOI) proposed by Han [10].

Within the `PSM-A` the chosen attributes used to cluster alerts are: time, IP addresses, and ports used. Their generalization structures are shown in Figure 8. Multi-level alert clustering is performed by combining alert features and their abstractions. Alerts that match with lower levels of generalization receive higher scores.

When an alert for a target is considered to be included into a cluster, it is assigned a *candidacy score* reflecting its membership degree. This score is assigned as in Table 2, and it can vary between 3 (a very general similarity at level 1) and 12 (all attributes are the same at level 4). This value is then mapped into a fuzzy variable with normalized range $[0,1]$ and aggregated using the Mamdani[2] model. Afterwards a crisp candidacy score is obtained by a centroid defuzzyfication.

An alert can only belong to one cluster. Once the alert has been clustered, it is necessary to determine the *strength* of the relationships in all clusters. This indicates the similarity in the

[2]Fuzzy inference model that derives a conclusion from multiple rules through super-imposition



(a) Time Taxonomy (b) IP Address Taxonomy (c) Port Taxonomy

Figure 8: Attribute Taxonomy

Levels of Generalization		IP Address	IP Port	Time
General	1	Any-IP	Any-Port	Within-Day
	2	Network	(Non-)Privileged	Within-Hours
	3	Subnet	Actual port	Within-Minutes
Specific	4	Host		timestamp

Table 2: Domain Generalization Paths with Similarity Scores to be used in PSM-A

features of the alerts making up a cluster. This value is an average of the candidacy scores of all alerts constituting a cluster at each similarity level. Then, the combined impact of all clusters on the affected resource is calculated in the Cluster Association Strength (CAS). This calculation averages for a resource R_i all cluster strength levels $S(C_j)$ and their impact I_{ji}, based on the generalization level each cluster presents at time t_n. This calculation is shown in Equation 1.

$$CAS(R_i)_{t_{n+1}} = \left[\frac{\sum_{j=1}^{n} S(C_j)_{t_n} * I_{ji}(t_n)}{\sum_{j=1}^{n} I_{ji}(t_n)} \right] \tag{1}$$

When both the CAS and the C level are known, an assessment is to be made. Both crisp values enter the assessment process. They will be transformed into possibility distributions by fuzzyfying them. Once fuzzyfied, they represent a Degree of Concern (DOC). Both DOC are fused using an operator that behaves differently, depending on whether there is total agreement, total disagreement, or a weak or strong measure of conflict, or consensus, among them. The measure of consensus is given by intersecting both DOC and calculating $1-h$ where h is the height of their intersection. h is referred to as *consensus degree* and is calculated for two possibility distributions ($\Pi_i(\omega)$ and $\Pi_c(\omega)$) as in Equation 2.

$$[htbp]h(\Pi_i(\omega),\Pi_c(\omega)) = sup_{\omega\in\Omega}(min(\Pi_i(\omega),\Pi_c(\omega))) \tag{2}$$

For partial agreement the sensitivity is raised through the Hamacher Sum, whereas for partial disagreement a compromising behavior is used in form of the mean operator. Partial agreement is achieved when both distributions are smaller than h. Partial disagreement occurs when any of them is bigger than h.

Finally the Overall Degree of Concern (ODOC) is calculated as a crisp output. Centroid defuzzyfication is used because all contributions of the distributed data are considered. It calculates the weighted average of a fuzzy set. It is expressed as in Equation 3.

$$ODOC = \frac{\sum_i \Pi_o(\omega_i) \times x_i}{\sum_i \Pi_o(\omega_i)} \tag{3}$$

This value is a quantitative measure of the overall security situation assessment. The greater this value is, the more severe the overall security situation for this host is.

When the ODOC for all affected hosts is known, it is possible to calculate the Network Degree of Concern (NDOC). This is done as a weighted average. The weight used for each ODOC, corresponds to the C.I.A. values it has defined in the SDL. The NDOC is then used as an indicator for the overall health of the network. It has 5 possible values: severe, high, elevated, moderate, and low. They split evenly the percentage range $[0, 100]$ in steps of 20%. Depending on the number of defined security levels in the SPL, a transition from one level to the superior or inferior one can accompanied by a change in security policies.

5 Conclusions and future work

The assessment method presented in this article combines different sensor types to determine the individual health of a system. These sensors are used in a complementary fashion, so that what one of them reports can be checked by another. This allows to obtain confirming or denying evidence about a host's state.

By prioritizing incoming alerts, those assets with a higher strategic value are first taken to determine their individual health and take some action to correct the problem or inform the administrator. Individual health values are then used to calculate the network's overall health.

The network overall health value is then categorized by a five-step scale. This has two advantages: it allows for automatic changes in the policies, according to preset rules, and provides in just a single value a measure of network "sickness". This can help an administrator take manual action when the automatic policy changes are not enough to counter a situation. By also providing individual health values for each system, the attention can be directed to those systems that are in a worse state.

The framework presented in this article is a work in progress that is expected to be finished in the second half of 2007. It is currently in the integration phase. It will be tested in three different environments for compliance: academic, industrial, and governmental. When integrated and deployed all of POSITIF's characteristics will be tested. It will help in deploying and securing a network infrastructure.

References

[1] Siraj Ambareen. *A Unified Alert Fusion Model For Intelligent Analysis of SsensorData in an Intrusion Detection Environment*. PhD thesis, Department of Computer Science and Engineering, Mississippi State University, Mississippi, USA, August 2006.

[2] Siraj Ambareen, Rayford Vaughn, and Susan Bridges. Decision making for network health assessment in an intelligent intrusion detection system architecture. *International Journal of Information Technology & Decision Making (IJITDM)*, 3(2):281–306, 2004.

[3] Argus Open Project. network audit record generation and utilization system. http://qosient.com/argus/.

[4] Matt Bishop. *Computer Security: Art and Science*. Addison Wesley, March 2003.

[5] Yao-Min Chen and Yanyan Yang. Policy management for network-based intrusion detection and prevention. In *IEEE Network Operations and Management Symposium*, 2004.

[6] Dinesh C.Verma. Simplifying network administration using policy-based management. *IEEE Network*, 16(2), March/April 2002.

[7] Dmtf. http://www.dmtf.org.

[8] DMTF. Common information model (CIM) standard. http://www.dmtf.org/standards/cim/.

[9] Till Dörges and Klaus P. Kossakowski. Proactive security monitoring in a policy managed network. In *18th Annual FIRST Conference*, 2006.

[10] Jiawei Han, Yandong Cai, and Nick Cercone. Data-driven discovery of quantitative rules in relational databases. In *Transactions on Knowledge and Data Engineering*, volume 5, pages 29–40, 1993.

[11] IETF-IDWG. Intrusion detection message exchange format. http://www.ietf.org/rfc/rfc4765.txt.

[12] Policy core information model. http://www.ietf.org/html.charters/OLD/policy-charter.html.

[13] Intrusion Lab. raprelude. http://www.intrusion-lab.net/raprelude/.

[14] Klaus Julisch. *Using Root Cause Analysis to Handle Intrusion Detection Alarms*. PhD thesis, Fachbereich Informatik, University Dortmund, Germany, 2003.

[15] Sullivan KJ, Knight KC, Du X, and Geist S. Information survivability control systems. In *Proceedings of the 21st International Conference on Software Engineering*, pages 184–193, May 1999.

[16] Bart Kosko. Fuzzy cognitive maps. *Man-Machine Studies*, 24:65–75, 1986.

[17] R. C. Linger, N. R. Mead, and H. F. Lipson. Requirements definition for survivable network systems. In *Third International Conference on Requirements Engineering (ICRE'98)*, 1988.

[18] Leonidas Lymberopoulos, Emil Lupu, and Morris Sloman1. An adaptive policy-based framework for network services management. *Journal of Network and Systems Management*, 11(3), september 2003.

[19] Domenico Cecchini Marco Castaldi. A reconfiguration based intrusion response system. 2004.

[20] OSSIM – Open Source Security Information Management. http://www.ossim.net/.

[21] Gregorio Martínez Pérez, Félix J. García Clemente, and Antonio F. Gómez Skarmeta. *Web and Information Security*, chapter Policy-Based Management of Web and Information Systems Security: an Emerging Technology. IRM Press, 2006.

[22] POSITIF. Policy-based Security Tools and Framework. http://www.positif.org/.

[23] POSITIF Consortium, Universidad de Murcia. Positif project framework. http://positif.dif.um.es/index.html.

[24] Snort – network intrusion prevention and detection system. http://www.snort.org.

[25] Victoria Stavridou, Bruno Dutertre, and R. A. Riemenschneider. Intrusion tolerant software architectures. May 2001.

[26] Fredrik Valeur, Giovanni Vigna, Christopher Krügel, and Richard Kemmerer. A comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on Dependable and Secure Computing*, 1(3):146–169, July-September 2004.