CSIT (December 2013) 1(4):317–329
DOI 10.1007/s40012-013-0029-x

ORIGINAL RESEARCH

A dynamic firewall architecture based on multi-source analysis

Muraleedharan Navarikuth · Subramanian Neelakantan · Kalpana Sachan · Uday Pratap Singh · Rahul Kumar · Antashree Mallick

Received: 15 December 2012 / Accepted: 8 October 2013 / Published online: 7 November 2013

Abstract Firewall forms the first layer of defense mechanism with respect to perimeter security in which organizational security policies are implemented as rules. Incoming and outgoing packets are screened to prevent violations against firewall rules. There are two fundamental issues that we observe in the present day firewall. Firstly, present day firewall is not intelligent enough to be aware of the breaches and attacks happening over the network. Second issue is that the formulated rules are not permanent and needs frequent firewall rule updates which are carried out manually. Also, present day attacks are more sophisticated and dynamic in nature. Hence, static nature of the present day firewall do not provide adequate protection to the dynamic network needs. In this paper, we present a comprehensive approach, whereby we carry out analysis from multiple sources to generate dynamic firewall rules. Uniqueness of our approach are

– in aggregating multiple inputs such as intrusion detection system alerts, SNMP events and flow records,
– devising individual analyzers based on statistical techniques such as K–S test and entropy based for generating dynamic firewall rules.

To validate our approach, we carried out attack and accuracy analysis over network test environment. The results show that our approach is effective in generating new firewall rules especially for scan and flood type of attacks.

Keywords Reactive firewall · Dynamic firewall · Self-configuration · Active firewall · Automatic firewall rule-formulation · Adaptive firewall · Firewall analyzers · Anomaly detection

M. Navarikuth (✉) · S. Neelakantan · K. Sachan · U. P. Singh · R. Kumar · A. Mallick
Centre for Development of Advanced Computing (C-DAC), Bangalore, India
e-mail: haimuru@gmail.com; murali@cdac.in

S. Neelakantan
e-mail: subbu@cdac.in

K. Sachan
e-mail: kalpana@cdac.in

U. P. Singh
e-mail: udaypratap@cdac.in

R. Kumar
e-mail: rahul@cdac.in

A. Mallick
e-mail: antashree@cdac.in

1 Introduction

Firewall forms the first layer of defense mechanism with respect to perimeter security. Firewalls are configured to block unauthorized accesses, while at the same time permitting authorized communications, based upon a set of rules/policies. They examine each packet that is coming inside to the network as well as outgoing packets, check against a set of rules configured and take decision to permit or deny the packet. Traditional firewall implementations are categorized under three major techniques viz., packet filtering, application proxies and state-full packet filtering. Ingham and Forrest [13] present a detailed survey of network firewalls and also present the challenges faced by firewalls.

1.1 Present day firewalls challenges

Present day firewalls are configured manually and the rules/policies stay static until new policies are enacted.

Also, the configuration of present day firewall becomes a cumbersome activity for the administrators, considering the evolution of new applications, the users and the usage of ports for their execution. Static firewall has limitations such as

- it does not keep track of the attacker's information,
- it does not take into account the actions inferred through other defense mechanisms.

We observe two fundamental issues in present day firewall:

- *Attack-based prevention* present day firewall implements rules to realize organization policies but lacks capabilities to keep track of attacks in network and to prevent the same.
- *Automatic rule formulation* firewall rules are not changed frequently based on need, however such firewall rule updates are formulated manually in present-day firewall.

1.1.1 Attack based prevention

Present-day firewall rules are formed based on organizational security policies which is usually about allowing or denying access based on application, host, network addresses and content inspection. Such rules do not essentially prevent all kinds of attacks that may happen in a network. For example, attacks such as scan or flood could still be possible within the allowed ranges. These days such attacks are quite dynamic and change their characteristics which is not detected by the firewall. Hence a firewall that understands attacks and keeps track of the same to take steps for prevention is required. This capability is lacking in present-day firewalls. In our proposed dynamic firewall, we incorporate such capabilities to the present day firewalls to be more vigilant and prevents attacks as well.

1.1.2 Automatic-rule-formulation

The implemented firewall rules are not permanent instead, as and when required, firewall rules are modified by the security administrator. However, this is carried out manually which is not effective. In continuation of the issue discussed above about attack-based-prevention, it required to have a system which automatically formulate new firewall rules as an outcome of attack analysis. Related area of research that is gaining momentum is to evolve self-configuring systems towards aimed at minimizing operational efforts and manual interventions by automating configuration and decision-making capabilities.

1.2 Our approach

In this paper, we present a comprehensive approach to address these specific challenges and generate firewall rules automatically. We achieve this by keeping track of behaviors in the network, identifying serious violations and by performing event correlation of various security events. For this, we utilize three sources of inputs viz., intrusion detection system (IDS) alerts, flow data and SNMP event in a network. IDS alerts are interesting security events that provide pointers of ongoing malicious activities. Hence firstly, we carry out analysis of IDS events to prune and obtain interesting ongoing security events. Secondly, we use the flow records of network traffic and by applying entropy based estimations, detect abnormal events. Thirdly, we apply Kolmogorov–Smirnov (KS) test over SNMP events and obtain anomalous events from the same.

We devise individual analyzers like IDS analyzer, flow analyzer and SNMP analyzer for obtaining interesting security events. We then carry out event correlation between these events based on time, host addresses, etc., to obtain the new set of firewall rules. For example, let us consider a scenario where there are huge number of IDS alerts for a particular host and at the same time period there are also anomalous events identified by flow analysis. Now when we correlate these IDS and flow analyzer events alone, it is clear that malicious activities are aimed at that particular host. Subsequently, we formulate firewall rules to prevent traffic to and from that particular host. Similarly, correlation between the output of IDS analyzer and SNMP analyzer may reveal such interesting attacks.

The main contributions of this paper are (a) In utilizing three different source of inputs to understand network attacks, (b) Individual analyzers of IDS, flow and SNMP for generating security events and (c) carrying out event correlation of analyzer events to formulate firewall rules. Rest of the paper is arranged as follows. Section 2 brings out the related work, Sect. 3 presents the architecture for adaptive firewall and brings out the internals of the components in detail. Section 4 brings out the implementation and analysis. Results and result analysis presented in Sects. 5 and 6 concludes the paper with future directions.

2 Related work

Since this paper focuses on firewall rule generation, based on anomaly detection and event correlation, we carry out survey of related work and research specifically into traffic analysis, event correlation and firewall rule-optimization and prioritisation. A traffic aware firewall optimization techniques is explained in [1]. In this work, an optimum rule set for

firewall which reflects the current characteristics of the traffic without violating the semantic integrity of the initial rule set is produced using the current traffic log. Our approach also uses the current traffic information for generating dynamic rule but unlike their approach, it uses multiple input sources and correlating those events for generating dynamic rules.

An architecture-aware adaptive deployment of contextual security policies are presented in [25]. They propose a solution for deploying a security policy in systems. This work is more related to deployment of the security policies and not to generate new rules based on the traffic trend. Another traffic aware firewall approximation algorithm for selecting the top-N most frequently matched subset of rules from the original rule set is presented in [16]. In this paper, reordering of the existing rules based on the traffic are considered but not generating any new rule. Krueger et al. [15] proposed an intelligent mangling technique for HTTP protocol, based on the decision of previously trained anomaly detectors to replace suspicious parts in requests of HTTP protocol. Compared to our approach, the target of attack are different in this work. It applies anomaly detection techniques for HTTP related attacks.

Castiglione et al. [3] presents an enhanced firewall scheme for containment of emerging security threats. This active firewall architecture is based on the dynamic and adaptive policy management by facilitating new rules and policies to the firewall. In this architecture they used data mining techniques for analyzing the traffic traces and network and system logs. The input and analysis for generating the new rules by our system is different from this approach. Moreover, we considered a multi-source events generated for the same traffic and correlating those events to achieve high accuracy for rule generation.

There are recent efforts towards research in the area of event correlation. Real-time monitoring and analysis of events from multiple network security devices is presented in [22]. Cross-correlation from multiple sources is performed based on a set of rules. Zhang and Zhang [30] analyzes the events from multiple sources like firewalls, IDSs and Vulnerability Assessment Systems and propose an event correlation system. This approach unifies the alerts from multiple sources, removes the redundant alerts and generates rules utilizing data mining. However, their approach has constraints related to compression-based and filter-based approaches and do not provide attention for specific attacks.

In the work of Zhaojun et al. [9] propose a security event correlation algorithm for multi-source analysis based on similarity of the attributes. The focus is to find out a similarity relationship among the security events for reducing the false alarms generated by multiple sources. Weights are being assigned for each element of the event for obtaining such a similarity relationship. Kang and Na [14] uses a rule based approach for event correlation. The rules describe an attack scenario which primarily targets the multistage attacks. Correlation is performed on the physical and logical system events that are deployed in an enterprise. In this approach, rule requires manual definitions of malicious behavior. Only predefined multistage attack patterns can be identified by this approach. Liu et al. [20] describes architecture for event correlation for middleware based applications. Their approach is sequential-based correlation and limited to some correlation patterns. Griffith et al. [8] addresses event correlation from multiple systems using temporal patterns. Subramanian et al. [28] propose an event correlation techniques for IDS alerts to reduce the false alarms.

Performance enhancement of the firewall based on the arrangement of the policies or rules are another key area of related work that are being carried out now. Salah et al. [26] presents an embedded Markov chain based analytical queueing model to study and analyze the performance of rule based firewall. They conclude that the position of the rule in the rule set can have an impact on the performance of the firewall under denial of service (DoS) type of attacks. The rule set is dynamically rearranged such that the bottom rules can be served at the top of the rule set to minimize the impact of DoS type of attack in firewalls performance. This paper deals with the dynamic arrangement of existing rules to minimize the impact of DoS type of attack. But in our approach, it generates dynamic rules and add to the firewall instead of analyzing the existing rules in the firewall. Hwang et al. [12] proposes a systematic structural testing approach for testing the firewall policies. The focus of this work is to helps the firewall to ensure the correctness of the policies. Nurika et al. [23] discusses the different models of deployment of firewall which speedup the filtering process and improve the precision of rules, etc. Similarly [18] presents the theory and algorithm for firewall policy change impact analysis for addition, deletion, modification and swapping of the firewall rules. These techniques help to analyze the impact of a firewall rules before committing that rule into the firewall rule list.

In addition to the above specific works, there are other interesting research related to firewall such as Liu and Gouda [19] proposes a diverse firewall design method. This method can be used to compare two firewall and functional discrepancies between them and impact analysis of the firewall rules. Gouda and Liu [7] proposes a structured firewall design method to address the issues related to the consistency, completeness and compactness of the firewall rules. An anomaly management framework to identify and resolve policy anomalies in firewall presented in [11]. Uribe et al. presents an integrated, constraint-based

approach for modeling and reasoning about the network IDS (NIDS) and firewalls. This will process firewall rules, and analyze abstract NIDS configurations to determine whether a detection policy is enforced or not [29]. An adaptive method which utilizes the traffic characteristics coupled with the analysis of policy is explained in [4]. Our approach integrates alerts generated from three different sources: IDS, flow and SNMP and based on high severity alerts new rules will be formed.

3 Dynamic firewall architecture

Dynamic firewall architecture is presented in Fig. 1. It consists of the following major components

- *Agents* to collect inputs from different input sources such as IDS, SNMP and flow.
- *Dynamic-rule-generator* to carry out individual analysis of multi-source input and to carry out event correlation to generate new firewall rules.
- *Basic firewall capability* which accepts/rejects traffic based on firewall rules.

Of these, agents along with analyzers and event correlation are carried out near real-time whereas the basic firewall capability is carried out in real-time. The near real-time methods work as control-plane and the outcome that is

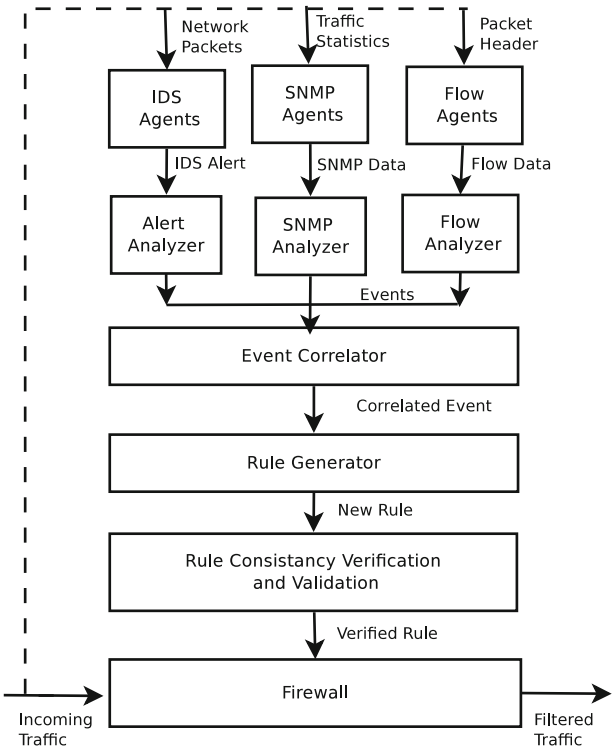

Fig. 1 Architecture diagram

Table 1 Multi-source input

Input sources	Analysis	Detection capability
IDS	Packets	Known attacks in which signatures are available
SNMP	Interface statistics	Anomalous events such as DoS, DDoS, flood
Flow	Traffic summary	Anomalous events such as DoS, DDoS, scan, worm spread

generated new firewall rules are used to update the real-time firewall rule.

The system take inputs from external components such as IDS, SNMP agents and flow generator. As part of this system, different analyzers are provided to handle inputs given from external entities. These analyzers, after analysis, generates alerts and these alerts are then correlated. The system generates new firewall rules from the correlated events. These rules are used to update the firewall rules.

3.1 Input agents

It will be difficult to achieve high accuracy using single input data source because the data can have only limited parameters in it. The usage of multiple sources of related inputs data can be very useful to confirm events across analyzers. Hence we considered three sources of inputs related to network traffic in different aspects such as IDS alerts (IDS inspects the packets and generates the alerts for known attacks), flow data (provides connection details and network anomalies) and SNMP events (provides network interface statistics to detect network anomalies). The reasons of choosing these inputs are that they offer key pointers to formulate new firewall rules as given below:

- *IDS* IDS alerts notify malicious incidents that happen in a network. Carrying out analysis such as off-line analysis, real-time analysis, alert pruning, etc. shall bring out critical events and attack trends.
- *SNMP* this is an active network data collection approach that presents the interface statistics using which we can identify the anomalies.
- *Flow* this is a passive network data collection approach that presents the statistics of applications, hosts, etc. using which we can identify anomalies.

Table 1 shows summary of different source of input and their detection capabilities.

3.1.1 IDS

A NIDS inspects network traffic to detect attacks and notify it as alerts. Based on the detection techniques, IDS can be

classified as signature based and anomaly based IDS. Signature based IDS inspects network traffic for detecting the occurrence of well-known attack pattern called signatures whereas anomaly based IDS detects abnormal behavior in a network based on network traffic [6].

Alerts generated by an IDS usually contain elements such as

- *Time stamp* provides the time of detected attack,
- *Attacker IP and port* gives details about the attacker,
- *Victim IP and port* gives detail of victim and targeted application,
- *Severity* provides the severity of detected attacks,
- *Category* the attack category such as buffer overflow, remote access, scanning, etc.

In the following description A shows the set of all IDS alerts and C is the set of all critical alerts. H and V shows the set of attacker and victim hosts for critical alerts, respectively.

$$A = \{a_1, a_2, a_3, \ldots, a_n\},$$
$$C = \{A \mid c \in A \quad \text{and} \quad \text{criticality} = \text{high}\},$$
$$H = \{C \mid h \in C \quad \text{where } h \text{ is an attacker IP}\},$$
$$V = \{C \mid v \in C \quad \text{where} \quad v \quad \text{is a victim IP}\}.$$

To carry out IDS alert analysis, we have defined following functions

$$T_n\{H\} = \{\text{Top } n \quad \text{attacker}\}, \tag{1}$$

$$T_n\{V\} = \{\ \text{Top} \quad n \quad \text{victim}\}, \tag{2}$$

$$A_{as}V = \{H \cap V\}, \tag{3}$$

$$A_{many}V = \{T_1\{H\} \quad \text{to different victim}\}, \tag{4}$$

$$A_{same}V = \{T_1\{V\} \quad \text{from different attacker}\}, \tag{5}$$

where (1) provides top attacker list and all the anomalous traffic from this machine can be blocked. Similarly (2) shows top victim and all the anomalous traffic to this machine can be blocked. Equation (3) shows machines which act as both attacker and victim. Attack from one attacker to many victim and many attacker to one victim are shown in (4) and (5), respectively. These machines can be added in the list of blocked IPs.

3.1.2 SNMP analyzer

SNMP protocol provides information about network and hosts in the network. Periodic polling is performed using active network monitoring mechanism for collecting this information. The collected information given as an input for analysis, which identifies the changes in traffic pattern and hosts trend. Observing the changes in traffic pattern is useful when attacks performed such as DoS or flood, which lead to a significant change in the pattern.

Table 2 SNMP parameter

Parameters	Normal behavior	Attack behavior
In octets	Avg	High
In uinicast pkts	Avg	High
In non unicast pkts	Low	High
Out octets	Avg	High
Out unicast pkts	Avg	High
Out non unicast pkts	Low	High

The KS test used in the intrusion detection areas to identify the anomalous activities [2, 5]. To detect the anomalies in the network traffic, we also applied a two-sample KS test on collected SNMP data. The two-sample KS test is used to test whether two samples come from the same distribution and it uses the maximal distance between cumulative frequency distributions of these two samples as the statistic. The empirical distributive function F_n for n observation X_i is defined as

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} IX_i \leq x,$$

where I is the indicator function, equal to 1 if $X_i \leq x$ and equal to 0 otherwise.

We used the following equation for two sample KS test

$$D_{n,n'} = sup_x \mid F_{1,n}(x) - F_{2,n}(x) \mid, \tag{6}$$

where sup_x is the supremum of the set of distances and $F_{1,n}$ and $F_{2,n'}$ are the empirical distribution functions of the first and the second samples, respectively. The null hypothesis that the sample X conforms to the distribution function F is rejected or accepted for a given level of significance α by looking it up in a table of values and by comparison with the statistic $D_{n,n'}$ obtained. The null hypothesis is rejected at level α if

$$D_{n,n'} > c(\alpha)\sqrt{\frac{n+n'}{nn'}}. \tag{7}$$

The value of $c(\alpha)$ is given in the table defined by KS test for each level of α.

To identify the normalcy of the network traffic using SNMP data, we profile six different parameters. The list of profiled parameters and their behavior in normal and attack scenario are shown in Table 2.

The SNMP analyzer works in two different phases namely learning phase and detection phase. In learning phase, the analyzer learns the normalcy and sets the threshold, and in detection phase, it identifies the deviation from threshold and notify the events.

During the learning period, we poll OID and prepare training data set. The entire learning phase is divided into polling period of same duration, e.g. if training time is

1 day and polling period is 20 s then the number of polling periods will be 4,320. Each polling period generate a row with six parameters in it. For testing data set, we replicate training data set to testing data set and modify it by replacing corresponding value with real time data during the detection phase.

The two sample KS test using training and testing data set is applied on two different data sets X_1 and X_2. The empirical distribution function is calculated using Eq. (6) for each data set and obtain KS statistics $D_{n,n'}$ where n and n' is number of rows in training and testing data sets, respectively. After obtaining $D_{n,n'}$ value, applied the null hypothesis using Eq. (7) and if it rejects null hypothesis, can identify that data set value as anomalous and generate an alert for respective value and its parameter.

3.1.3 Flow analyzer

Flow is defined as a set of IP packets passing an observation point in the network during a certain time interval. Unlike SNMP, flow data provides the network and host traffic statistics in a passive manner. Recently, due to lesser volume, flow data uses for security analysis and intrusion detection for high speed networks. Flow based intrusion detection can address the detection of attacks like DoS, network scan, flood, worm spread, botnet, etc. which changes the traffic pattern of network [27].

The proposed system collects the flow data from network and carries out analysis for identifying traffic intensive attacks such as flooding, DoS, scan and worm or bot-net spread. We have used unidirectional flow which contains different parameters such as time stamp, source and destination IP, port, protocol, number of packet transferred, number of byte transferred and Flag value in TCP packets. Once the system identifies the anomaly, it carries out analysis for identifying the attacker, victim IP, application and protocol. The alert generated from the flow analyzer consists of the details such as attack time, attacker and victim IP, attacker and victim port and protocol.

Entropy based methods have recently been used by researchers in the area of anomaly detection and traffic classification. An entropy-based anomaly detection method that identifies network anomalies by examining the characteristic traffic feature distributions is presented in [17]. This method is independent of network topology and traffic characteristics, and can be applied to monitor every type of network. Since entropy measures the randomness, it has been extensively used for anomaly detection purposes. A flow data based anomaly detection technique based on entropy measure is presented in [21, 24]. Our analyzer also uses entropy based technique using flow data to identify the anomalies and further to generate dynamic rules. The entropy of a random variable X is defined as

$$H(X) = -\sum_{i=1}^{N} p(x_i)\log_2(p(x_i)),$$

where $x_1,\ldots,x_N$ is the range of values for X and $p(x_i)$ represents the probability that X takes the value of x_i. Since entropy measures the randomness of a data set, it captures the degree of dispersal or concentration of distributions for traffic features. High entropy values signify a more dispersed probability distribution, while low entropy values denote concentration of a distribution. Entropy values range between 0 and $\log_2 N$. In order to have a metric independent of the number of distinct values of the data set, we normalize the entropy by dividing $H(X)$ with the maximum entropy value $\log_2 N$. The normalized entropy is given by the following equation and its values range in (0, 1).

$$H_n(x) = H(X)/H_{max}(X), \quad H_n(X) \in [0, 1]. \tag{8}$$

To detect anomalous network activities, we have calculated the entropy values of different parameters in flow data such as source and destination IP, source and destination port and size of the packet.

Table 3 Entropy values

Parameters	Normal entropy	Attack entropy			
		Scan	DoS	DDoS	Flood
Source IP	High	Low	Low	High	Low
Dest IP	High	Low	Low	Low	High
Source port	Low	High	High	High	–
Dest port	Low	High	Low	Low	–
Packet size	High	Low	Low	Low	Low

Table 3 shows the different parameters and variation in their entropy values during normal and attack traffic. During the normal network communication, the incoming traffic can be originated from many different sources so the entropy values of the source IP will be high. Similarly in normal traffic, the responses are distributed to different hosts causing high value in entropy. Since the users accessing limited number of applications such as web and mail, the entropy values of source and destination ports in normal traffic will be low. In normal traffic, we observed variation in the packet size and because of that entropy value of packet size will be high.

The changes in the entropy values of selected parameters during vertical scan traffic is shown in the table. In vertical scan, to know the status of port, attackers send packets to many ports of the targeted system. This will result many connection attempt to the targeted machines so

the entropy value of the destination IP will be low. Since all the traffic to the target machine dispersed to different ports, the entropy value of destination port will be high. Most of the scanning tools injects same size packets to different ports for scanning, then the entropy value of the packet size for scanned traffic also will be low.

Depend on the number of origins of scan traffic, the entropy value of source IP can be varied, i.e., if it is a distributed scanning attempt to a targeted machine, the entropy value of the source IP can be high. Similarly depend of the scan traffic origin port, the entropy of source port can be varied. If the scanning tool using the same source port for a scan traffic, the entropy value of source port will be low. To summarize this, destination IP, destination port and packet size values in the flow data can be used to detect scan attempt.

Unlike scan traffic, in DoS attack, all the traffic to the destination machine will be targeted to a single application. This will cause a very low entropy value in the destination port. In case of distributed DoS (DDoS) attack, since the traffic sources are distributed in many attacker machines, the entropy value of source IP will be high. During network flood, the traffic will target entire machines in the network or a range of IPs, and in such cases, entropy value of destination IP will be high and because of same size packets, the entropy value of packet size parameter will be low. All the variations of entropy values during DoS, DDoS and flood are shown in Table 3.

3.2 Event correlation

Event correlation is the process of correlating monitored events generated from multiple sources for further analysis. Multiple sources generate events independently, identification of the relationship between generated events is important to filter out the security event. The analyzers such as IDS, SNMP and flow generate alerts separately, the role of the correlator is to identify the common and vital events across this analyzer's outcome. The correlator collects the input from multiple sources and identifies the relationship between them.

The process event correlation becomes an essential component for dynamic rule generation as the dynamic rules are intended to achieve a specific target instantly. For the dynamic rule generation to be effective, the inference engine should be effective. Therefore, event correlation based on multiple sources becomes a convincing approach. The key challenge lies in event correlation is in providing the inference with substantial accuracy.

It should be noted that any attack need not occur as an instant event and mostly it is a sustained activity in a period of time window. Hence we fix a time window to find overlapping events across analyzers. This results in formulation of dynamic rules as we encounter serious security threats in a time window. It is possible to have many number of events from the same analyzer for a particular time window.

Let S be all the events generated by SNMP analyzer in a time window and it contains $s_1, s_2, s_3,\ldots,s_n$ where s_x is the xth event. Similarly F and I are set of events generated by flow and IDS module, respectively.

$$S = \{s_1, s_2, s_3, \ldots, s_n\},$$

$$F = \{f_1, f_2, f_3, \ldots, s_n\},$$

$$I = \{s_1, s_2, s_3, \ldots, s_n\}.$$

The structure of SNMP event is shown in (9) and it consists of time stamp of generated event, IP address of the host which created the anomaly and attack type such as scan, flood, DoS, etc. Similarly (10) and (9) shows the structure of generated flow and IDS event structure, respectively where *src_ip* and *dst_ip* are the source and destination IP address and *src_port* and *dst_port* are the source and destination port, at is the attack type.

$$s_x = \{ts, ip, at\}, \tag{9}$$

$$f_x = \{ts, src\_ip, dst\_ip, src\_port, dst\_port, at\}, \tag{10}$$

$$i_x = \{ts, src\_ip, dst\_ip, src\_port, dst\_port, at\}. \tag{11}$$

Since the attack can be a sustained activity, the analyzer may detect and notifies same event in multiple times and it causes to generate duplicate alerts, i.e., among $s_1, s_2,\ldots,s_n$ in S s_i, s_j may notify the same event. It is important to remove these duplicate alerts from same analyzer before the correlation. To remove the duplicate events from the same analyzer in a time window, we have applied the following formula where S_{ue} is the unique event generated by SNMP module F_{ue} and I_{ue} are the unique events from flow and IDS, respectively.

$$S_{ue} = \{s_1 \cup s_2 \cup s_3 \cup s_4 \ldots \cup s_n\}, \tag{12}$$

$$F_{ue} = \{f_1 \cup f_2 \cup f_3 \cup f_4 \ldots \cup f_n\}, \tag{13}$$

$$I_{ue} = \{i_1 \cup i_2 \cup i_3 \cup i_4 \ldots \cup i_n\}. \tag{14}$$

Once we remove all the duplicate events from each analyzer as shown in (12)–(14), we remove the time stamp entry from the resultant set S_{ue}, F_{ue} and I_{ue}. After removing the duplicate events and time stamp entry from analyzer output, we apply event correlation on those events across the analyzers. To identify the common events from different analyzer output, we used the following equation, where E_c is the set of correlated events and I_{ue}, S_{ue}, F_{ue} are the unique events from IDS, SNMP and flow analyzer for the same time window.

$$E_c = \{I_{ue} \cap S_{ue}\} \cup \{I_{ue} \cap F_{ue}\} \cup \{S_{ue} \cap F_{ue}\}. \tag{15}$$

From (15), we can ensures that at least two of the analyzers should detect and notify the same event for further action to

be taken. This helps in confirming the occurrence of a security threat with substantial accuracy. If the correlator could not find any common event in at least two of the analyzers output then $E_c = \varnothing$.

3.3 Rule generation

Rule generator formulates new firewall rules from correlated events by identifying the attack type and attackers. Depending on the severity and seriousness of the correlated events across analyzers, it can evolve a new firewall rule. Since the rule structure and syntax are changing from firewall to firewall, the rule generation module generates rules in a generic format.

$$R = \{src\_ip, dst\_ip, src\_port, dst\_port, action\}. \quad (16)$$

The format of the rule generated by rule generator is shown in (16) where *src_ip* and *dst_ip* are the source and destination IP address and *src_port* and *dst_port* are the source and destination port. The action field in the rule R indicate the action to be taken by the firewall once it detect and verify the attack. The action can be depend on the type of detected attack i.e., attack like scan, it can log the packet and in case of DoS, DDoS and flood attack, it can drop the packets coming from attacker. We have divided the firewall rule into two parts such as rule parameters and rule action. Rule parameters consist of the first four elements of the rule shown in (16) and rule action contains the action. The generated rule need not contains all the fields mentioned in (16) but at lest one field in the rule parameter and action field are mandatory for any valid rule. For example, instead of specific values such as 80 in the port number filed of the rule, it can be a generic value as '*' to represent all ports. Similarly to represent any machine we can use '*' in source/destination IP field of the rule.

If $E_c \neq \varnothing$ in (15), then the rule generator take the correlated output E_c and can generate rule from it. Since the generated event from SNMP analyzer contains only two fields other than time stamp in it, correlated output of SNMP analyzer with other two analyzers generate only two fields. The rule generation algorithm is shown in Algorithm 1 where R_p represent the parameter list of generated rules, S_{ue}, F_{ue}, I_{ue} are the unique events generated by SNMP, flow and IDS analyzer, respectively.

The rule generator maintains a list of attack and corresponding action to be taken by the firewall to mitigate it. Depending on the severity of attack and security policy, an administrator can configure these actions. For example, once the scanning activity is detected from a machine, then the firewall can log all the packets from that machine. But if it is a critical attack like DoS, then the action can be taken to drop the packets from that machine. To generate a rule with action, the system can lookup the action list and take the corresponding action for the detected attack and this action can be added to the generated rule parameter R_p.

input : Correlated event set, Lookup table consists of Attack type and action
output: Generated Rule R

if $(I_{ue} \cap S\_ue) \neq \emptyset$ **then**
  $R_p = \{I_{ue} \cup S_{ue}\}$
else if $(F_{ue} \cap S_{ue}) \neq \emptyset$ **then**
  $R_p = \{F_{ue} \cup S_{ue}\}$
else
  $R_p = \{F_{ue} \cup I_{ue}\}$
end
$R_a = \texttt{LookupAction}(attacktype)$;
$R = R_p \cup R_a$;

3.4 Re-configuration

Re-configuration is the process of verifying and adding the newly generated rules to the firewall when it is in operation. The input to this process is a set of validated rules that are ready to be re-configured. Rule generation process generates the actions to be taken by the firewall, where as re-configuration implements the actions in the firewall.

Rule generation process generates rules as and when the inferences obtained through analysis and correlation. It does not maintain a threshold on the number of rules being generated. This increases the number of firewall rules to be configured at any point in time. It is the job of re-configuration process to check the size of the rule set prior to the configuration and take steps to mitigate the rule set size. Also, rule set size makes an impact on the efficiency of the firewall rule search. If the number of rules is more, it increases the time taken by the search algorithm for matching a packet. Hence, steps are taken to reduce the size of the rule set to optimize the firewall rule search.

In view of the continuous and dynamic rule configuration, performance of the rule search becomes a key factor. Since the dynamic rule addition increases the responsibility of the rule search algorithm, reconfiguration should address the need for an efficient algorithm for rule search optimization. Re-configuration should also ensure the persistent addition of dynamic rules. As the rules get added when the firewall is in operation, the next firewall shutdown should keep up the dynamically configured rules. Dynamic rule configuration can cause a break in state-full firewall and established connections. It also needs to be addressed. Another aspects which we are not included as part of the scope of this work is that the optimization and redundancy verification of the firewall rules.

4 Experiment set-up

A test set-up for evaluating the proposed system is shown in Fig. 2. The firewall is connected in the gateway of the

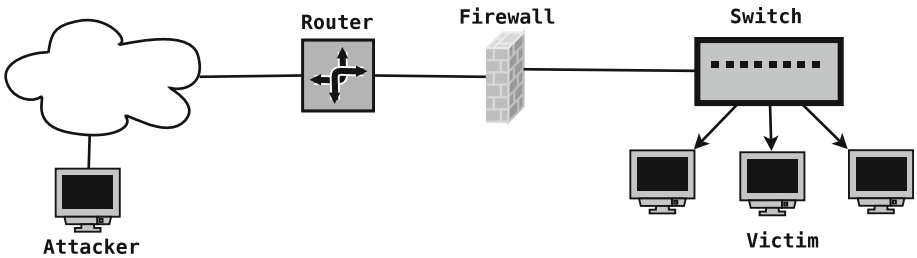

Fig. 2 Test set-up

network and can access all the incoming and outgoing traffic. To deploy the firewall, we used a machine based on Open BSD operating system and enabled the PF (packet filter) firewall in it [10]. We used the same machine to deploy IDS and SNMP analyzer. With the help of PF firewall state table, we enabled the flow probe in the PF firewall. To collect the generated flow data and further to analyze it, we deployed the 'flowd' analyzer in the same machine.

For collecting the SNMP data, instead of enabling the SNMP agent in each host, we enabled the SNMP agent in the switch. Each host is connected to a particular port in the switch, to collect the incoming and outgoing traffic to the host from the interface statistics of each port of the switch. We identified one machine which is connected to the switch as the victim. To test the detection and dynamic rule generation capabilities of the system, we carried out different types of attack to the victim machine from a machine which is outside to the firewall protected network.

The deployed IDS detects the attack using signature based detection and generated the alerts and same traffic is analyzed by SNMP and flow analyzer to detect the anomalous activities. Flow and SNMP analyzers generates alerts for anomalous activities and all the alerts are sent to the correlator for correlating the alerts for 2 min time window.

To detect the attacks and further to generate new rules for mitigating those attacks, we used SNMP and flow based anomaly detectors. Since the SNMP and flow analyzers relies on anomaly based techniques to identify the attacks, it depends on the training data set for identifying the threshold values. We have taken an assumption that there is no attack during the training period. To ensure the attack free training data set, our experiment and analysis are carried out over a controlled environment. Moreover, during the training period, we deployed network monitoring and intrusion detection tools to monitor the environment and ensured that the test data set contains only clean traffic in it. Once the training phase is over, SNMP analyzer identifies the threshold based on KS test and flow analyzer identifies the normal entropy values of the parameters.

5 Result and result analysis

This section brings out the obtained result of the experiment carried out on the test setup. The total learning period of SNMP and flow analyzer was 5 days. The polling period of SNMP data collector was 30 s and for each polling it requested for six different OID values. The SNMP data collector polled the 26 interfaces of the switch i.e., 26 different host statistics are profiled in the learning period. To test the detection capabilities of the system, we carried out attack such as network flooding, host scan and DoS. We configured the duration of the flooding attack for 10 min (600 s) and host scan is done on all the ports of the victim machine.

5.1 IDS alert analyzer

Other than processing the real time alerts and generates the combined alerts from it, we have given large set of generated alerts to the IDS alert analyzer to find out the common attacker and victim IP. Table 4 shows the result of IDS alert analysis. We grouped the total alerts into three different sets. First set (S_1) consists of 10,000 alerts, second set (S_2) consists of 30,000 alerts and third set (S_3) consists of 100,000 alerts.

Table 4 IDS alert analysis summary

Parameters	Alert data set		
	S_1	S_2	S_3
Total number of alerts	10,000	30,000	149,814
High severity alerts	1,041	3,261	18,887
Number of unique attackers	158	331	971
Number of unique victims	9	31	138
Number of alerts from top attacker	81	258	1,191
Number of distinct victim from top attacker	8	26	10,
Number of alerts targeted to top victim	960	2,804	11,437
Number of distinct attackers to top victim	157	307	769

Table 5 Conducted attack summary

Attack types	Number of attacks	SNMP analyzer		Flow analyzer		Correlated		
		Number of events	FP	Number of events	FP	Number of correlated events	FP	FN
Scan	57	136	46	57	0	57	0	0
Flood	53	258	205	103	50	51	0	2

Since the generated IDS alerts have different severity values such as low, medium and high, we considered only high severity alerts for further analysis. Because the high severity alerts shows the critical attack and it need to mitigate those attack by blocking the attack traffic. Third row of the table shows the number of unique attackers who have generated high severe alerts. In the given alert data set, minimum 5.14 % (data set S_3) and maximum 15.27 % (data set S_1) of attackers are unique attackers. In other way, in the given data set, if we block the traffic from 971 attackers, we can remove 18,887 alerts. Similarly fourth row of the table shows the analysis result of the generated alerts from the victim's perspective. Compared to the previous row, in the given IDS alert set, if we block the traffic to 138 victims we can remove 18,887 alerts. Details of the number of attack generated from the top attacker are shown in the fifth and sixth rows of the table. The fifth row shows the number of alerts generated from the top attacker and from this result we can identify that from a single attacker it was generated 1,191 alerts for S_3 data set. That is, if we add a single rule in the firewall to block the top attacker, it reduces the number of alerts.

From the IDS alert analysis results, it is clear that the IDS output can provide very useful information for dynamic rule generation. One of the basic use of this output is that it helps to create a list of blacklisted IP from the analyzed alerts. We can add the top attacker information into the list of black listed IP of firewall rules so that any further communication from that machine can be logged or blocked by the firewall. Since the black listed IP list is generating in a dynamic manner, the firewall can be respond to the attack which is not foreseen in the initial configuration of the firewall rule.

5.2 SNMP and flow analyzer

We enabled the SNMP agent in the switch and SNMP analyzer started in the firewall machine. Similarly the flow probe and flow analyzer started in the firewall machine. These two analyzers collect the corresponding inputs and analyze it independently and generate events for anomalous activities such as scan, flood and DoS attack. To test the detection capabilities of these modules, we conducted different type of attack to the victim machine from the attacker. The summary of injected attacks and detection capability of the proposed system is shown in Table 5.

As shown in the table, the first two columns show the type and number of attacks conducted. Next four columns show the SNMP and flow analyzer output such as number of alerts generated and false positive alerts. The correlation output includes correlated events, number of false positive and false negative events are shown in the last three columns.

To test the scan detection capability of the system, we conducted 57 different scanning activity to the victim machine. In each scan attack, we probed the entire ports in the victim machine and the delay between two consecutive scan activities are set as a 10 min. From the SNMP analyzer output we identified that it detected all the scan attack and it generated 136 events for the 57 scans conducted. This indicates that SNMP analyzer identified some of the non-scan activity as scan and generated events for that. From this, we can identify 79 false positive events from SNMP analyzer. The flow analyzer also detected all the scan activities and generated events for that. The total number of events generated by flow analyzer is 57 for 57 scan activities. This indicates that, the flow analyzer detected all the scan accurately and not generated any false events. When we apply the correlation algorithm on 136 event generated by SNMP and 57 flow events, it identified 57 common events across the analyzers. It shows that out of 57 scan attack conducted, the system identified 57 events and further to proceed for rule generation, i.e., the system detected all the scan activities accurately.

Last row of the table shows the results of flood attack. We conducted 53 different flood attacks and the duration of each flood is configured as 10 min. The SNMP analyzer detected all attacks and generated 258 events for the 53 flood attack. The number of false positive events from SNMP analyzer is 203. The flow analyzer also detected all flood events and generated 103 events which includes 50 false positive events as well.

The reason for generating more false positive events in case of flood attack is that, compared to the scan attack, the duration of flood attack was high (10 min). So the same attack may overlap to multiple polling period of SNMP and flow and the analyzers generate multiple events for that. From the correlation output, we can see that, it generated

Table 6 SNMP analyzer snapshot

Event details		IP	Threshold	
Time	Types		Learned	Detected
23:10:36	Flood	10.182.0.223	0.0356	0.3409
23:11:06	Flood	10.182.0.223	0.0356	0.3075
23:12:06	Flood	10.182.0.223	0.0356	0.2840
10:32:35	Scan	10.182.0.220	0.03556	0.0523
10:33:05	Scan	10.182.0.220	0.03556	0.0523
10:33:35	Scan	10.182.0.220	0.03556	0.0523
10:34:05	Scan	10.182.0.220	0.03556	0.0523

51 flood events. This indicates that, after event correlation, there are two flood events which cannot identified by the detection system but the system has not generated any false positive event.

Table 6 shows a 2 min window the snapshot of SNMP analyzer. It consists of the event details for 2 min window, detected attack type and IP address which generated the anomalies. Last two columns of the table show the threshold values calculated by the KS test at normal time and attack time, respectively. First three rows of the table shows events generated by flood attack and last four rows shows the events generated by scan attack. This indicates that SNMP analyzer detected the same flood attack three times and generated events for it. Similarly the same scan event identified four times by the analyzer.

Table 7 shows the profile (learning) time summary of the collected flows and entropy values of identified parameters. The duration of profile period was 432,000 s (5 days) and the export time of the flow set as 60 s. Normalized entropy values of the identified parameters are shown in last five columns of the table.

Table 8 shows a two minute window snapshot of the flow analyzer. It consists of the event details for 2 min window, detected attack type, attacker and victim IP and threshold values of five flow parameters. Events related to flood attack is shown in the first two rows of the table. We can verify that during the flood attack the entropy values of source and destination IP are less compared to normal entropy value. Because of more packets are coming from the same IP during flood, the entropy value will be less. Since many packets are generated from the attacker, the source port of that packet can be varied. This will create high entropy value for source port during flood. Since the flooding tool generates same packets for flooding, the entropy value of packet size parameter can be less.

As shown in the table, in a single time window, flow analyzer generated two events for same flood activity. Last row of the table shows the snapshot of scan activities. From the table, we can verify that during scanning, the entropy value of destination port will be high. The flow analyzer

Table 7 Flow profile output summary

Input details			Entropy values				
Total flows analyzed	Total packets	Total octets	SrcIP	DstIP	SrcPort	DstPort	PktSize
1,430,000	43,354,056	6,807,748,451	0.1060	0.0890	0.5027	0.4810	0.2349

Table 8 Flow analyzer snapshot

Event details		Detection details		Entropy values				
Time	Types	Attacker IP	Victim IP	SrcIP	DstIP	SrcPort	DstPort	PktSize
23:11:15	Flood	10.182.0.54	10.182.0.223	0.0296	0.0277	0.9517	0.0721	0.0109
23:11:45	Flood	10.182.0.54	10.182.0.223	0.0225	0.0183	0.9899	0.0327	0.0189
10:33:15	Scan	10.182.0.44	10.182.0.220	0.0411	0.0406	0.1138	0.9266	0.0049

Table 9 Event correlation snapshot

Received time	Event types	Detected analyzer	Attacker IP	Victim IP	Approved for correlation	Correlation output
23:12:06	Flood	SNMP	–	10.182.0.223		
23:11:45	Flood	Flow	10.182.0.54	10.182.0.223	Yes	Flood on 10.182.0.223 from 10.182.0.54
10:34:05	Scan	SNMP	–	10.182.0.220		
10:33:15	Scan	Flow	10.182.0.44	10.182.0.220	Yes	Scan on 10.182.0.220 from 10.182.0.44

identified the same scan activities two times. Multiple events in the same time window can be removed and the unique events are passed for correlation.

As described in the architecture, the output of SNMP and flow analyzer goes to the event correlator. The event correlator takes the generated events from all the analyzers and identifies the common event from it to generates new rule. So in the above mentioned snapshot, from the SNMP and flow analyzer, event correlator get total two events for flood attacks. The collected events are correlated based on time stamp, IP address and type of attack. The output of flood event correlation is shown in the first two rows of the table. Similarly, the correlated output of scan events are shown in the last two rows of the table. Table 9 shows the output of event correlator. Time of received events from different analyzer is shown in the first column, type of received event is shown in the next column and the analyzer which detected this event is shown in the third column of the table. Since the collected SNMP data doesn't contain any source IP information, the SNMP analyzer can provide only the information related to the affected machine by mapping the port number of the affected interface of the switch. But from the flow analyzer, we can identify both the attacker and victim IP. The sixth column of the table shows that whether the received event can be correlated based on the 2 min time window. If multiple events are present from different analyzer in the 2 min time window, then the event correlator can select those events for correlation. The last column of the table shows the output of event correlator. The correlated output can be used to generate the firewall rules. Since the syntax of the firewall rules are different for different firewall, the rule generator with the input of events, attacker and victim IP, generates rule according to the deployed firewall.

5.3 Effectiveness of the system

From Table 5, it is clear that the proposed system detect and prevent attacks like flood and scan. The usage of multi-source input helps the system to reduce the chances of false alarms. This is clearly brought out from the analyzers (SNMP and flow) output in the table. For 57 scan events, the SNMP analyzer generated 136 events and flow generated 57 events. If we used only one input source like SNMP, it creates more alerts. If the system generates rule for mitigating the identified attacks, then it may interrupt or block the genuine traffic. The flood attack detection shown in the last row of the table clearly shows the capability of the system to reduce false positive alerts. In that case, though both the analyzers generates number of false positive (205 for SNMP and 50 for flow) events, the final correlated events does not have any false positive in it.

6 Conclusion and future directions

In this paper, we presented an architecture for adaptive firewall system. We also presented the challenges involved in adaptive firewall, internals of the architecture in detail and the implementation and analysis carried out. Further, to this work, we wish to explore effective behavior based model for automatic rule generation for worm attack.

Acknowledgments This work is based on the ongoing research and development project funded by Department of Electronics and Information Technology (DietY), Government of India.

References

1. Acharya S, Wang J, Ge Z, Znati TF, Greenberg (2006) A traffic-aware firewall optimization strategies. In: IEEE international conference on communications, 2006. ICC'06, vol 5. IEEE, New York, pp 2225–2230
2. Caberera JBD, Ravichandran B, Mehra RK (2000) Statistical traffic modeling for network intrusion detection. In: Proceedings of the 8th international symposium on modeling, analysis and simulation of computer and telecommunication systems, 2000. IEEE, New York, pp 466–473
3. Castiglione A, De Santis A, Fiore U, Palmieri F (2010) An enhanced firewall scheme for dynamic and adaptive containment of emerging security threats. In: 2010 International conference on broadband, wireless computing, communication and applications (BWCCA). IEEE, New York, pp 475–481
4. El-Atawy A, Samak T, Al-Shaer E, Li H (2007) Using online traffic statistical matching for optimizing packet filtering performance. In: 26th IEEE international conference on computer communications, INFOCOM 2007. IEEE, New York, pp 866–874
5. Estévez-Tapiador JM, Garcıa-Teodoro P, Dıaz-Verdejo JE (2004) Measuring normality in http traffic for anomaly-based intrusion detection. Comput Netw 45(2):175–193
6. Garcia-Teodoro P, Diaz-Verdejo J, Maciá-Fernández G, Vázquez E (2009) Anomaly-based network intrusion detection: techniques, systems and challenges. Comput Secur 28(1):18–28
7. Gouda MG, Liu AX (2007) Structured firewall design. Comput Netw 51(4):1106–1120
8. Griffith R, Hellerstein J, Kaiser G, Diao Y (2006) Dynamic adaptation of temporal event correlation for QOS management in distributed systems. In: 14th IEEE international workshop on quality of service, 2006. IWQoS 2006. IEEE, New York, pp 290–294
9. Gu Z, Li Y (2011) Research of security event correlation based on attribute similarity. JDCTA Int J Digit Content Technol Appl 5(6):222–228
10. Hansteen PNM (2010) The book of PF: a no-nonsense guide to the OpenBSD firewall. No Starch Press, San Francisco
11. Hu H, Ahn GJ, Kulkarni K (2012) Detecting and resolving firewall policy anomalies. IEEE Trans Dependable Secur Comput 9(3):318–331
12. Hwang JH, Xie T, Chen F, Liu AX (2008) Systematic structural testing of firewall policies. In: IEEE symposium on reliable distributed systems, 2008. SRDS'08. IEEE, New York, pp 105–114
13. Ingham K, Forrest S (2002) A history and survey of network firewalls. Technical Report. University of New Mexico
14. Kang D, Na J (2012) A rule based event correlation approach for physical and logical security convergence. IJCSNS 12(1):28

15. Krueger T, Gehl C, Rieck K, Laskov P (2010) TokDoc: a self-healing web application firewall. In: ACM symposium on applied computing, SAC 2010, pp 1846–1853
16. Lam HY, Wang D, Chao HJ (2011) A traffic-aware top-n firewall approximation algorithm. In: 2011 IEEE conference on computer communications workshops (INFOCOM WKSHPS). IEEE, New York, pp 1036–1041
17. Lee W, Xiang D (2001) Information-theoretic measures for anomaly detection. In: Proceedings of the 2001 IEEE symposium on security and privacy, 2001. S&P 2001. IEEE, New York, pp 130–143
18. Liu AX (2012) Firewall policy change-impact analysis. ACM Trans Internet Technol (TOIT), 11(4):15
19. Liu AX, Gouda MG (2008) Diverse firewall design. IEEE Trans Parallel Distrib Syst 19(9):1237–1251
20. Liu Y, Gorton I, Lee VK (2008) The architecture of an event correlation service for adaptive middleware-based applications. J Syst Softw 81(12):2134–2145
21. Muraleedharan N, Parmar A (2010) ADRISYA: a flow based anomaly detection system for slow and fast scan. Int J Netw Security Appl 234–245
22. Njemanze HS, Kothari PS et al (2008) Real time monitoring and analysis of events from multiple network security devices. US Patent 7,376,969, 20 May 2008
23. Nurika O, Aminz M, Rahman ASBA, Zakaria MNB et al (2012) Review of various firewall deployment models. In: 2012 International conference on computer and information science (ICCIS), vol 2. IEEE, New York, pp 825–829
24. Nychis G, Sekar V, Andersen DG, Kim H, Zhang H (2008) An empirical evaluation of entropy-based traffic anomaly detection. In: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement. ACM, New York, pp 151–156
25. Preda S, Cuppens-Boulahia N, Cuppens F, Toutain L (2010) Architecture-aware adaptive deployment of contextual security policies. In: ARES'10 international conference on availability, reliability, and security, 2010. IEEE, New York, pp 87–95
26. Salah K, Elbadawi K, Boutaba R (2012) Performance modeling and analysis of network firewalls. IEEE Trans Netw Serv Manag 9(1):12–21
27. Sperotto A, Schaffrath G, Sadre R, Morariu C, Pras A, Stiller B (2010) An overview of IP flow-based intrusion detection. IEEE Commun Surv Tutor 12(3):343–356
28. Subramanian N, Pawar PS, Bhatnagar M, Khedekar NS, Guntupalli S, Satyanarayana N, Vijaykumar NK, Ampatt PK, Ranjan R, Pandit PJ (2005) Development of a comprehensive intrusion detection system-challenges and approaches. Inf Syst Secur 3803:332–335
29. Uribe TE, Cheung S (2007) Automatic analysis of firewall and network intrusion detection system configurations. J Comput Secur 15(6):691–715
30. Zhang D, Zhang D (2011) The analysis of event correlation in security operations center. In: 2011 International conference on intelligent computation technology and automation (ICICTA), vol 2. IEEE, New York, pp 1214–1216