ISSN 2278-3091
Volume 8, No.6, November – December 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse65862019.pdf
https://doi.org/10.30534/ijatcse/2019/65862019

Experimental Evaluation of a Hybrid Intrusion Detection System for Cloud Computing

Abdallah Ghourabi[1], Mohamed Jelidi[2]
[1] Jouf University, Saudi Arabia, aghourabi@ju.edu.sa
[2] University of Kairouan, Tunisia, jelidi.mohamad@gmail.com

ABSTRACT

Cloud computing is becoming an integral part of many businesses today. It offers wide range of benefits and competitive advantage over companies that do not shift to the cloud. This growing popularity makes cloud computing subject to several security issues. In this paper, we propose an approach to protect the cloud by providing a hybrid solution based on the distribution of intrusion detectors and the centralization of alerts for management purposes. The purpose of our approach is to protect the most important layers of the cloud using intrusion detection systems. Each layer has its properties that makes it different from other layers. This leads us to use specific intrusion detectors for each layer. The detection model relies on two techniques: signature-based detection and anomaly-based detection. The first technique targets previously known attacks, the second technique targets unknown malicious events. In this article, we describe the architecture of our approach and present some experimental results to evaluate its effectiveness.

Key words: Intrusion Detection System, Cloud Computing, Signature-based detection, Anomaly-based detection

1. INTRODUCTION

The cloud computing is becoming popular with large companies nowadays, mainly because they share valuable resources in a cost-effective way. This increasing migration towards cloud computing results in an escalating security threat, which is becoming a major issue. Many studies [1] [2] show that there are major security issues to take in consideration before moving onto the cloud. On the other hand, there is a wide repository of hacking techniques used by pirates to gather sensitive data and stolen credentials from cloud servers [8][9]. The most popular attacks concern SQL injections, Cross Site Scripting (XSS) and Denial of Service (DoS). The SQL injections and XSS attacks can easily pass through Intrusion Detection System (IDS), such as Snort, without being detected as it lacks the ability to detect attacks using hex-encoded values. The migration of networks and servers over the cloud involves that these attacks target cloud-based Web servers. The purpose of our approach is to protect the entire cloud environment by providing an intrusion detection system for each layer on the cloud. This approach is reinforced by adding an anomaly-based detection system at the web layer to detect unknown attacks.

The remaining parts of the paper are organized as follows: Section 2 reviews the related works. Section 3 presents the model design of our approach. Section 4 presents the experimental results. Finally, we conclude the paper in Section 5.

2. RELATED WORKS

Intrusion Detection System faces difficulties when transferred from traditional networks to cloud-based design, in this section we cover related works on different key layers of the cloud. Application level security refers to the usage of software and hardware resources for providing security to applications such that the attackers are not able to get control over applications and make unwanted changes. Grobert et al. [3] proposed a specific work to the problem of cryptography, which makes the IDS helpless when it cannot see the real traffic. The purpose of this study is to pick up which algorithms and keys are used by malicious programs which helps analysts to figure out quickly what a given binary program does. Various methods presented to identify cryptographic primitives (e.g., entire algorithms or only keys) within a given binary program. The evaluation shows that these methods of detection were able to extract cryptographic keys from a malicious binary program using improved heuristics, yet it has limitations, for example compilers can generate code differently which makes the malicious software invisible to the machine.

Boggs et al. [4] suggested an IDS that can identify zero day attacks by correlating Content Anomaly Detection (CAD) alerts from multiple sites. The correlation process compares each unique content alert from the local sensors against the Bloom filter representation of each unique content alert for mother sites. The evaluation of the system was based on building datasets from incoming traffic to their university, the content then is normalized for the CAD to create its model, after the models will be compared and extract normal models from abnormal ones. Experimental results show that this method decreases false positive rate and 80% of the compared models fit the filters with a false positive rate of 0.03%. The technique used in this contribution is effective only if the collaborative sites are done in a controlled and privacy preserving manner it can detect zero-day attacks.

The network is the backbone of Cloud, and hence vulnerabilities in network directly affect the security of the Cloud. Shaikh et al. [5] proposes a new anomaly detection method targeting the problem of detection of incompleteness

and inconsistency in access control, facing the major shortcomings of access control policies. The technique is mainly based on data classification technique using well-known Data mining tools. An experimentation showing the efficiency of the method for the discovery of inconsistency, incompleteness and redundancy in access control policy. Mishra et al. [6] suggests an advanced way to using snort the popular NIDS on the cloud focusing on securing the network layer specifically and centralizing outputs alerts of the system. This study uses snort rules to detect intrusive behavior rather than simple attacks, yet this approach still has difficulties to detect zero-day attacks and the system needs constant rule updates.

The host is the layer that carries virtual machines if it is vulnerable and can be exploited by hackers and threatens the isolation between guests. Since the majority of the host-based intrusion detection systems are based on the system log file analysis, lots of works have been conducted in this particular area. Ali and Len [7] have proposed a model of host-based IDS which focuses on log file analysis of Microsoft Windows XP. In that model, the intrusion is identified by matching the pre-defined intrusion pattern with the event logs of Microsoft Windows OS.

In the table 1 we present a comparison between the most important covered contributions on the area of study.

3. PROPOSED MODEL

The overall goal of our approach is to propose a hybrid intrusion detection system for the different layers of the cloud. It is designed to be integrated in a cloud computing environment, supporting the infrastructure layer within the same cloud provider. The proof of concept prototype is based on both a signature analysis and anomaly analysis. The key feature of the prototype design is that locations of the different IDS sensors are distributed in different cloud layers. Moreover, Correlated data will be logged for further forensics investigations.

3.1. Model Architecture

The study purpose is to provide a complete system architecture to secure the cloud entirely using IDS technology. Our methodology for this approach is to think in layers and depth, rather than on isolated and individualized issues. To think securely, we must significantly broaden our thoughts in order to refrain from focusing only on a single element of security or to defend against all type of attacks by securing one layer. Therefore, we propose a model of security for the entire cloud with focus on specificity of attacks on each layer.

The figure 1 shows the different entities of lab environment that was used for the test and presents the lines of defense, and figure 2 shows the interaction with the system by showing more actors and detailing the detection scenarios.

This network architecture design would be the most appropriate architecture to ensure security and high efficiency of the network. However, for the rest of the system design, instances for Intrusion Detection Systems and the attacker will be implemented within the cloud infrastructure. Implementing these instances on the cloud would be easier to manipulate and provide the same results than if the IDS/attacker were outside the cloud.

The detection process is segmented into two zones

1. Signature-based detection zone: to defend against known attacks.
2. Anomaly-based detection zone: to defend against previously unknown attacks.

Many signature-based IDS solutions in the market, one of the most known and effective solution is ”Snort”, which is the IDS that we choose to use in our experiments for the network layer. For the host layer, we chose OSSEC the open-source host-based intrusion detection system (HIDS). Host-based IDS involves loading a piece or pieces of software on the system to be monitored. The loaded software uses log files and/or the system’s auditing agents as sources of data. In contrast, a network- based ID system monitors the traffic on its network segment as a data source. Finally, for the web layer, we chose Modesecurity as a Web Intrusion Detection System (WIDS). WIDSs are designed to protect web applications/servers from web-based attacks that IPSs, NIDS and HIDS cannot detect.

The second segment of the model, is to secure the cloud against known and unknown attacks. Snort, OSSEC and Modsecurity are Signature based IDSs that protects the strategic layers of the cloud from known attacks. After that we need more intelligent techniques to secure the cloud from unknown attacks which introduces the necessity of the Recommender module that is trained against dataset of normal traffic and tested with real traffic on the server. What is considered to be normal traffic on the web will be scanned again by a trained anomaly detector to uncover unknown attacks that later will be recommended to the admin to manually investigate the context and nature of attacks and then update signature detection zone with new rules.

Table 1. Comparative summary of cloud ids most important covered solutions

Ref	Deployment	Architecture	Layers of interest	Detection approach	Proposed work	Security Objectives
Mishra et al. [6]	IaaS	Centralized	Network	Signature-based	System	Not Specified
Al-Mousa et al. [10]	IaaS	Cooperative	Network	Signature-based, Anomaly-based	System	Not Specified
Ghosh et al. [11]	IaaS	Centralized	Network, Host	Anomaly-based	System	Not Specified
Sangeetha et al.[12]	SaaS	Centralized	Application	Signature-based	System	Web attacks
Grobert et al.[3]	SaaS	Centralized	Host	Heuristic-based, Signature-based	Algorithm	Cryptography
Boggs et al.[4]	SaaS	Centralized	Application	Anomaly-based	Defense Framework	Zero-Day attacks

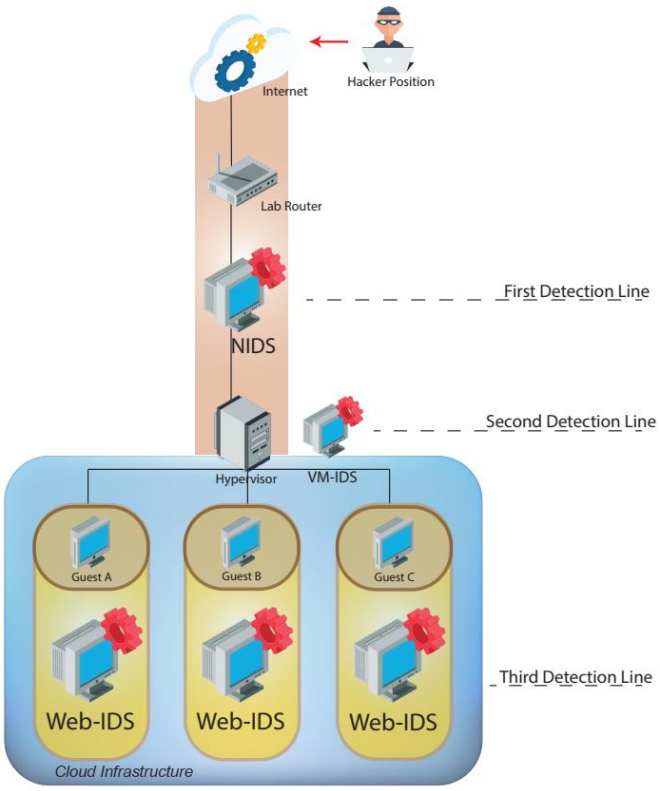

Figure. 1. The Proposed Network Architecture Design for Cloud IDS Model

Our contribution in this module was focused on preprocessing and feature selection, therefore, the idea was to design a selection process of the most significant attributes of the dataset based on the calculation of entropy. By removing less significant and non-significant attributes, it will bring us to a lighter set of data to be operated to get the classification results. All the alerts are centralized into one place using "syslogng" and correlated and visualized with "sguil".

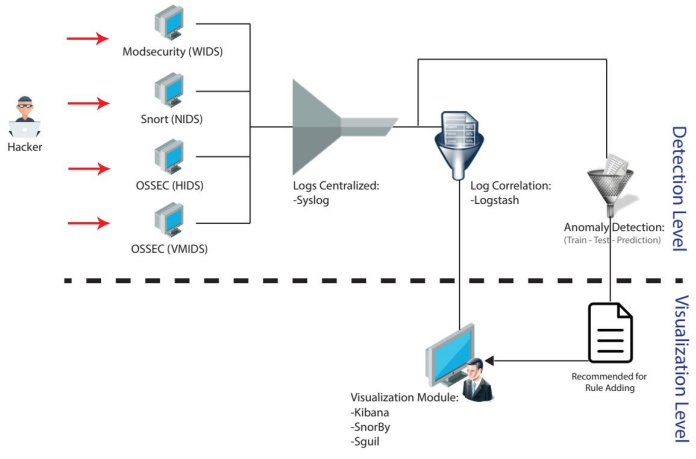

Figure. 2. Detailed architecture of the deployment locations of IDSs

3.2. Abstract detection scenario

To better visualize the detection process of the proposed model and to show the need for the combination of two detection methods in one solution, we put detailed steps in the flow chart presented in Fig 3. First, in the front-end of the network a router is placed to receive requests from end-users, usually any organization puts a firewall to block requests of IP that is classified in the black list, so we took that into consideration, if the IP is not in the black list and the user is not considered to be as a malicious user then the request is passed to the network IDS (Snort). In this case the Network IDS analyzes the request using its signatures and raise alarms in case the request matches one or more of snort's rules, else no alarms will be raised and the traffic is forwarded to the next stage of analysis, which is the host IDS (Ossec). Ossec is watching events happening on the Hypervisor and on the network so it is in a powerful position to see what other IDS can't see. Ossec analyzes the request and other events that are happening in the local machine and in the network using its rule database and if something matches one or more rules than alarms will be raised, else the request continue to flow to the server. Now the server is the most important part of the architecture because it is where queries are executed and tasks are being processed so it is very important to make it secure as much as possible. The request when it is on the server Modsecurity does exactly like Snort and Ossec in previous stages, it analyzes the request using its set of rules and raise alarms in case of matching

The thing to be added is the use of anomaly-based detection module that defends against unknown attacks to the system using machine learning techniques. This module is placed just next to Modsecurity waiting for abnormal requests, these unusual requests can be attack attempts. The system will be trained to distinguish between what is (normal) and what is (abnormal) by applying supervised learning in this stage of the model, and if the module finds something that looks like an attack or it is abnormal, then it will recommend it to the administrator to verify that event by hand and analyze logs manually and get sure if it is truly a malicious event. In case, that recommended event is an attack then the administrator will write a set of rules and adds them to other IDS.

4. EXPERIMENTAL EVALUATION

In order to test our approach, we created a private cloud based on the model architecture presented in Fig. 1 and Fig. 2.

4.1. Experimental environment

To build the architecture presented in the previous section we used "Virtualbox" as the hypervisor and installed "LUbuntu15" with an apache running webserver, we have also set and configured all the IDSs in places respectively

1) *Network*: Snort (Network packets)
2) *Host*: Ossec (Logfiles/Processes)
3) *Web*: Modsecurity (Http and various traffic)
4) *Web*: Anomaly Detector (Normal traffic)

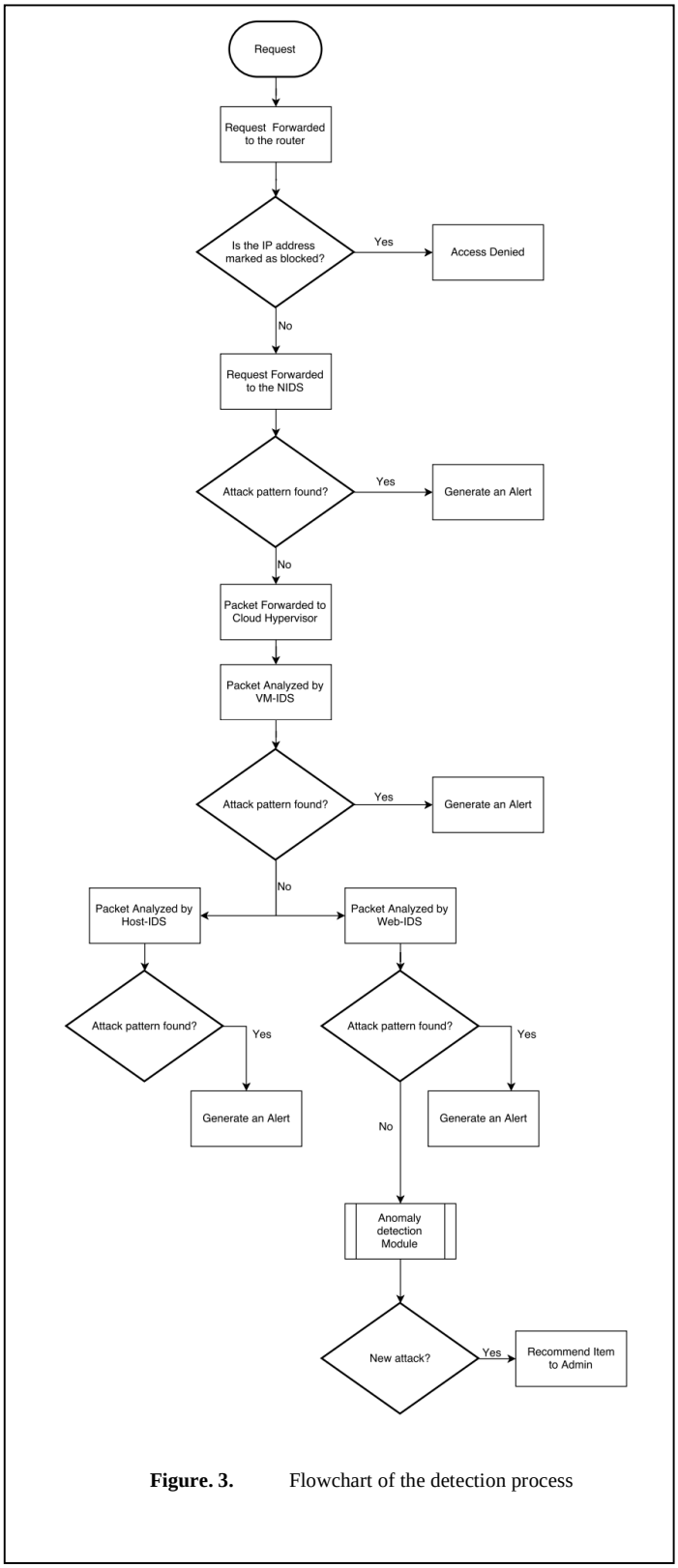

Figure. 3. Flowchart of the detection process

In order to test the capability to protect the entire cloud we have 2 types of datasets, one to test Signature-detection zone, and the other is to train and test the Anomaly-based detection zone detailed in the table below.

Table 2. Dataset characteristics for first and second zone

Number of resources	Targeted layers	Datasets Total Size	Dataset/Tools	Detection Method	Detection Zone
70	Network, Host, Web	More than 235 MB	Pcap Files and W3af	Signature -Based	1st
36,000 normal	Web	More than 99 MB	CSIC 2010 HTTP	Anomaly-Based	2nd

Real Pcap file traces of attacks were collected from network traffic blog [13]. W3af [14], a free Web Application Attack and audit framework, is used to test Modsecurity against the top 10 widely used attacks by OWASP (Open Web Application Security Project).

Centralizing raised logs is very important step for analysis, therefore, we used ElasticSearch which is a server log, and logstash to be able to read all logs in various format. Sguil is used for visualization and reporting of the signature-based zone.

4.2. Experimental results

4.2.1. Signature-based detection zone

After installing, configuring the testing environment and deploying IDSs to the strategic positions, we launched various attacks taking into consideration the differences of each layer. We generated specific attacks to target the network, host and the webserver. The collected data have come from different sources:

- Real traffic from the Network.
- Web vulnerability scanner (W3af).
- Simulated attacks on the host.

These attacks are distributed against the different private cloud layers:

1) 52% of attacks target the network.
2) 34% of attacks target the webserver.
3) 14% of attacks target the host.

a) Quantitative results

These differences of each layer of the cloud shows systematically the differences between each type of IDS, Network IDS are made to capture and analyze data from the network so it is provided with tools and softwares to work on the network, and it is not designed to capture and analyze web traffic because the HTTP protocol is an application layer protocol and NIDS are not designed to analyze these types of data. This is the central idea of the approach; we use on each layer the Intrusion Detection System that is designed for it. Next, in table 3 we present the dataset and tools used to test and examine IDS on the network, host and the web.

Table 3. Dataset and security tools characteristics

Number of resources	Targeted layers	Datasets Total Size	Dataset/Tools	Number of sessions
70	Network,	More than	Pcap Files and	88

	Host, Web	235 MB	W3af	

We used 36 datasets in the form of Pcap files of real attack attempts collected from 2014 until 2016 from network traffic blog [13], these Pcap files contain real IP addresses of really infected computers with payloads and malicious files. we used these datasets to test the detection on the network, knowing that none of our IDSs where able to detect these attacks because none of OSSEC or Modsecurity is listening to the network, they are just listening to their specific location. We can learn from this that a NIDS cannot be replaced by a HIDS or WEB-IDS and vice-versa. In table 4 below, we present more details about the dataset and the platform used to lunch the attacks such as Exploit Kit which is a platform used to create the payload that can be used to attack the victim machine. An attack can be launched one time and relaunched again by the hacker for this reason we can have multiple sessions more than the number of individual attacks.

Table 4. Dataset and security tools used in the Network

Number of resources	Targeted layers	Platform / Payloads	IDS	Total Number of sessions
36	Network	Exploit Kit Angler Exploit Kit Fiesta Exploit Kit Neutrino Exploit Kit Angler Exploit Kit Magnitude Exploit Kit Nuclear Exploit Kit RIG Exploit Kit Upatre downloader Malspam	Snort	53

In the next table 5 we present ten false manipulations of the host by trying to access or remove sensitive files from the system, which is considered as an intrusion by OSSEC, these bad manipulations like a simple user tries to install a malicious software by using root commands cannot be detected by Modsecurity or Snort, because they are not designed for monitoring the host.

Table 5. Dataset and security tools used on the Host/Guest

Number of resources	Targeted layers	Host/Guest	IDS	Total Number of sessions
10	Host	LUbuntu 15	OSSEC	10

In the next table 6, we present the details of attacks launched to test Modsecurity using W3af to simulate web attacks, many of these attacks have been listed in the logs of OSSEC, but this does not mean that OSSEC has detected them, one of the characteristics of the HIDS is that it collects data from log files on the local machine, and the web server is a software running on that machine for this reason Ossec was able to see Apache2 log files and analyzes it. But we can remark that severe attacks that are classified by Modsecurity as "Critical" are only considered as level 2 issue by OSSEC. OSSEC Level 2 alerts should be interpreted as ignored because Level 1 and 2 are just levels used to reduce False positives, for this reason many critical attacks are considered as False positives by OSSEC, which can mislead the administrator if he uses a HIDS only in place of a Web-IDS. We can learn that a HIDS cannot replace a Web-IDS even it can see its log files.

Table 6. Dataset and security tools used on the Web

Number of resources	Targeted layers	Platform / Payloads	IDS	Total Number of sessions
24	Web	W3af Blind_sqli Buffer_overflow csrf dav eval file_upload format_string frontpage generic global_redirect htaccess_methods ldapi lfi mx_injection os_commanding phishing_vector preg_replace redos response_splitting rfi sqli xpath xss xst	ModSecurity	24

After testing all the sensors placed on the private cloud, we have had good detection rate, as the next table 7 shows the percentage of detection in all the system.

Table 7. Percentage of detection in Signature detection zone

TP/FN	Number of attacks	Percentage
True Positives	64	91.43%
False Negatives	6	8.57%

True positives mean that real attacks on the network, host or the web are detected, logged and alerted as attacks. From our results, the detection rate is 91.43%. On the other hand, False negatives means that real attacks on the network, host or the web are not detected and the system was infected by these intrusions, here FN is only 8.57%. We used only standard and free rulesets for each IDS, and to the best of our ability, we used the most UpToDate attacks against the system trying to make the working environment meeting more realistic attacks, what is sure is that by adding more rules to the IDSs the detection rate of True Positives should be increased systematically.

The problem of adding rules constantly to the IDSs describes the weaknesses of this method "Signature-based detection", where it is not able to see new attacks and custom-made attacks by professional hackers which can harm the system. We

continue the amelioration of our approach by adding a second module that can detect zero-day attacks by using anomaly detection techniques which will be discussed in the paragraph "Anomaly-based detection zone" of this section.

b) Qualitative results

To deceive the Intrusion detection systems, some evasion techniques can be used by hackers in order to bypass IDSs, the following are the most common methods of evasion attacks:

- **Obfuscation**: (or String matching) it is the process of manipulating data in order to make IDS's signature do not match the packet, the network flow or the Http requests. For instance, sending a packet encoded differently or adding external null characters can deceive a Signature based IDS.
- **Fragmentation**: (or Session splicing) it is the process of breaking an attack into multiple packets.
- **Encryption**: In this scenario the attacker uses any attack against the HTTPS Web site, such as SQL injection, buffer overflows, and directory traversals, that would work on the HTTP site.
- **Denial of Service**: it is the process of overloading the NIDS and causing it to crash.

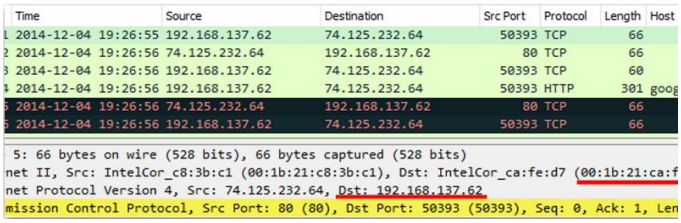

Figure 4. Mac and IP address of the infected host

We take one real scenario that happened in 2014 where the machine named "IntelCor_8" with a MAC address 00:1b:21:ca:fe:d7 and IP address 192.168.137.62 and running on Windows. This victim visited a compromised website "*www.earsurgery.org*" with a static IP address equal to 216.9.81.189. After visiting the compromised website the victim was redirected to "*qwe.mvdunalterableairreport.net*" that delivered the exploit kit EK and malware payload to the host, the IP address that delivered the exploit kit and malware payload was 192.99.198.158. In the figure 4 we show some information about the infected victim.

The redirect URL that points to the exploit kit (EK) landing page is "lifeinsidedetroit.com-POST/02024870e4644b68814aadfbb58a75bc.php?q=e8bd3799ee8799332593b0b9caa1f426" as shown in the figure 5 below.

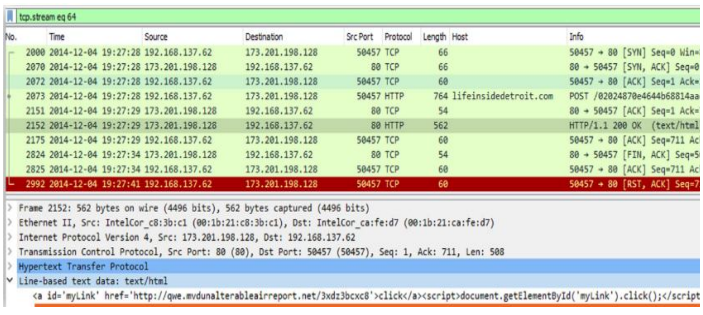

Figure 5. The redirect URL that points to the exploit kit landing page

After that, we have extracted the binary from the network traffic to examine it, and we used a free hexadecimal editor to open the file then we found that the exploit was obfuscated with "adR2b4nh" string as shown in the next figure 6.

The exploit was CVE-2013-2551 Internet Explorer exploit lunched from Angler Exploit Kit which tries always to encrypt

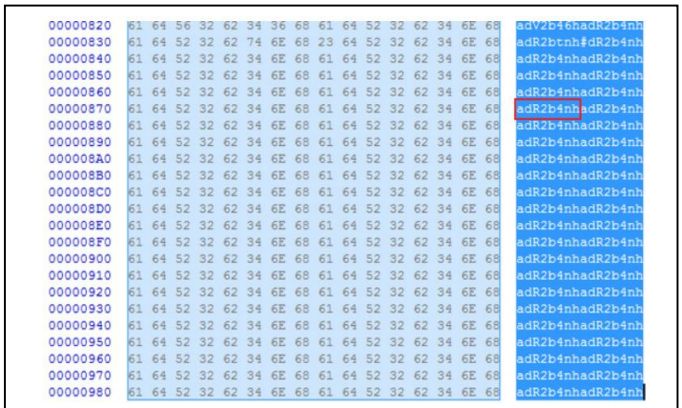

Figure 6. Showing the obfuscating string of the payload "adR2b4nh"

or obfuscate the payload, and the infected browser was Internet Explorer 9. These payloads are injected to the landing page, after doing all the work of generating the exploit and obfuscation it will be injected in the landing page and we show in the figure 7 an example of the used webpage from the dataset.

Below are the Snort alerts generated by this Pcap file:

ET CURRENT_EVENTS 32-byte by 32-byte PHP EK Gate with HTTP POST (sid:2018442)
ET TROJAN Zeus GameOver Possible DGA NXDOMAIN Responses (sid:2018316)
ET CURRENT_EVENTS DRIVEBY Angler EK Apr 01 2014 (sid:2019224)
ET CURRENT_EVENTS Angler EK Oct 22 2014 (sid:2019488)
ET CURRENT_EVENTS Angler EK Flash Exploit URI Struct (sid:2019513)
ET TROJAN Bedep SSL Cert (sid:2019645)

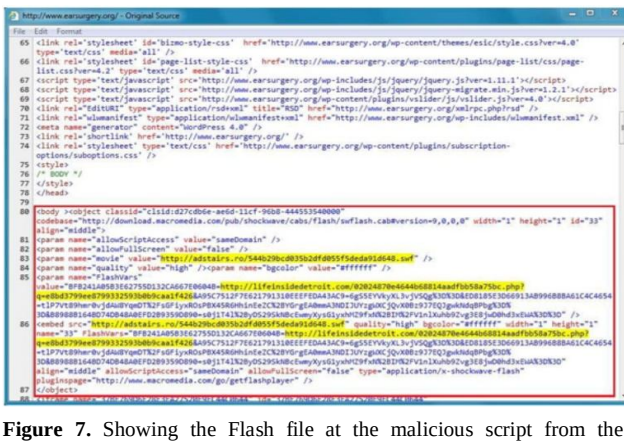

Figure 7. Showing the Flash file at the malicious script from the

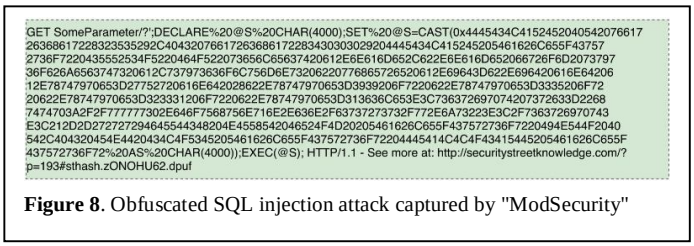

Figure 8. Obfuscated SQL injection attack captured by "ModSecurity"

which means that snort has detected this attack even obfuscation was used, and in most cases Angler EK has some shell code at the beginning of the file containing the payload. Now this was a successful example of an attack that uses an evasion technique to deceive Snort, but Snort was able to detect this attack because the level of obfuscation was not complicated enough for Snort to get deceived, for this reason we demonstrate another example. On the other hand, the evasion technique was applied to an SQL Injection attack that targets the webserver as shown in figure 8, ModSecurity was able to detect it, but snort didn't notice it at all.

As shown in figure 9, Modsecurity successfully detected and alerted the obfuscated sql injection attack, because it can handle Internationalization (I18N) and thus properly handle various data encodings including Unicode and UTF-8 in order to prevent not only evasion techniques but also to minimize false positives.

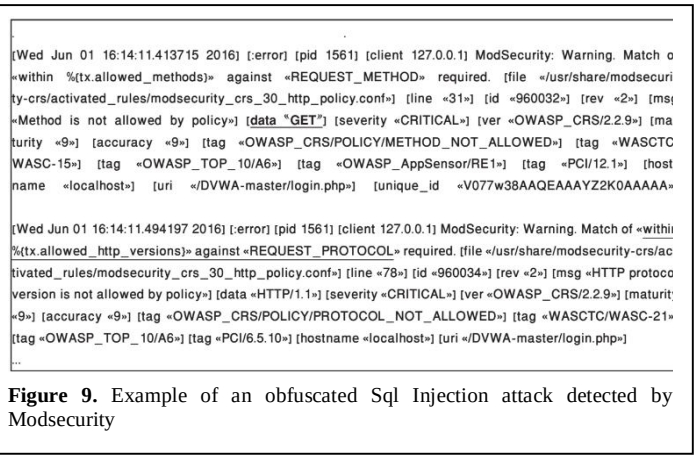

Figure 9. Example of an obfuscated Sql Injection attack detected by Modsecurity

Web-based intrusion detection systems (WIDS) like Modsecurity have the potential to be more effective than network-based IDSs (NIDS) like Snort, because a web firewall can see the request exactly as it will be handled by the web server, but a network-based IDS cannot. On the other hand, Network-based systems, like Snort, can only see the network packets, so they have to guess how the web server will interpret them. This introduces additional possibilities for evasion attacks against network-based IDSs. Evasion attacks pose a greater challenge for network-based solutions, than a web-based IDS like modsecurity. The advantage of network-based solutions is that they can be easier to deploy which can provide us with a first layer of security. If we have a number of servers on the datacenter, we can deploy a network-based IDS at a chokepoint in the network, and they protect every server on the datacenter, without needing to install and configure Host-based IDS at each one.

We learn from these two cases that it is extremely important to take into consideration how to assign IDSs: what IDS in what layer.

4.2.2. Anomaly-based detection zone

In the second zone of detection that solves the problem of zero-day attacks, we focus here on preprocessing which is 90% of the overall process. Our proposed method to clean the data resides on two steps, the first step is to remove noisy attributes and the second step is to keep reducing the number of attributes until having the briefest set of significant attributes. Noisy attributes are a source of errors, thus, can be a cause to misclassify security related events occurring on the server. Below we describe in-depth the transformation done on the dataset.

We used The "CSIC 2010 HTTP Dataset" [15] in CSV format which contains thousands of web requests. It can be used for the testing of web attack protection systems. It was developed at the "Information Security Institute" of CSIC (Spanish Research National Council). The HTTP dataset CSIC 2010 contains the generated traffic targeted an e- Commerce web application. In this web application, users can buy items using a shopping cart and register by providing some personal information. As it is a web application in Spanish, the data set contains some Latin characters. The dataset contains 36,000 normal requests and more than 25,000 anomalous requests. The HTTP requests are labeled as normal or anomalous and the dataset includes attacks such as SQL injection, buffer overflow, information gathering, files disclosure, CRLF injection, XSS, server side include, parameter tampering and so on.

The dataset was processed, according to the following process:

1) The dataset was evaluated by measuring the information gain with respect to the class "norm", "anorm" by applying the entropy formula.

$$InfoGain(Class, attribute) = H(Class) - H(Class|Attribute) \quad (1)$$

What InfoGain basically does is measuring how each feature contributes in decreasing the overall entropy, knowing that the Entropy, H(X), is defined as follows:

$$H(X) = -\sum_{i}(P_i * log_2(P_i)) \quad (2)$$

With P_i being the probability of the class i in the dataset, and log_2 the base 2 logarithm. Entropy basically measures the degree of "impurity". The closest to 0 it is, the less impurity there is in the dataset.

2) Step 1 outputs the list of attributes ranked from higher to lower. Therefore, in Step 2 lower ranked attributes with value of "0" are removed from the dataset.
3) Repeat Step 1 and Step 2, until removing all the noisy attributes.
4) Digitize the features of the reduced set of significant attributes.

5) Dataset is ready to use for train and test.

We Cleaned the dataset based on this process as detailed in table 8.

Table 8. Applied Example on the HTTP Dataset

Step	Task
1	- Ranked attributes: 0.99649 16 cookie 0.42637 17 payload 0.29471 1 index 0.12669 3 url 0.10206 14 contentLength 0.01273 2 method 0.00892 12 host 0.00492 15 contentType 0 6 pragma 0 4 protocol 0 5 userAgent 0 7 cacheControl 0 13 connection 0 11 acceptLanguage 0 10 acceptCharset 0 8 accept 0 9 acceptEncoding
2	- Set of Significant attributes = {cookie, payload, index, url, contentLength, method, host, contentType} - Set of Noisy attributes = {pragma, protocol, userAgent, cacheControl, connection, acceptLanguage, acceptCharset, accept, acceptEncoding}
3	Previous steps(1 and 2) were repeated many times until having Set of Significant attributes = {payload}
4	GET Replaced by 1 POST Replaced by 2 PUT Replaced by 3 localhost:8080 Replaced by 5 payload Replaced with the length of the string …
5	We have many sub-datasets that contains only significant attributes. dataset = {sub-dataset_1, sub-dataset_2, ... ,sub-dataset_n}

To evaluate the efficiency of the cleaning phase, we tested the dataset on several classifiers before and after the cleaning process. The results showed that this step improves the detection rate of attacks.

4.3. Events visualization

Intrusion detection systems are tools that help the admin of the cloud to monitor current and past state of the machine. Because of the high rates of interaction on the cloud due to the services that it offers to end users, the admin might find huge number of logs collected by network devices and intrusion detectors that are waiting for examination. For this reason, it is really difficult to examine all the logs manually without any tool that helps to visualize and correlate different events and logs.

In this approach, we used sguil [16], an Open Source Network Security Monitoring tool, to visualize the events collected by our detection sensors. Figure 10 shows a capture screen of alerts in sguil after launching our attacks against the victim guest. These alerts can be correlated and regrouped to see the full alerts that come from one specific source IP and that are critical. It also shows the content of the malicious packets as well as the rule that was triggered by that particular packet as illustrated in figure 11.

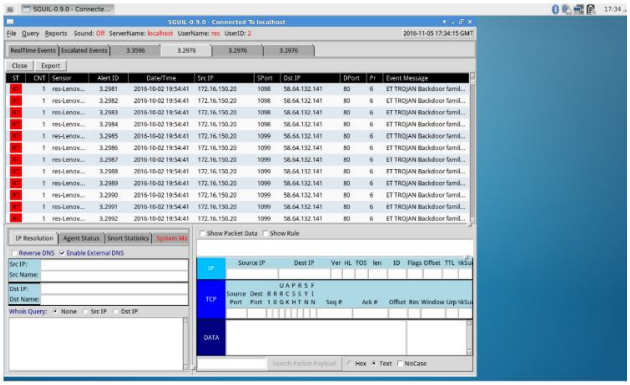

Figure 10: Visualizing Logs with Sguil

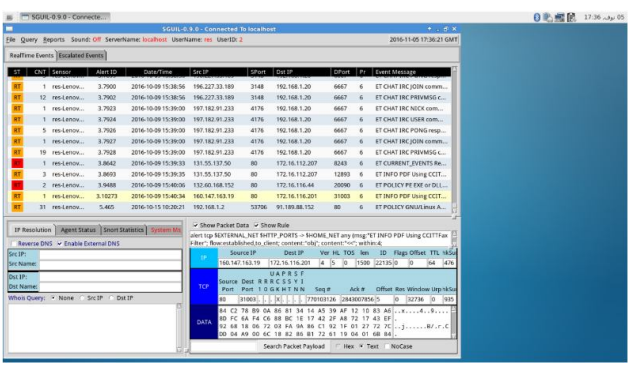

Figure 11: Visualizing the content of a malicious event with Sguil

5. CONCLUSION

In this paper, we proposed a hybrid system framework to fully protect the cloud using intrusion detection system technology. The creation of this framework system was essentially based on the use of two detection methods: Signature based and Anomaly based methods using open source softwares to produce a concept proof work. Experimental results show that it is accurate to use such combination of IDS in precisely located positions on the cloud to have a wide monitoring area as well using the strengths of each sensor. The presented and implemented model is focusing on centralizing logs generated by intrusion detection systems from various layers to guarantee that IDS are distributed and having good management environment by centralizing logs. In the second detection zone that uses anomaly techniques we focused on the preprocessing phase, by cleaning data and selecting the most appropriate attributes that are significant for the detection.

In this approach, the anomaly-based detection is limited to the web layer. As future work, we plan to apply this technique on the other layers. In addition, the model proposed in this paper contains several detection tools that collaborate together, which can influence the performance of the system. For this reason, we also plan to work on the optimization of this model by seeking faster techniques while maintaining a good level of accuracy.

REFERENCES

1. K. Ren, C. Wang and Q. Wang. **Security Challenges for the Public Cloud**, in IEEE Internet Computing, vol. 16, no. 1, pp. 69-73, 2012. https://doi.org/10.1109/MIC.2012.14
2. N. V. Juliadotter and K. R. Choo. **Cloud Attack and Risk Assessment Taxonomy**, in IEEE Cloud Computing, vol. 2, no. 1, pp. 14-20, 2015. https://doi.org/10.1109/MCC.2015.2
3. F. Grobert, C. Willems, and T. Holz. **Automated identification of cryptographic primitives in binary programs**, in RAID, Springer, vol. 6961, 2011, pp. 41–60. https://doi.org/10.1007/978-3-642-23644-0_3
4. N. Boggs, S. Hiremagalore, A. Stavrou, and S. J. Stolfo. **Cross-domain collaborative anomaly detection: So far yet so close**, in Recent Advances in Intrusion Detection, Springer, 2011, pp. 142–160.
5. R. A. Shaikh, K. Adi, and L. Logrippo. **A data classification method for inconsistency and incompleteness detection in access control policy sets**, International Journal of Information Security, pp. 1–23, 2016.
6. V. Mishra, V. K. Vijay, and S. Tazi. **Intrusion detection system with snort in cloud computing: Advanced ids**, in Proceedings of International Conference on ICT for Sustainable Development, Springer, 2016, pp. 457–465. https://doi.org/10.1007/978-981-10-0129-1_48
7. F. A. B. H. Ali and Y. Y. Len. **Development of host based intrusion detection system for log files**, in Business, Engineering and Industrial Applications (ISBEIA), 2011 IEEE Symposium on, IEEE, 2011, pp. 281–285.
8. A. E. Karrar, M. F. I. Fadl. **Security Protocol for Data Transmission in Cloud Computing**, International Journal of Advanced Trends in Computer Science and Engineering, vol. 7, no. 1, pp. 1-5, 2018 https://doi.org/10.30534/ijatcse/2018/01712018
9. G. Amrulla, M. Mourya, R. R. Sanikommu and A. A. Afroz. **A Survey of : Securing Cloud Data under Key Exposure**, International Journal of Advanced Trends in Computer Science and Engineering, vol. 7, no. 3, pp. 30-33, 2018. https://doi.org/10.30534/ijatcse/2018/01732018
10. Z. Al-Mousa and Q. Nasir. **Cl-cidps: A cloud computing based cooperative intrusion detection and prevention system framework**, in Future Network Systems and Security, Springer, 2015, pp. 181–194.
11. P. Ghosh, A. K. Mandal, and R. Kumar. **An efficient cloud network intrusion detection system**, in Information Systems Design and Intelligent Applications, Springer, 2015, pp. 91–99.
12. S Sangeetha, R Ramya, M. Dharani, P Sathya, et al. **Signature based semantic intrusion detection system on cloud**, in Information Systems Design and Intelligent Applications, Springer, 2015, pp. 657–666. https://doi.org/10.1007/978-81-322-2250-7_66
13. Network traffic analysis, www.malware-traffic-analysis.net
14. w3af a Web Application Attack and Audit Framework, www.w3af.org
15. Carmen Torrano Giménez, Alejandro Pérez Villegas, Gonzalo Álvarez Marañón. **HTTP Dataset CSIC 2010**, CSIC (Spanish Research National Council), 2012, http://www.isi.csic.es/dataset/
16. Sguil: The Analyst Console for Network Security Monitoring, https://bammv.github.io/sguil/index.html