Macrothink Institute™

Network Protocols and Algorithms
ISSN 1943-3581
2010, Vol. 2, No. 3

Achieving Security by Intrusion-Tolerance Based on Event Correlation

Massimo Ficco

Laboratorio ITeM “Carlo Savy”,

Consorzio Interuniversitario Nazionale per l'Informatica (CINI),

Via Cinthia - Edificio 1, 80126 Napoli, IT

Dipartimento per le Tecnologie,

Università degli Studi di Napoli “Parthenope”,

Centro Direzionale di Napoli, IT

E-mail: massimo.ficco@consorzio-cini.it

Received: June 22, 2010 Accepted: August 31, 2010 DOI: 10.5296/npa.v2i3.420

Abstract

Despite the increased focus on security, complex networked systems remain vulnerable to attacks. Intrusion Tolerance is an emerging paradigm for developing systems, which continue to operate correctly, and provide acceptable services even in the face of an intrusion. The effectiveness of this approach is strongly dependent on the efficiency of the adopted detection and diagnosis mechanisms. In this work, we propose an architectural framework, which collects information at several architectural levels, using multiple security probes, which are deployed as a distributed architecture, to perform event correlation and diagnosis analysis of intrusion symptoms. The experimental results show that the use of different security information sources can improve the detection and the diagnosis of attacks.

Keywords: correlation; detection; diagnosis; reaction

1. Introduction

Traditional approaches for building secure systems mainly focus on avoiding attacks to be successful. Such approaches are becoming insufficient when used in the context of complex networked systems, which are characterized by vulnerabilities that are impossible to identify and correct all before they are put in operation [1].

In the last years, Intrusion Tolerance (IT) has emerged as a new paradigm for developing networked systems. It aims to ensure that the considered system provides correct services in the presence of malicious intentional actions, which may lead to violates of the system security properties. IT is the ability of react, counteract, recover, mask errors, which may lend to failures if nothing is done to counter their effect on the system state. Unfortunately, the effectiveness of the IT approach is strongly dependent on the efficiency of both *Intrusion Detection* and *Intrusion Diagnosis*.

Nowadays, Intrusion Detection Systems (IDSs) are the main solution to protect networked systems. Regrettably, products which are currently available do not directly detect intrusions, but only the attacks symptoms, *i.e.*, the erroneous or anomalous states in a system component. Moreover, several works have observed that IDSs can generate thousands of alerts per days, up to 99% of which are false positives that make very difficult to identify the real attacks in progress over the system [2][3]. Essentially, neither diagnosis is performed in order to help the administrator to identify whether an alarm is a false positive or not, nor information are provided about consequences of the detected attacks [4].

On the other hand, clues of what is happening in real-time in the system are available, but spread out all over the network and the system components. For instance, many application servers write warnings and failures of the operations into their logs, all operating systems can log system and security events, and at the entry and exit points of the network, firewalls can track the packets that drop the security rules. Therefore, since an attack typically leaves different signs of its presence, correlating information found in multiple heterogeneous logs in near real-time, many false alerts (*i.e.*, false positives), and attacks that are not evident by analyzing a single log, should be detected. Moreover, diversity and correlation capabilities potentially improves diagnosis performance [5][6] through the aggregation of different views of the same incident retrieved by sources of diverse nature that are physically located in different locations within the system [7].

In this paper, we propose an architectural framework, which exploits an approach based on diversity both in information sources and detection methods used to monitor malicious activities. It collects streams of information at several architectural levels (*i.e.*, network, operating system, and application), using multiple security probes, which are deployed as a distributed architecture. Both anomaly-based and misuse-based detection methods are adopted to processing security information produced by probes [8]. The triggered events are correlated and rearranged based on their confidence level, which indicates the likelihood that correlated events are symptomatic of an ongoing attack on the system.

Detect different symptoms of the same attack allows to improve the intrusion diagnosis capability to assess of the damage in individual system components, which can be used to infer the least costly and most effective error treatment actions in order to avoid intrusions from generating a system failure.

Our preliminary experiments focus on the Distributed Denial of Service (DDoS) attacks. DDoSs have been one of the most frequent problems on the Internet. Currently, it is still a serious threat for mission and business critical systems [9]. The recent tide of DDoS attacks against high-profile web systems demonstrate how devastating DDoS attacks are [10]. Current IDSs, and especially the ones based on anomaly detection are not very efficient at detecting DDoSs. Indeed, it is hard to distinguish DDoS from traffic that presents marked legitimate variations. The experimental tests have shown that the proposed approach results in a better performance of the IDS, in terms of increasing detection capacity, as well as reducing the attack detection latency.

The rest of the paper is organized as follows. Section 2 provides an overview of the solutions to intrusion tolerance and event correlation, and discusses their main limitations. Section 3 presents the architecture and operation of the adopted architecture. Section 4 describes the proposed approach. Section 5 provides a description of the case study, and presents preliminary experimental results. Finally, Section 6 gives some concluding remarks, together with information concerning our future work in this field.

2. Related work

Even if IT notion is not a new concept, real research programs dedicated to this topic are relatively recent. MAFTIA [1] was the first project that uniformly applied the tolerance paradigm to the dependability of complex critical system. Its major innovation was a comprehensive approach for tolerating both accidental faults and malicious attacks. In accordance with the MAFTIA results, we agree that IT and Intrusion Detection are two tightly linked topics, but we claim that also on-line intrusion diagnosis is a key building block of IT. In order to operate the correct reaction to current intrusion, we propose an architecture that presents both diagnosis and detection properties. Moreover, in our work, the intrusion detection process is not implemented through the use of the combination of classical IDSs, but through a correlation approach based on diversity both in information sources and detection methods.

Saidom et *al.* [11] propose a generic IT architecture for Web services based on adaptive redundancy and diversification principles. The redundancy level is selected according to the current alert level. On the other hand, they do not propose any mechanism to estimate and assign weights to the raised alerts according to the credibility of each error detection mechanism. We address this problem in this paper.

Majarczyk et *al.* [12] propose an IT architecture based on COTS diversification applied to Web applications. The authors adopt an approach for anomaly detection using redundancy and diversification techniques. Although, the proposed solution provides a high coverage of detection, and a low level of false positives, no diagnostic function is defined in the system.

In order to improve the attack detection rate, enrich the semantics of alerts, and reduce the overall number of false alerts, different work propose explicit alarms correlation approaches [7], [13], [14]. In particular, [15] proposes a correlation workflow intended to unify the various steps of the correlation process. They adopt an approach that combines events that represent the independent detection of the same attack occurrence by different probes. Our work is strongly inspired by the this framework, but we proposed a weight-based correlation approach, in which each monitored symptom is weighted on the base of the trustworthiness of probe to monitor a specific attack.

3. Architecture and operation

Understanding the security status of the monitored networked complex system needs the correlation of observations performed on different system components, and across several entities distributed within the network.

In Figure 1, the architecture of the proposed framework is shown. It is characterized by several logical entities hierarchically organized: *Probes*, *Agents*, *Decision Engine*, *Remediators*, and *Monitors*.

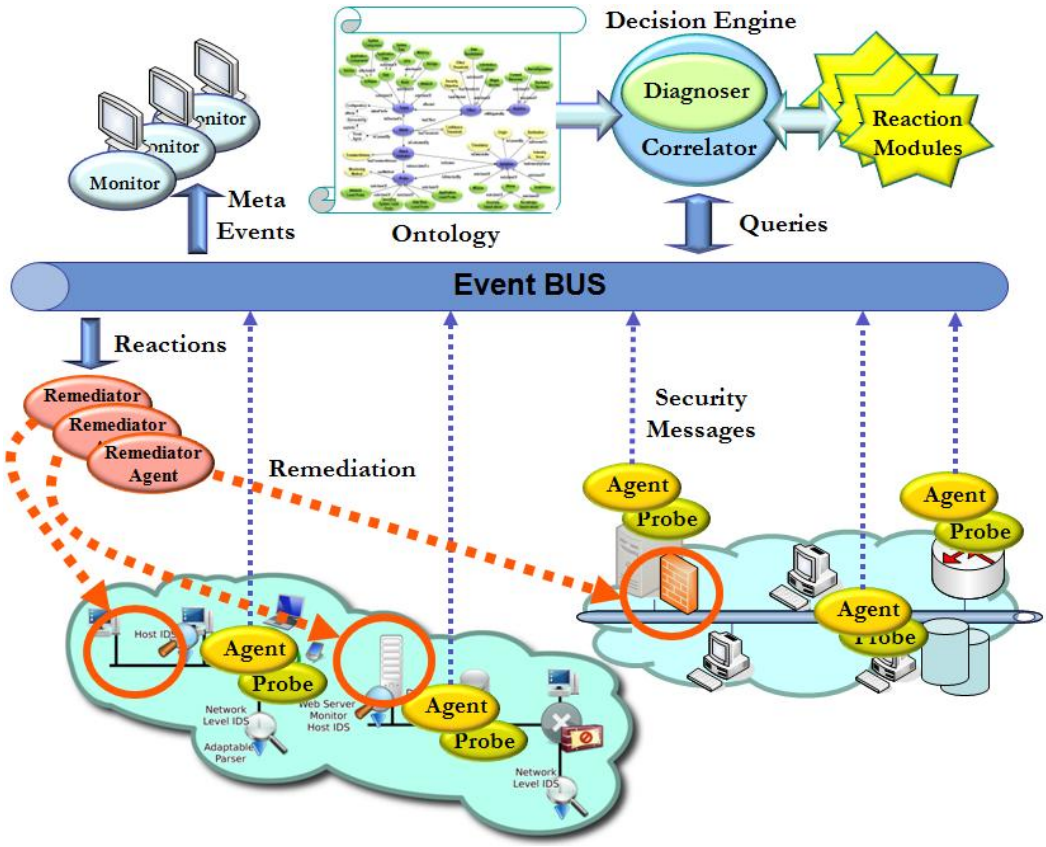

Figure 1. The proposed architectural framework.

- ***Probes*** : They are entities deployed in strategic points in the system (*i.e.*, network, host), and at different system architectural levels (*e.g.*, network, operating system, application), which ensure the diversity with respect to information sources. They infer attack symptoms from real-time traffic and specific logs. Each probe monitors specific features of a single node or a network segment. For example, a network IDS monitors anomalous packets rate in order to discover if an intruder is attempting to cause a denial of service attack, a host monitor measures CPU utilization, user logins, disk activity, and so on.

  Each probe can assess if an anomalous activity is underway on the base of its local view, and send detailed security messages to nearby Agent. They can adopt different anomaly-based detection methods (both anomaly-based and misuse-based), which triggers different security message formats.

- ***Agents*** : They are autonomous software components that analyze, filter, and forward probe messages to the Decision Engine. Each Agent relies on a knowledge base to convert raw security data into attributes usable by the Decision Engine (*e.g.*, timestamps, probe identifier, source and destination of the anomalous action).

  To decouple Decision Engine from the specific format of the Probe messages, Agents perform a normalization process, that enables different kind of Probes to generate messages using an unique language. In this work we use a standard representation of symptoms based on the IDMEF data model.

- ***Decision Engine*** : It performs both detection and diagnosis activities. It allows to recognize what attack can be hypothesized to be the cause of the monitored symptoms, and what parts of the system are supposed to be the targets of such attack. The enabling components are the *Correlator*, the *Diagnoser*, and the *Reaction Modules*.

  The Correlator aggregates continuous events in real-time. It uses a correlation schema defined by the ontology (not presented in this work) to infer a set of queries to be performed on the incoming messages streams. For each monitored attack type it extracts the corresponding monitored symptoms, and forwards them to the Diagnoser.

  The Diagnoser performs a ranking and filtering process of the produced event, in order to determine if the aggregated symptoms represent a real attack in progress in the system. Moreover, it identifies what part of the system is target of the attack.

  Finally, a security alert is forwarded to the Reaction Module, which is able to assess the attack effects on the target component. The information produced during this activity are used to instigate a remediation.

- ***Remediators*** : They are components designed to remediate to a specific attack or intrusion. In particular, they receive control information from the Reaction Modules concerning a particular fault treatment, and perform the associated actions designed to react to the attack (*e.g.*, network reconfiguration, revoke permits access).

- ***Monitors*** : They are central analysis servers, which receive data from a Decision Engine. They provide a Web interface that allows: *(i)* to see the current attacks status; *(ii)* to make interactive querying of attack data for analysis, and *(iii)* to identify attack patterns.

Finally, we adopt an open source framework, named Prelude [16], which acts as an event bus. It provides common features and standard API to enable IDMEF communication among the different involved entities.

4. Reaction process

The proposed solution emphasizes the relation among symptoms detection, intrusion diagnosis and reaction. In particular, at every time, it provides a synoptic view showing *(i)* what kind of attack is hypothesized to be the cause of monitored symptoms, *(ii)* what part of the system is under attack, and *(iii)* what are the attack effects. The results of this process can be used to active a reaction in order to mitigate the effects of the detected attack. The detection process is based on the correlation of symptoms monitored by multiple probes spread over the system.

Alerts are not generated as results of all the monitored symptoms, but only when a confidence assessment of correlated symptoms indicates a really potential attack, hence reducing the number of false positives, and improving the detection capability of the overall system.

As previously described, the Decision Engine receives the monitored symptoms by the Agents, and tries to correlate them by using correlation rules. In this work, we consider a correlation rule that aggregates symptoms based on the attack type, the target component and the temporal proximity. For each monitored target, an ontology identifies the symptoms to correlate for each potential attack associated with that target. The temporal proximity correlation is based on a time window *Tc* that is fixed as a parameter by the administrator. However, this approach requires that all clocks at probes be synchronized, *e.g.*, by using a total ordering based on timestamps, or a finer-grained mechanism [17]. At the end of this phase, for each target a meta-event $E(k) = \{e_{A1}(k), ..., e_{Am}(k)\}$ is triggered. For each possible attack A_i, $e_{Ai}(k)$ contains the symptoms correlated during the time window *k*.

In order to reduce the number of false positives generated by probes, we adopted an approach based on the *confidence* (C) of the events. Assuming that $e_{Ai}(k) = \{s_1, ..., s_z\}$ is the set of correlated symptoms related with the attack of type A_i during the time window *k*, the confidence is the likelihood that the monitored symptoms represent an underlining attack of such a type.

$C_{Ai}(k)$ is the sum of several terms, one for each symptom. Each term is characterized by a weights w_p and an *Intensity Scores* (IS). The IS is a probability value related to each monitored symptom. It is estimated by method used by probe to monitor the symptom. It reflects the likelihood that the observed symptom represents a malicious behavior. Since, the observed features are not commensurable, ISs are normalized to zero mean and unit variance. $w_p(s)$ is associated with trustworthiness of the probe p to monitor a symptom of the attack A_i. It is assigned on the base of a prior knowledge of the effectiveness of monitoring method being used for the given attack type. Although simple in implementation, choosing proper weights is of critical importance to highlighting the proper features under various attacks. The events ordered on the base of the C_{Ai} given an indication of most likely cause of detected anomalous behaviors at the step k. Such events are subjected to a filtering. If the confidence does not exceed a threshold (specific of each attack) estimated during a training phase, the event will be discarded (*i.e.*, it is considered as a false positive).

Finally, a compact diagnosis report is built and provided to both the system security administrator and the Reaction Module. For each produced meta-event, the report contains a summary of the information produced during the previous phases, including the likely cause analysis of the detected anomalous behaviors (*i.e.*, the attack A_i), and the target T_i. Based on the results of this diagnosis, the security administrator can: *(i)* determine whether the error processing mechanism is appropriate (*e.g.*, because the observed attack is not successful); *(ii)* prepare an adequate remediation action (*e.g.*, by system reconfiguration or by removing a vulnerability); *(iii)* improve the quality of the error detection (*e.g.*, to reconfigure the IDS by tuning of an anomaly detection threshold); and *(iv)* find out who/what performed the attack.

During the reaction phase, for each target component identified in the diagnosis phase, the degree of success of the intruder in terms of damage or corruption is assessed. In particular, it is verified whether attack effects are present on the considered target (*e.g.*, the memory consumption exceeds a fixed threshold). If an effect is identified, the Reaction Module determines the less costly and most effective remediation based on the severity of the error affecting the system component. It identifies the remediation action by using a grid of correspondence that associates the possible fault treatment with the observed effect.

During this process, the state of the target component can be: TRUSTED, TRUSTWORTHY, SUSPECTED, or CORRUPTED. Figure 2 describes the evolution of the state according to the received events.

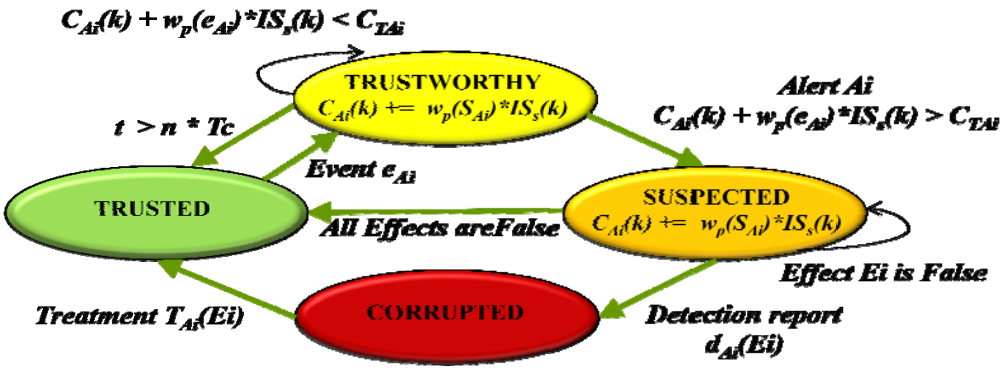

Figure 2. Different states of the target component.

When the Diagnoser receives an event from the Correlator, it can process or ignore this event according to its confidence. In particular, when an event e_{Ai} is received, the target component state is sets to TRUSTWORTHY, and remains in this state until the confidence level exceeds the confidence threshold C_{Ai} fixed during a training phase or a timer expires. If the timer expires, the security event is considered a false positive and the state returns to TRUSTED. If the confidence threshold C_{TAi} is exceeded, the SUSPECTED state is reached and an alert is generated. In this state, the Reaction Module decides reaction to operate based on the severity of the intrusion. Using the ontology schema, all the possible effects of detected attack A_i are verified. If a corrupted state of the component is found, the CORRUPTED state is reached, otherwise the state is set to TRUSTED. The resulting diagnosis data are sent to the Remediator, which performs the reaction to restore the TRUSTED state of the component.

5. Case study and experimental results

In this section we present preliminary results achieved by applying the proposed approach in a laboratory experimental setup. The experiments show that the use of different information collected at several architectural levels, using multiple security probes, allows to improve the detection and diagnosis capabilities. We consider Joombla (v.1.5), a well-known open source content management system written in PHP. It runs on an Apache v1.3.34 web server, and uses PHP v4.4.2 and MySQL v4.1.11.

This work focuses on Distributed Denial of Service (DDoS), which is a distributed attack on a computer system or a network, that causes a loss of service to legitimate users typically through the consumption of the victim resources, including bandwidth, CPU usage, memory, and disk space. It aims at overwhelming a target server with an immense volume of useless traffic from distributed and coordinated attack sources. In particular, SYN Flood attack is the most popular and effective brute-force DoS attack [18]. SYN flood exploits vulnerability of the TCP three-way handshake. During SYN flood, an attacker sends a lot of TCP SYN packets with spoofed IP addresses. Then, the victim returns an acknowledgement (ACK) to the attacker, but it never receives a reply back, so the connection is not established fully. This kind of connection is called *half-connection*. Since the number of malicious requests is large, the system runs out of resources, and starts the dropping of normal connection requests.

Figure 3 shows the effects of an DDoS attack on our web server (Pentium IV 2.4MHz with Mandrake 10.1). In particular, since experiments with SYN flood on commercial platforms show that the minimum flooding rate to overwhelm an unprotected server is 500 SYN packets per second [19], we injected attack traffic for the duration of 10 minutes with five agents. The flooding rate used by each considered agent to overwhelm the server is 120 SYN packets per second. The performed attack requests so many connections that after about 3 seconds, the target machine memory is completely exhausted by allocating data structures for half-open TCP connections. When the maximum number of half-open connections is reached, the target operating system discards all new incoming connection requests. The server does not accept new connection until either the handshake is completed (an ACK message is received), or until the connection establishment timer fixed by the operating system expires. In our scenario, the timer is set to about 400 seconds. When the timer expires, memory structures associated with the connection are reallocated, and the server accepts new connections. On the other hand, since the attack is already in progress, the system resources are again exhausted by allocating data structures for new half-open connections.

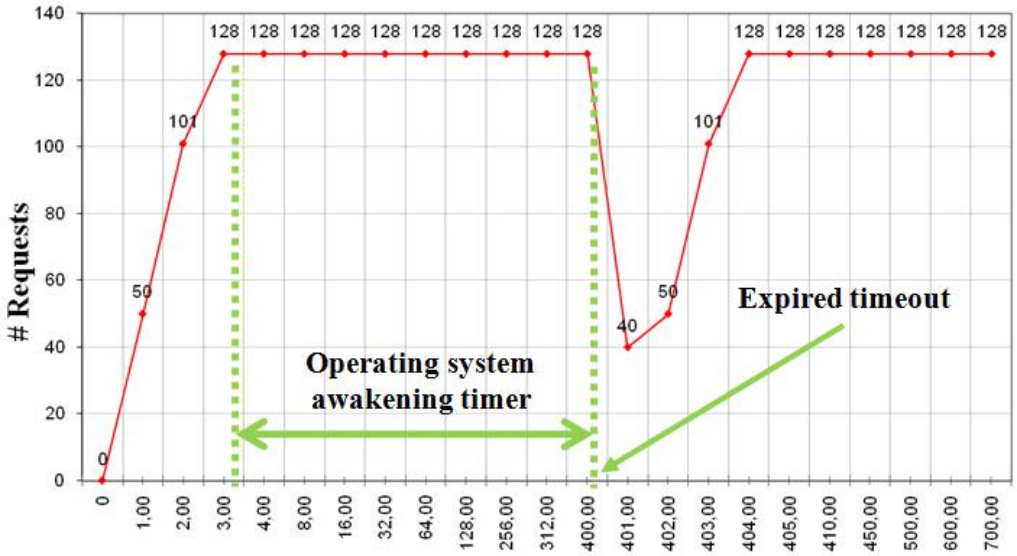

Figure 3. System under DDoS attack.

5.1 Detection methods

In order to monitor the attack symptoms, we adopt different probes, which use both anomaly detection and misuse detection methods. By using anomaly detection methods, intensity score values (ISs) are assigned to the triggered events, which reflect the anomaly levels with regard to an established profile. For each observed feature, the anomaly detection approach can perform in one of two phases: *training*, and *testing*. In the training phase, data sets are used to parameterize the monitoring method (necessary to determine the characteristics of '*normal*' behavior), as well as to establish the threshold, that is used to distinguish between regular and anomalous behaviors during the testing phase. In the testing phase, anomaly detection models are used to monitor anomaly behaviors with respect to

normal profile computed during the training phase. The threshold choice is the main problem in this process. It is a trade-off process between the number of false positives and the expected detection accuracy. A low threshold can result in many false positives, whereas a high threshold can result in many false negatives. Once the profiles and thresholds have been derived, the testing phase is operated. If the computed IS exceeds the fixed threshold a message is reported. For misuse detection method the IS value is fixed to 1.

We briefly describe the considered features, and the models adopted to perform the monitoring methods.

- ***Memory Consumption (MC)*** : Operating systems use different data structures of memory allocation (named *backlog queues*) for TCP connections establishment. There is a limit on the number of concurrent TCP connections that can be in a half-connection state. This limit is related with the length of the backlog queue. SYN flood can request so many connections that the target machine backlog queue can be completely exhausted.

  We adopt an anomaly-based model that estimates an approximation of the actual distribution of the backlog length, and monitors abnormal increasing of the queue with respect to normal profile. In particular, during the training phase we estimated an empirical mean and variance of the real queue length distribution. In order to estimate the observed queue length *l*, we used a *Chebyshev* function described by Equation 1, where *x* is a random variable.

$$p\ (|x-\mu| \geq k*\sigma) \leq \frac{1}{k^2} \qquad with \qquad k=\frac{|l-\mu|}{\sigma} \qquad (1)$$

  Replacing $k*\sigma$ with the distance $|l - \mu|$ is possible to obtain an upper bound *p(l)* on the probability that the attribute length deviates more from the mean than the current instance.

  Assuming that the training data set is attack free traffic, during the testing phase Equation 2 can be used to determine the deviation of the observed query attribute length from the normal behavior.

$$p(l)=\frac{\sigma^2}{(l-\mu)^2} \qquad (2)$$

  IS is computed as one minus *1/p(l)*. Moreover, we choose the threshold that does not produce false alerts with training data sets.

- ***Application Requests (AR)*** : The number of application requests can be used to detect anomalous behaviors. Typically, the average number of such requests do not vary much during the same period of the day. This behavior may change when SYN flood occurs (*i.e.*, the number of requests can dramatically decrease).

We adopted an anomaly detection model to capture the abnormal rate of application requests with respect to '*normal*' profile. During the training phase, for each week day, and for different period of the day, the relative frequencies of requests are computed. During the detection phase a "*chi-square*" function is adopted to identify anomalous behaviors. During a slicing time window *T*, if the rate of requests decreases below the threshold estimated during the training phase, the behavior is marked as anomalous and an event is triggered. *T* is a fairly short period of time (few minutes), since DDoS are usually attempted in a burst of half-connection requests.

The *chi-square* (X^2) values is estimated by using Equation 3.

$$X^2 = \sum_{i=1}^{n} \frac{(Oi - Ei)^2}{Ei} \qquad (3)$$

where O_i represents the observed value, whereas E_i is the expected value. Using the X^2 and the well-known lookup table [20], we compute the probability *p*. The IS is equal to one minus probability *p*. A probability *p* close to zero indicates an anomalous behavior that should yield a high IS. Considering that the training traffic is representative of the actual traffic, we choose the threshold that does not produce false alerts with the training data sets. In particular, we fix this threshold to the highest anomaly score seen during the training phase. During the detection phase the X^2 function is adopted to identify anomalous behaviors. If the anomaly score exceeds the threshold an event is triggered.

- ***Pattern Recognition (PR)*** : During SYN Flood the attacker exploits multiple agents to generate attack on its behalf. For each agent attacker uses many random source ports to connect to a single destination port of the victim. Therefore, the PR is a misuse method that computes the number of ports with the same IP source within a time slicing window, and controls that this value does not exceed a fixed threshold. Pukkawanna et *al*. [21] proposed a threshold for PR method equal to 10,000 source ports per IP with an analysis interval of two minutes. In our experimentation we found that threshold of about 8,500 source ports is sufficient to monitor SYN flood attacks.

5.2 Experimental evaluations

For monitoring the attack symptoms, we considered different probes (one for each adopted method) : *(i)* an Apache log analyzer examines the rate of the application requests, *(ii)* a system analyzer examines the backlog queue of the operating system, and *(iii)* a network protocol analyzer explores a specific signatures contained in the HTTP traffic respectively. During the training phase, in order to estimate the desiderate profile and the threshold for each anomaly method, we used real traffic collected from production servers at the university in which the considered application runs, during an interval time of one month.

During the testing phase, we performed four experiments. For each one, we injected one hundred attack instances as described in Section 5. We added real background traffic (different from that used during the training phase) on top of DDoS attacks traffic. The background traffic is collected from server during an interval time of one week. Then the attacks are injected randomly in the background traffic (about one attack per hour).

The experiments show, that using the estimated threshold, each method monitors correctly all the injected attacks. However, both anomaly-based methods present false positives. In particular, MC method triggers 22 false alarms, whereas AR triggers 76 false alarms.

Subsequently, results predicted by single models are compared to correlation process effects. During the correlation process the events produced by single models are aggregated, and the resulting meta-events are ordered based on their C_f . C_f is computed fixing w_m to 1 (for each model). As we expected, although false positives are still present, results show that almost all false positives provide a confidence below 43 (*e.g.*, false alerts that are not correlated with other events, or present low anomaly scores). A threshold value of 45 allows to reduce the false positives of about 99%, while leaving the number of detected attacks unaffected (only two attack instance are discarded).

During experiments, we observed that the use of multiple complementary detection methods allows also: *(i)* to improve the coverage of different attack strategies, and *(ii)* to reduce the attack latency, *i.e.*, the amount of time that an ongoing attack to the system has been undetected. For example, we observed that reducing the number of SYN packets per second injected by each agent below 70 source ports, the method PR does not monitor the attacks, although the attacks are successful. Only MC and AS monitor the malicious behaviors. A SYN attack can impact the server connectivity (*i.e.*, memory resources) without necessarily opening a great deal of ports source per agent. Regarding latency, as presented in Figure 3, the MC should be already in charge of monitoring the attack within few seconds (after 3 seconds the backlog queue is exhausted), whereas PR and AS require at last two minutes to detect the same attack.

We want to point out that the accuracy of the approach depends highly on setting appropriate thresholds and levels of confidence. However, the threshold for DoS attack may depend on many environmental factors, such as available bandwidth of current network, characteristic of DoS attack tools, number of generated attacks, duration of attack, operating system, and computer architecture of the target host. Therefore, the values used in this work may be specific for our experimental setup only. Administrators may need to adjust their values according to their operational environment to achieve high accuracy.

Finally, in order to determine the remediation to perform in case of attack successful, it is necessary to identify the possible effects on the target components. A possible solution for SYN flood may be based on an active monitoring approach. It consists in monitoring TCP/IP traffic and collecting communication control information to generate a view of all half-open connections on the target host. When a DoS alert is received, the Reaction Module can active a recovery action on the base of the host resources status. In particular, when the allocated

resources exceed a fixed severity level (estimated during a testing phase), the Reaction Module alerts a Remediation Agent, which sends packets to reset the half-open connections by generating the third message of the three-way handshake. The purpose of this action is to release the resources allocated at the target machine for connection establishments.

In the following, we show preliminary results of the considered recovery approach. In particular, we consider a scenario in which the attacker starts several processes that send many half-open connection requests with spoofed source addresses to the victim's target simultaneously. In Figure 4 is shown a simple example of remediation effect. When the Reaction Module detects the condition of a remediation action (the attack is in progress and the half-open established connections exceed the 95% of the maximum number of connection acceptable by the server), all the suspicious connections are reset. This kind of reaction allows to counter/mitigate the effects of the intrusion in order to ensure provision of the services (although in a degraded manner). Meanwhile, a reaction that requires more time can be undertaken. For instance, by using the information provided by the PR method, *i.e.*, the malicious detected IP sources, an upstream Access Control Lists or a routing reconfiguration can be performed.

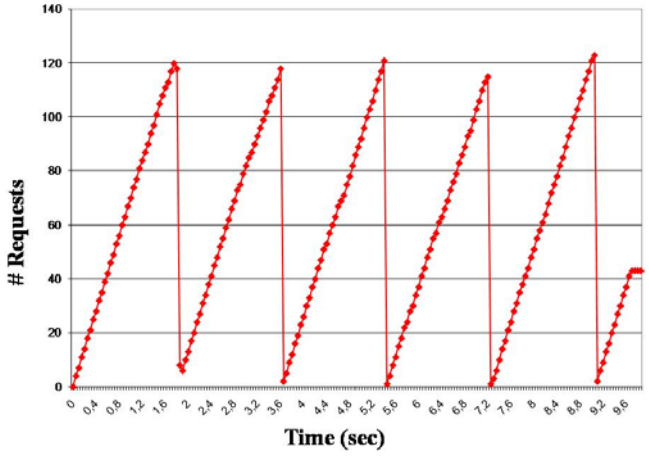

Figure 4. DDoS remediation effects.

6. Conclusions

Preliminary results presented in this paper show that the correlation of information collected at several architectural levels, using multiple security probes, allows to improve the detection and diagnosis capabilities, in order to infer the fastest and most effective error treatment actions.

Future work will follow two main directions. The first objective will be consider different types of DoS attacks (*e.g.*, application level floods, ICMP floods), which will be used to estimate the capability of the proposed solution to provide a detailed diagnosis of: *(i)* the attack that has produced the detected symptoms, *(ii)* the attack target, and *(iii)* the attack effects on individual system component. The second objective will be to conduct a thorough experimental analysis, in order to collect evidence of the effectiveness of the approach to reduce false positives on the base of the confidence values.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement no. 225553 (INSPIRE Project).

References

[1] Verissimo, P., Neves, N. F. , Correia, M., "Intrusion-tolerant architectures: Concepts and design", in Architecting Dependable Systems, LNCS 2677. Pp 23-36, Springer-Verlag. May 2003.

[2] Axelsson, S., "The base-rate fallacy and the difficulty of intrusion detection", in ACM Transactions on Information and System Security (TISSEC). Vol. 3, Issue 3. Pp. 186-205. August 2000. http://dx.doi.org/10.1145/357830.357849

[3] Manganaris, S., Christensen, M., and Hermiz, K., "A data mining analysis of RTID alarms", in IEEE Computer Network. Vol. 34. Issue 4. Pp. 571-577. October 2000. http://dx.doi.org/10.1016/S1389-1286(00)00138-9

[4] Majorczyk, F., Totel, E., Mé, L., and Saïdane, A. "Anomaly Detection with Diagnosis in Diversified Systems using Information Flow Graphs", in IFIP International Federation for Information Processing, LNCS. Vol. 278. Pp. 301-315. Springer Boston 2008. http://dx.doi.org/10.1007/978-0-387-09699-5_20

[5] Al-Mamory, S, and Zhang, H., "Intrusion detection alarms reduction using root cause analysis and clustering", in ACM Computer Communications. Vol. 32. Issue 2. Pp. 419-430. February 2009. http://dx.doi.org/10.1016/j.comcom.2008.11.012

[6] Julisch, K., "Clustering intrusion detection alarms to support root cause analysis", in ACM Transactions on Information and System Security (TISSEC). Vol. 6. Issue 4. Pp. 443-471. November 2003. http://dx.doi.org/10.1145/950191.950192

[7] Yu, D., and Frincke, D., "Alert Confidence Fusion in Intrusion Detection Systems with Extended Dempster-Shafer Theory", 43rd ACM Southeast Regional Conference. Vol. 2. Pp. 142-147. September 2005. http://dx.doi.org/10.1145/1167253.1167289

[8] Hervé, D., Dacier, M., Wespi, A., "Towards a taxonomy of intrusion-detection systems", in Computer Networks: The International Journal of Computer and Telecommunications Networking. Vol.9. Pp. 805-822. April 1999.

[9] Xu, J., and Le, W., "Sustaining availability of Web services under distributed denial of service attacks", in IEEE Transactions on Computers. Vol. 52. Issue 2. Pp. 195-208. February 2003. http://dx.doi.org/10.1109/TC.2003.1176986

[10] Witter, Facebook fend off DoS attacks, at: http://www.securityfocus.com/brief/992. Published: Aug. 2009.

[11] A. Saidane, V. Nicomette, and Y. Deswarte, "The design of a generic intrusion-tolerant architecture for web servers", IEEE Transactions on Dependable and Secure Computing. Vol. 6. Issue 1. Pp. 45-58. January 2009. http://dx.doi.org/10.1109/10.1109/TDSC.2008.1

[12] Majorczyk, F., Totel, and L. Me´., L., "COTS Diversity Based Intrusion Detection and Application to Web Servers", in Proceeding of the 8th International Symposium on Recent Advances in Intrusion Detection (RAID '05). Pp. 43-62. Sept. 2005.

[13] Haibin, M., and Jian, G., "Intrusion Alert Correlation based on D-S Evidence Theory",

2th IEEE International Conference on Communications and Networking. Pp. 377-381. August 2007. http://dx.doi.org/10.1109/CHINACOM.2007.4469406

[14] Morin, B., and Debar, H., "Correlation of Intrusion Symptoms: an Application of Chronicles", in Proceeding of the 6th International Conference on Recent Advances in Intrusion Detection (RAID'03). September 2003.

[15] Valeur, F., Vigna, G., Kemmerer, A., "A Comprehensive Approach to Intrusion Detection Alert Correlation", in IEEE Transactions on Dependable and Secure Computing. Vol. 1. Issue 3. Pp. 146-169. July 2004. http://dx.doi.org/10.1109/TDSC.2004.21

[16] Prelude, a hybrid ID system. Available at: http://www.prelude-ids.com.

[17] Bondavalli, A., Ceccarelli, A., Falai, L., "Assuring Resilient Time Synchronization", in Proceeding of the IEEE Symposium on Reliable Distributed Systems (SRDS'08). Pp. 3-12. Oct. 2008. http://dx.doi.org/10.1109/SRDS.2008.12

[18] Gordon, L., and et al., "CSI/FBI Computer Crime and Security Survey". Computer Security Institute. Available at: http://i.cmpnet.com/gocsi/db_area/pdfs/fbi/FBI2004.pdf (2004).

[19] Darmohray, T., and Oliver, R., "Hot Spares for DoS attacks". Available at: http://www.usenix.org/publications/login/2000-7/apropos.html. November 2000.

[20] The Chi-Square Probabilities Table. Available at http://people.richland.edu/james/lecture/m170/tbl-chi.html. December 2008.

[21] Pukkawanna, S., Visoottiviseth, V., and Pongpaibool, P., "Lightweight Detection of DoS Attacks", in Proceeding of the 15th IEEE International Conference on Networks. Pp. 77-82. November 2007. http://dx.doi.org/10.1109/ICON.2007.4444065

Copyright Disclaimer