Scalable IDS System: Exploring the Design Space

Ranjan Mohan
ranjanmo@buffalo.edu

Tanmay Bhagwat
tanmaybh@buffalo.edu

Kun Yang
kunyang@buffalo.edu

Aaron Huber
ahuber@buffalo.edu

ABSTRACT

Extensive research is being carried out in the field of IDS design to build a highly scalable IDS without compromising efficiency and security. The idea of this project is to develop an effective and scalable hybrid IDS design using a Complex Event Processing Module as the core and come up with a generic hybrid IDS design to bring forth a standardization in the IDS Framework portion of the computer security domain and to evaluate it in terms of performance and security requirements.

1. INTRODUCTION

Network security has become a very challenging aspect, and intrusion detection systems (IDSs) are becoming essential solutions to identify different kinds of attacks. There are several types of Intrusion Detection Systems with a fair share of merits and demerits, thus making IDS design a field suitable for further research and development. Another big concern of IDS design is that as the computer networks are growing exponentially, there is a need for an effective and highly scalable IDS that allow more deployments to manage and defend a larger network. This project combines multiple intrusion detection techniques in an optimal manner to meet the requirements in terms of security and scalable performance.

1.1 Motivation

Some prior works have been published to address these problems. For detection strategy, there is a hybrid system that uses a signature-based detection along with an anomaly based detection technique, namely Multivariate Correlation Analysis (MCA) [2]. To improve the scalability of the system, IDS as a service (IDSaaS) has been proposed to provide an easy to access IDS using the cloud [3]. There is also a parallel design that effectively distributes the detection workload among multiple cores [4]. However, the existing solutions stated above focus mainly on either the security or scalability and not both. In this project, we adapt the idea of using IDS as a service (IDSaaS) as well as make use of the Multivariate Correlation Analysis algorithm. Our proposed system is a hybrid IDS that uses signature based detection, anomaly based detection strategies and Complex Event Processing (CEP) for intrusion detection. In our CEP module, we implement MCA to estimate and characterize the normal state. So compared to a traditional IDS, our design can provide better defense against Zero day attacks as well. Also the IDSaaS approach ensures scalability of our system by offering the IDS capability using network connectivity only. Thus, we attempt to achieve security as well as scalability.

1.2 Challenges

The major challenges we faced during designing and implementing this project were:

1. Identifying and integrating relevant libraries
   - In feature extraction, we extracted 16 unique features from the host system. In order to extract system details like number of processes running, firewall status, etc. we had to make us of a library called Hyperic-SIGAR (System Information Gatherer and Reporter) [7]. Implementing the use of such external libraries brought about its own set of challenges. With such libraries, documentation is much more limited when it comes to trouble-shooting library and machine malfunction. After hours of searching for why SIGAR was not capturing data on the particular machine, it was discovered that the NIC of the machine in question would not operate in promiscuous mode. After a quick change in the code, SIGAR functioned as intended.
   - Finding libraries for matrix computations (JAMA)[12], Effective rule execution (jDrools) [11].
   - Additionally, we not only captured live traffic - incoming and outgoing, but also analyze it in order to learn, which transport layer and application layer protocols are being used, which IP addresses and port numbers are being accessed, list of TCP/UDP connections, list of LAN-internal IPs, list of LAN-external IPs, etc. In order to make this possible we had to use a library called jNetpcap [14]. This library required a lot of dependencies like winpcap, libpcap, etc.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

2. Design Implementation and debugging

   - The challenges faced in building SBM included getting external libraries to function on particular machines, debugging code, and discovering efficient and effective strategies for correctly identifying specific signature based attacks. Also, when building an SBM, two things are desirable, namely, precise detection and efficient implementation. Solid research was needed to define and employ effective methods with those characteristics. While the SBM module is a clean and robust model, more research will be of use to increase detection precision, efficiency, and the number of attacks being analyzed and detected.
   - Debugging code is considered to be just a part of the process when developing an application. However, with a limited window of time to have the application up and running, debugging code was a challenge as the time spent in the process took away from the time remaining for the creative process.
   - Heavy Matrix computations needed to be performed for the MCA Computation. And the runtime was close to 19 hours, which required a lot of optimization to the code to bring it down to about 16 hours.

3. Merging and integrating all the modules

   - Considering the scale of the project, we maintained some level of abstraction to each others implementation. In view of that merging the project took us considerable amount of time.
   - Proper management of dependencies. Initially on using the Java SE version of eclipse we had a tough time maintaining the dependencies on shifting the code from one system to another. Then we switched to Java EE and use Maven [18] to manage the dependencies and build configuration which makes it easy for any user with the project folder to automatically resolve all dependencies.
   - Heavy Matrix computations needed to be performed for the MCA Computation. And the runtime was close to 19 hours, which required a lot of optimization to the code to bring it down to about 16 hrs.

4. Developing and integrating the system with a professional front-end

   - Developing a professional and Administrator friendly front-end to cater to the reporting of IDS alerts and statistics. To cater to this purpose, we used the Gentella Admin Template [16] with heavy modifications hosted on an Apache Server using the Jersey Restful API to maintain updated values on the webpage. This required extensive coding in HTML, CSS and JavaScript as well.

1.3 Contribution

The main contributions of this project are as follows:

1. We demonstrate that by deploying CEP module separately, we avoid processor intensive computation like MCA from inhibiting the host and gateway performance

2. The IDSaaS idea can help our CEP gather important information and provide more accurate analysis as well as provide scalability.

3. By using relational databases for storing all the rules, the efficiency of rule searching was considerably improved.

1.4 Organization of Report

In section 2, we summarize the base concepts on which this projects foundation is laid on. In section 3, we state the functional and security requirements that enforce the need of the proposed system. In section 4, we introduce the outline of the proposed system. In section 5, we comprehensively state the design details and methods used to implement system. In section 6, we introduce the evaluation methods, use them to evaluate our IDS design against the stated requirements stated in section 4. In section 7, we provide information on related work and their comparison with our proposed system. In section 8 we provide a perspective on how this work can be taken further in terms of future developments. In section 9, we give our concluding remarks on this project and the purpose it has served followed by section 10 which encompasses all the references used to build this project.

2. PRELIMINARIES

In traditional IDS designs, there are two widely used detection approaches:

- Signature-based detection - Uses known parameter values to detect intrusions

- Anomaly-based detection - Identifies abnormal variation in parameters to detect possible intrusions There are pros and cons for the detection approaches. Signature-based detection is not effective against new variants of any attack or Zero day attacks, and on the other side anomaly-based detection techniques require heavy computation.

Some other preliminaries that are needed to have a thorough understanding of this project are:

- Complex Event Processing (CEP) - Receives feature and event related information from various sources and tries to correlate the events by using predefined rules.

- Multivariate Correlation Analysis (MCA) - The anomaly based detection algorithm used in the CEP Module. This algorithm is explained in detail later in this report.

3. REQUIREMENTS

The requirements that are to be mandatorily addressed by any IDS Design are listed below:

3.1 Functional Requirements

The functional requirements of the IDS are as follows:

- *Modularity* - The design and implementation of the system should be split into clearly separated functional modules. Each module is responsible for a separate work, and they should have well defined boundaries and interfaces between each other. This property is essential for this project in order to ensure scalability and ease of upgrading parts running on different devices.

- *Cost Effectiveness* - The system should be deployed with a relatively low cost. The cost mainly consists of two parts, infrastructure deployment cost and operating cost. The users should be able to adjust their cost according to their specific requirements.

- *Efficiency* - The system should be efficient. In users point of view, their machine should not be imposed with heavy computation. For the overall system, the efficiency should only be limited by connectivity factors, not computational limitations.

- *Ease of Deployment* - We should be able to deploy the system easily. More precisely, the system should be easy to install in a given network setup, add more user machines to the IDS watch list when needed, and add new networks to the system to manage security.

3.2 Security Requirements

The security requirements of the IDS are as follows:

- *Communication integrity and continuous flow* - The communication between local CEPs and Cloud server should be secure. The communication between the local user machines and local CEPs should also be secure. Attackers should not be able to block, capture, or modify the communication in any part of the system.

- *Cloud reliability and trustworthiness* - The system relies on the cloud service for centralized control and maintenance, so the Cloud service provider and the setup used should be reliable. Sensitive data is processed by the Cloud server, so data confidentiality is important for this system. There should be no vulnerabilities in this portion of deployment for the adversary to compromise the data or masquerade the server.

- *Efficiency* - The system should be efficient. From the users point of view, their machine should not be imposed with heavy security related computation..

- *Less Invasiveness/Small Attack Surface* - The system should have minimal security risks. The portion of deployment prone to Software, Network or Social Engineering related attacks should be minimized.

4. DESIGN

The proposed setup as shown in Fig 1, is divided into two portions based on their deployment:

4.1 Intra-network Implementation

This corresponds to the portion of deployment confined within a single network and consists of:

- **Network IDS (NIDS)**: The NIDS is placed at the gateway of the network in concern in order to capture all network related traffic at a single point. This deployment extracts features that characterize the network and also detects threats based on a signature based detection module. Additionally, it sends the logged features to the Local CEP Module using Transport Layer Security (TLS) for further analysis and log maintenance.

- **Host IDS (HIDS)**: The HIDS is deployed at each and every host in the network and it closely monitors and logs host-based features to detect threats based on a signature based detection module implemented as a part of it. Besides this, it also sends the captured features to the Local CEP Module using Transport Layer Security (TLS) for further analysis and logging purposes.

- **Local CEP Module**: This module is a separate hardware deployment whose placement is identical to that of a host. It aggregates all the feature updates sent by the HIDs and NIDS of the network in concern and uses anomaly based detection techniques like Multivariate Correlation Analysis (MCA). Additionally, it also uses a rule based detection technique with a wider rule set compared to the NIDS and HIDS to correlate multiple related security incidents that seem independent to each other.

4.2 Inter-network Implementation

This comprises of a deployment that has access to feature information over multiple networks, namely:

- **Comprehensive IDS offered as a SaaS (Software as a Service)** This is deployed as a cloud service. It aggregates the results from various local CEP modules, thus making it an ultimate point of log collection and an apt location for a comprehensive analysis of the collected data to identify sophisticated threats that are not evident from a local networks perspective.

5. IMPLEMENTATION AND DESIGN DECISIONS

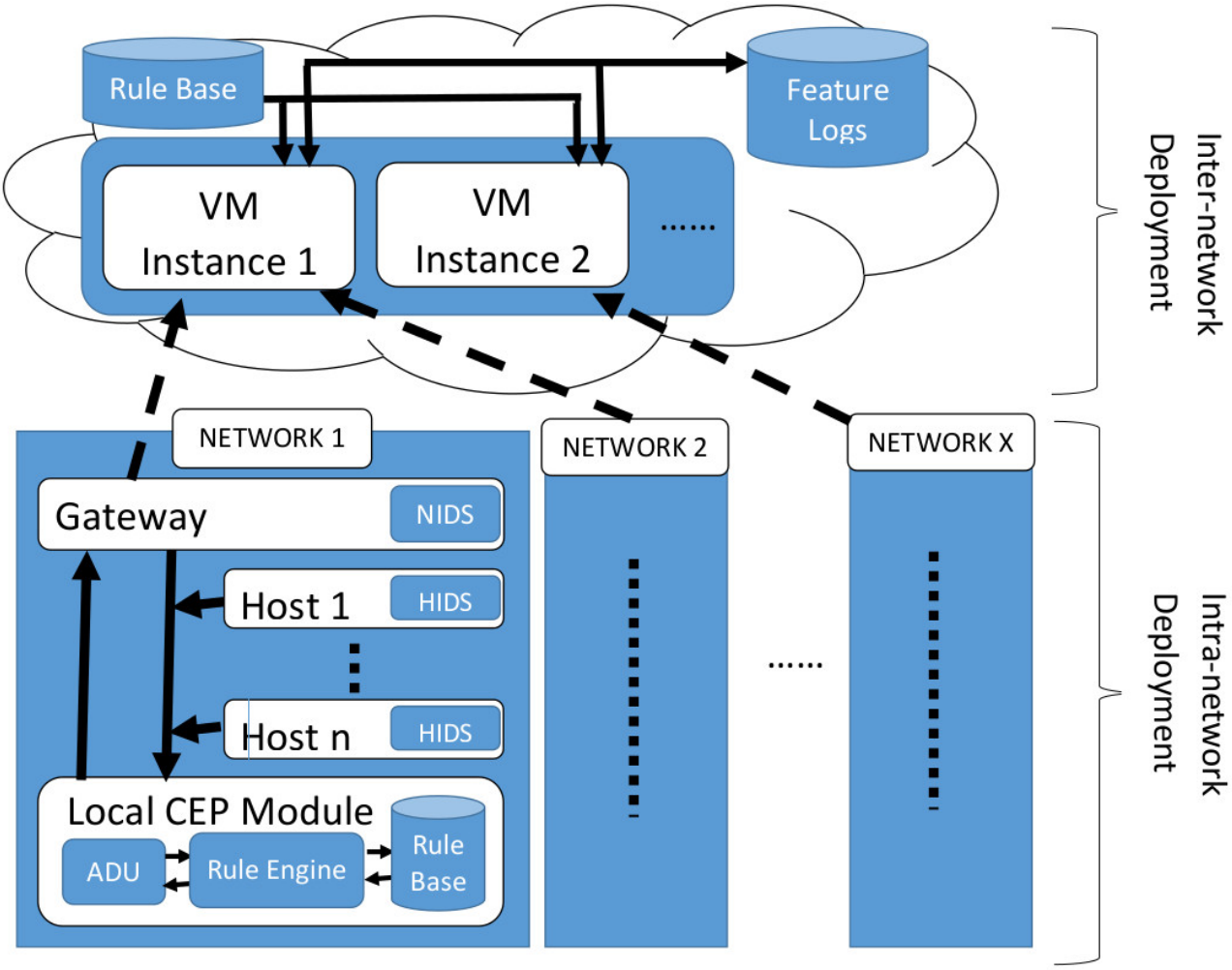
The proposed setup is divided into two portions based on their deployment:

5.1 Intra-network Implementation

This setup consists of the following deployments:

- **Network IDS (NIDS)**: In the NIDS module as shown in Figure 1, we implemented a sub-module to extract 16 unique features from the machine on which the NIDS will be deployed on. This was done with the help of a java library called SIGAR - System Information Gatherer and Reporter [7] Additionally, we not only analyze the system features, but also capture and analyze live network traffic with the help of the jNetPcap library [14]. The analysis includes evaluating data

Figure 1: Proposed Hybrid IDS Framework Diagram



from live packets for the purpose of identifying signature based attacks such as (but not limited to) port scan, SYN flood, and land attack. When an attack is detected, an alert flag is set, and the CEP is sent the feature vectors in consideration. The extracted system features will complement the network traffic to identify different anomalies. For E.g. If the RAM usage spikes up and there are 3000 SYN packets being received we can alert the CEP. But if the RAM usage spikes up and there is normal network traffic then it could just be because of some heavy application. The list of extracted features are as follows:

1. System related features
   - General Features
     * Name of operating system
     * Version and patch of operating system
     * Size of the swap space
     * Firewall status (User group and domain information)
     * List of all the processes and users responsible for them
     * List of all the users on the system along with their privileges
   - Resource Utilization
     * Memory usage in percentage
     * CPU usage in percentage
     * Number of bytes being read from the disk
     * Number of bytes being written to the disk
2. Network related Features
   - List of network adapters with their MAC addresses, IP addresses, downlink and uplink speeds respectively
   - List of all connections, with packets sent/received, speed, start time/end time, type of connection values for each connection respectively
   - Deep inspection of packets to find out which application layer protocols are being used
   - Deep inspection of packets to find out which transport layer protocols are being used
   - Deep inspection of packets to find study header fields of transport and application layer protocols to gain in-depth knowledge

– List of all LAN-external IP addresses

Then, this data will be filtered and sent to the CEP module using Transport Layer Security (TLS). The NIDS module has the additional capability of scanning basic attacks at the end systems so as to avoid transmission overhead. These signature based attacks will be detected at the end user systems itself using SBM, instead of sending the data to CEP and CEP reporting it back to the end users. The Signature Based Model (SBM) has been designed to detect attacks that can easily be observed through packet analysis. Currently, SBM detects three such attacks, Land, SYN-flood, and Port-scan respectively. SBM has been implemented in the following way. First packets captured by the Feature Extraction module are inspected in real time. Packet inspection occurs within a specified time threshold. Upon expiration of the time threshold, data from the analysis is then compared to predefined thresholds to detect the possibility of either of the aforementioned attacks. Should an attack be detected, a new Alert object is created, and the alert is subsequently sent to the Update module, and then in turn sent to CEP. To illustrate concretely, SBM serially accesses packets from Feature Extraction for a period of time T. During this time, SBM tallies the number of unacknowledged (no corresponding ACK) SYN flags and the count of unique ports queried. SBM additionally compares Source and Destination IP/Port numbers to detect a potential Land attack. Should Source/Destination IP/Port numbers match, an Alert is immediately sent to the Update module. Once T has expired, SBM evaluates the live data analysis. ÂăShould the number of unacknowledged SYN flags received be greater than a manually defined threshold value and also 25% greater than a training (machine learned) value, SBM generates an Alert. Similarly, should the number of unique ports of unacknowledged SYN segments be greater than the manually prescribed threshold value, SBM generates an Alert with the appropriate code. A complete documentation for the Host, Network and Mobile IDS can be found here [8].

- **Host IDS (HIDS)**: The HIDS deployment in figure 1.1 is identical to the NIDS deployment and uses jNetPcap for packet capture, SIGAR for local system feature extraction and java serialization to store the Host instances and send them as updates to the local CEP module using Transport Layer Security (TLS).

- **Local CEP Module**: This takes take inputs from host based and network based intrusion detection systems and attempts to make correlations among these inputs to classify abnormal behavior resembling a DDoS Attack using the Multivariate Correlation Analysis Technique (MCA) [2]. ADU refers to the Anomaly Detection Unit. It houses the MCA Algorithm and other anomaly detection algorithms that are to be implemented can be kept there. Multivariate Correlation Analysis, in which the triangle area map generation module is applied to extract the correlations between two distinct features within each traffic record. The MCA implementation as shown in Fig 2 consists of two phases; they are as follows:

  – **Training phase** - The normal behavior of the network is characterized here in terms of Mahalanobis Distance and Standard Deviation as shown in Fig 3

Figure 3: MCA Training Phase [2]

Require: $X^{normal}_{TAM_{lower}}$ with g elements
1: $\overline{TAM^{normal}_{lower}} \leftarrow \frac{1}{g}\sum_{i=1}^{g} TAM^{normal,i}_{lower}$
2: Generate covariance matrix Cov for $X^{normal}_{TAM_{lower}}$ using (12)
3: **for** $i = 1$ to g **do**
4: $MD^{normal,i} \leftarrow MD(TAM^{normal,i}_{lower}, \overline{TAM^{normal}_{lower}})$ {Mahalanobis distance between $TAM^{normal,i}_{lower}$ and $\overline{TAM^{normal}_{lower}}$ computed using (14)}
5: **end for**
6: $\mu \leftarrow \frac{1}{g}\sum_{i=1}^{g} MD^{normal,i}$
7: $\sigma \leftarrow \sqrt{\frac{1}{g-1}\sum_{i=1}^{g}(MD^{normal,i} - \mu)^2}$
8: $Pro \leftarrow (N(\mu, \sigma^2), \overline{TAM^{normal}_{lower}}, Cov)$
9: **return** Pro

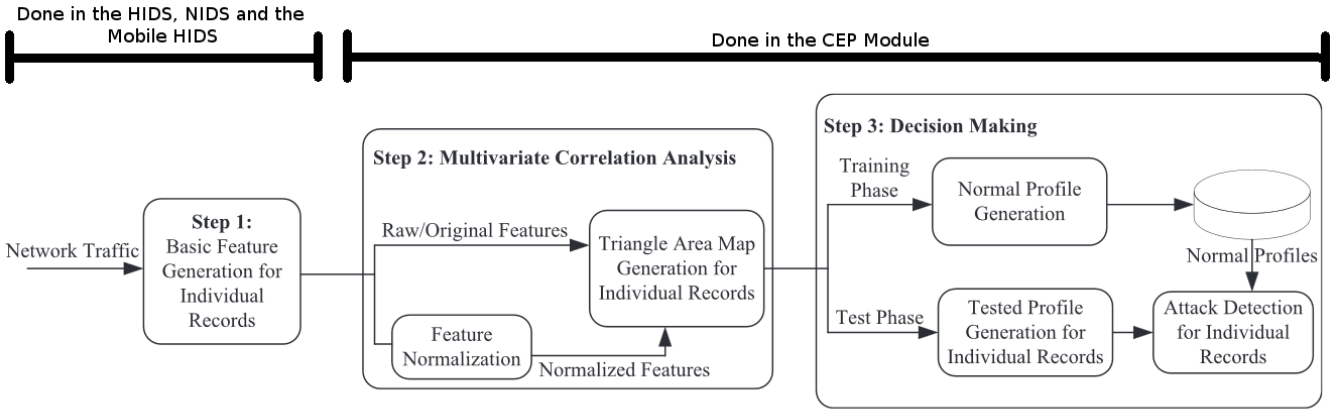
  – **Testing Phase** - As shown in Fig 4, this phase is used to perform anomaly detection. Classification is done based on the MD and standard deviation computed in the training phase. The feature vectors that lie within the range denoted in step 3 of Fig 4, where alpha is in the range [1,3] for normal distribution, correspond to the normal operation of the network, whereas all outliers in the range are classified as anomalies.

Figure 4: MCA Testing Phase [2]

Require: Observed traffic record $x^{observed}$, normal profile $Pro : (N(\mu, \sigma^2), \overline{TAM^{normal}_{lower}}, Cov)$ and parameter α
1: Generate $TAM^{observed}_{lower}$ for the observed traffic record $x^{observed}$
2: $MD^{observed} \leftarrow MD(TAM^{observed}_{lower}, \overline{TAM^{normal}_{lower}})$
3: **if** $(\mu - \sigma * \alpha) \leq MD^{observed} \leq (\mu + \sigma * \alpha)$ **then**
4: **return** Normal
5: **else**
6: **return** Attack
7: **end if**

Thus CEP Rule Engine will be able to detect an incoming DDoS and raise alerts in case of any deviation from the normal working state recorded by MCA. Since scalability is a big concern in our project, the MCA class should be able to run multiple instances in a highly efficient manner. We optimized the efficiency of MCA by using better data structures in our design and by minimizing all redundancies. To reduce the time of data encoding and processing, we decided to send the serialized Host/Network instances [8] instead of sending feature values. This design simplifies the communication process. The MCA algorithm is chosen for the following reasons: The Mahalanobis distance computed as a part of the MCA, yields a better detection accuracy with lower false positives in the

Figure 2: MCA Framework Diagram [2]



dataset used in the reference paper [15]. The MCA algorithm detects anomalies that show a linear variation in features as well The 6% gureKDD Dataset [19] is used to evaluate the MCA Algorithm and the details of the attacks present in it are presented in Table 1 and Fig 5. In addition to the above mentioned tasks, this local CEP module forwards the received features to the IDS deployed in the cloud for further analysis and log keeping.

6. EVALUATION

The evaluation of the proposed implementation have been evaluated as follows:

6.1 Functional Evaluation

The functional requirements are evaluated as follows:

- *Modularity* - We have successfully achieved the optimal level granularity by splitting the proposed system into subsystems and components to ensure ease of modification. This is achieved by modularizing right from the code level, where we split the work of local deployments among several classes as mentioned in our code documentation [8, 15]. Aside the code level modularization, using different deployments for Hosts and Networks and the outsourcing of anomaly detection in a network to a local CEP Module and hosting an IDS as a SaaS to aggregate the data from multiple networks represents the deployment level modularization and is done in a way such that the workload is distributed as evenly as possible without compromise in the security and performance of the IDS.

- *Cost effectiveness* - The major computing in HIDS/NIDS is done in the feature extraction of 16 unique features and live packet capturing and its local analysis before being sent to the CEP. After multiple runs, we found out that HIDS and NIDS implementation require 5-10% of CPU (CPU: AMD FX 6300 - Six Core processor @ 3.5 GHz) resources and 250MB to 700MB of RAM. To derive average attack detection time for SBM, the time for SBM to evaluate aggregated data (this value excludes time threshold, described previously in SBM implementation) and subsequently send or not send alerts was calculated over multiple runs. The average was then computed over those time values, yielding an average detection time of 14 milliseconds for SBM.As far as infrastructure deployment it only requires the cloud service and one additional host per network to house the local CEP module. As the providers cloud service hardware configuration can be dynamically varied as per requirement, thus enforcing the concept of pay for exact usage without the requirement to invest in heavy hardware.

- *Efficiency* - The feature extraction extracts 16 unique features continuously after certain intervals of time. On average the feature extraction takes only 5 seconds on Windows machines and 4 seconds on Linux machines. Additionally, we monitor the live traffic - incoming and outgoing. This process is continuous with 99% packet classification accuracy i.e. whether TCP/UDP/ICMP. Since this deployment enforces the heavy computations only on dedicated machines such as the local CEP module and the Cloud Service it does not compromise the performance of user machines on the network. The bottleneck in terms of the deployments efficiency would be the connectivity factors (Latency, Throughput etc).

- *Ease of deployment* - For Host and Gateway Machines, with easy to install software it makes it possible to deploy the HIDS and NIDS very efficiently. Also, each HIDS and NIDS copy, once installed, will have a unique identifiers hard-coded which will prevent some adversary machine to masquerade any one of the HIDS or NIDS. Additionally, if a duplicate HIDS/NIDS identifier is found at any point of time, the CEP will generate an alert. Only one dedicated hardware deployment per N/W is needed for the CEP module.

6.2 Security Evaluation

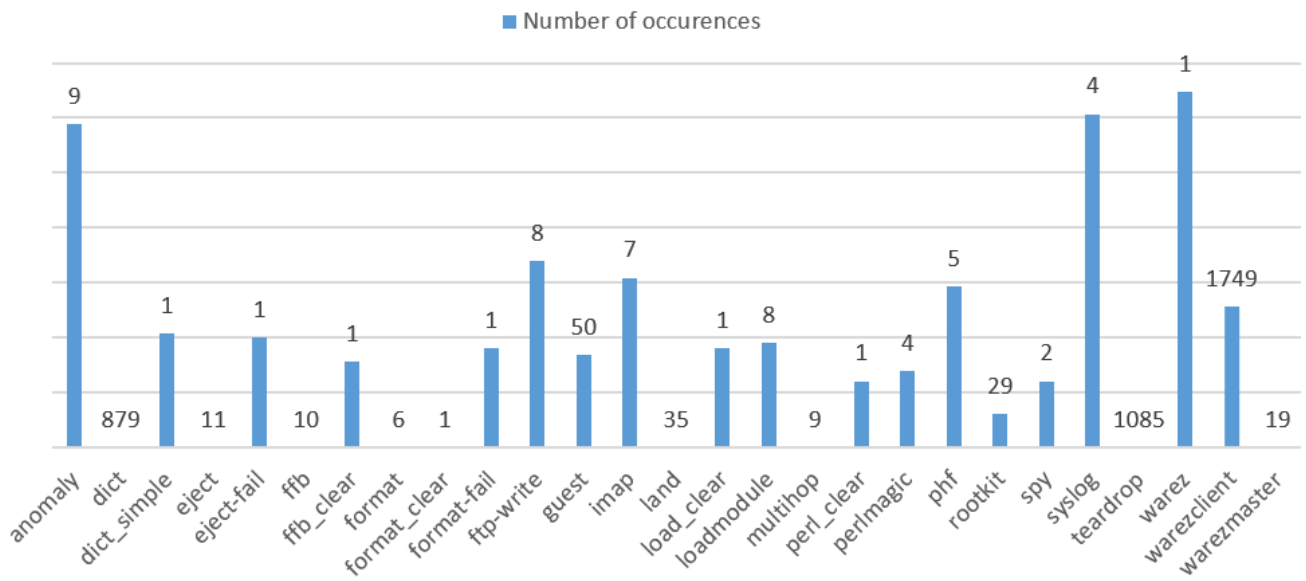
The security requirements are evaluated as follows:

- *SBM Detection* - For each attack in SBM, we used a test bench to evaluate attack detection. Ostinato software was implemented to simulate Land Attack and SYN-flood. 20 runs were used for each attack. SBM

Table 1: List of Attacks present in the 6% GureKDD Dataset

Attack	Description
anomaly	The identification of items, events or observations which do not conform to an expected pattern or other items in a dataset.
dict	The Dictionary attack is a Remote to Local User attack in which an attacker tries to gain access to some machine by making repeated guesses at possible usernames and passwords.
dict_simple	Simple version of dict, just lowercase and check for stopword.
eject	The Eject attack is a User to Root Attack. It exploits a buffer overflow is the 'eject' binary distributed with Solaris 2.5.
eject-fail	A subtype of Eject attack.
ffb	Ffbconfig attack is a User to Root Attack. It exploits a buffer overflow is the 'ffbconfig' program distributed with Solaris 2.5.
ffb_clear	A subtype of Ffbconfig attack.
format	The Fdformat attack is a User to Root Attack. It exploits a buffer overflow is the 'fdformat' program distributed with Solaris 2.5.
format_clear	A subtype of Fdformat attack.
format-fail	A subtype of Fdformat attack.
ftp-write	The Ftp-write attack is a Remote to Local User attack that takes advantage of a common anonymous ftp misconfiguration.
guest	The Guest attack is a Remote to User Attack. It is a variant of the Dictionary attack.
imap	The Imap attack is a Remote to Local User attack. It exploits a buffer overflow in the Imap server of Redhat Linux 4.2 that allows remote attackers to execute arbitrary instructions with root privileges.
land	The Land attack is a denial of service attack that is effective against some older TCP/IP implementations.
load_clear	A subtype of Loadmodule attack.
loadmodule	The Loadmodule attack is a User to Root attack against SunOS 4.1 systems that use the xnews window system.
multihop	An attack against Multi-hop network. Multi-hop network is a wireless networks use two or more wireless hops to convey information from a source to a destination.
normal	Normal state.
perl_clear	A subtype of Perl attack.
perlmagic	The Perl attack is a User to Root attack that exploits a bug in some Perl implementations.
phf	The phf attack is a Remote to Local Attack that abuses a badly written CGI script to execute commands with the privilege level of the http server.
rootkit	A rootkit is a clandestine computer program designed to provide continued privileged access to a computer while actively hiding its presence.
spy	Software that enables a user to obtain covert information about another's computer activities by transmitting data covertly from their hard drive.
syslog	The Syslogd is a Denial of Service Attacks that exploit is a denial of service attack that allows an attacker to remotely kill the syslogd service on a Solaris server.
teardrop	The teardrop exploit is a denial of service attack that exploits a flaw in the implementation of older TCP/IP stacks.
warez	During this attack, a warezmaster or warezlicent logs into an anonymous FTP site and creates a file of a hidden directory.
warezclient	A subtype of warez attack. Slave portion, compromised and under the control of the master.
warezmaster	A subtype of warez attack. Master portion, responsible for control.

Figure 5: Attack Distribution in the 6% GureKDD Dataset



successfully identified each attack simulation with an accuracy 100%. To simulate a Land Attack, packets were formed by Ostinato with the same source/destination IP addresses and source/destination port numbers where SBM correctly identified the malformed packet. A SYN-flood was created by programming Ostinato to send 100,000 SYN packets at a rate of 100 packets/s. After an average of 1,294 packets were sent, SBM successfully detected the attack. Port Scanning was implemented using NMAP stealthy scan. The stealthy scan simply sends unacknowledged TCP SYN packets to probe the network. For each scan tested, SBM successfully detected the port scan within one-time threshold of packet inspection.

- *Communication Integrity* - Enforcing TLS in all the IDS traffic ensures computationally secure authentication and confidentiality. The rules which constantly monitor for any signs of spoofing also help enforce the communication integrity.
- *Cloud reliability and trustworthiness* - The data stored in the cloud is done using SecureDB [10] which provides static data confidentiality and can be operated with TLS to ensure authentication over update related connections. Since only the obfuscated executable is deployed in the cloud, it will be infeasible for the adversary to change the change or view the IDS code even if he gains access to the Cloud Deployment.
- *Small Attack Surface / Less Invasiveness* - This deployment is designed keeping in mind to minimize the Network Attack Surface (By enforcing authentication (TLS), confidentiality (TLS) and internal transparency in tunneling in all IDS related communication) and Software Attack Surface (By code obfuscation, not using non-default port numbers for common services used by the IDS deployment, disabling the standard ICMP for IDS related deployments and using a custom version of ICMP).
- *Need based rules* - The proposed model pulls rules from the cloud based deployment based on the alerts detected. This increases the chances of related attacks being detected at a much earlier stage and in a much more efficient manner. For example, if a particular set of hosts show excessive power usage, then the appropriate power based DoS attack rules are pulled from the cloud for verification at the CEP end.
- *Detection Accuracy of MCA* - The MCA Algorithm was trained for 10000 records and tested over 178810 records in the 6% gureKDD dataset [13]. The results obtained were as shown in Table 2.

Table 2: List of Attacks present in the 6% GureKDD Dataset

	Detection Accuracy	False Positive	False Negatives
Count	174873	0	3937
Percentage	97.7982	0	2.2018

Considering the above stated facts in terms of making use of computationally secure authentication and confidentiality in communication and storage with minimum attack surface, it considerably minimizes the chances of this model being exploited.

7. RELATED WORK

Extensive research has been conducted to improve the scalability of IDS. Various approaches to this problem have been proposed. E.g. Privacy-Preserving Signature-based IDSaaS [3], Parallel design for Network Intrusion Detection Systems [4], and CEP-based IDS for Internet of Things [5]. In the Privacy-Preserving Signature-based IDSaaS [3], cloud computing can provide IDS capabilities such as monitoring

and logging suspicious network behaviors between virtual machines and users. The major advantages of cloud-based IDS are scalability and availability. Users can scale the service based on their demand, and can rely on cloud provider for availability. The main difference of this system and other IDSaaS is that it is a privacy-preserving IDS, which means a cloud provider will not be able to learn any sensitive data during the packet inspection. Thus, the system is still secure in scenarios where the cloud provider goes rouge. This paper focuses uses IDSaaS mainly for offloading the expensive operations, while we propose to use IDSaaS for scalability and as well as offloading. In Parallel design for Network Intrusion Detection Systems [4], TILERAGX36 Processor (a 36-core processor) is used to speed up the detection engine. In this design, they exploit the computational power of many-core processors by using a parallel architecture that combines both data and pipeline parallelism. Each packet is processed in separate cores and the packet processing is divided into several sequential stages. Their experiments showed almost linear speedup and can handle up to 7.2 Gbps traffic with 100-bytes packets. This paper only discusses improving performance of IDS which is directly dependent on the hardware. On the other hand, we implement a system which can handle similar data even without 36-core processors, due to the IDSaaS idea. In CEP-based IDS for Internet of Things [5], authors demonstrate that CEP mechanism is more suitable for applications and services which need to process large among of data. One useful strategy we adapted from this design is that by using relational database for rules, we can improve the efficiency of rule searching. For our project, relational database for rules can also avoid data duplication and inconsistent record. Finally, we studied a paper [6], where the authors propose a novel hybrid detection system referred to as H-IDS, which is comprised of anomaly-based and signature-based detection techniques for more accurate DDoS attack detection. Their proposed detection system can be adopted to various networks with varying traffic patterns due to the flexibility provided through the used decision combiner and the associated sensitivity parameter. They have tested the proposed systems performance against systems based on non-hybrid detection by using two distinct datasets (DARPA and a commercial bank penetration test). The results are satisfactory, which shows that the proposed hybrid system can be an efficient solution in the DDoS detection process. They also found out that some sophisticated DDoS attacks may evade the signature-based detector rules, which are commonly known, and the system performance may decrease as the detection success solely depends on the anomaly detector. Also the training need of anomaly detector stands as a limitation on the overall system performance. The training data may not reflect the real network model in a practical system or even may be unavailable, which may result in decreased performance. We use a better algorithm called MCA which overcomes this problem.

8. FUTURE WORK

We have implemented a full-fledged working system for a local network. We have proposed the extension of this system which would be scalable as well as secure. Future work would include the following:

- Adding full-fledged TLS Support for all IDS related communication.
- IDSaaS Implementation

9. CONCLUDING REMARKS

We have implemented Host IDS (HIDS) and Local CEP Module of this effective and scalable hybrid IDS design and have also proposed a generic hybrid IDS design to bring forth a standardization in the IDS Framework portion of the computer security domain and to evaluate it in terms of performance and security. Implementation of Network IDS (NIDS) is partially complete and we project that it will have been done till the next report. All the modules have been tested independently but testing and evaluation as a whole system is yet to be done. Comprehensive IDS offered as a SaaS (Software as a Service) is proposed for exploring the IDS design space and is part of future work.

10. REFERENCES

[1] R. Mohan, V. Vaidehi, A. Krishna, M. M, S. S. Chakkaravarthy and R. Pathak. *Complex Event Processing Based Hybrid Intrusion Detection System*. 3rd International Conference on Signal Processing, Communication and Networking (ICSCN), Chennai, 2015.

[2] Z. Tan, A. Jamdagni, X. He, P. Nanda and P. R. Liu. *A System for Denial-of-Service Attack Detection Based on Multivariate Correlation Analysis*. IEEE Transactions on Parallel and Distributed Systems, 2014.

[3] Y. Meng. *Towards Designing Privacy-Preserving Signature-based IDS as a Service: A Study and Practice*. 5th International Conference on Intelligent Networking and Collaborative Systems (INCoS), 2013.

[4] H. Jiang. *Scalable High-Performance Parallel Design for Network Intrusion Detection Systems on Many-Core Processors*. ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 2013.

[5] C. Jun. *Design of Complex Event-Processing IDS in Internet of Things*. Sixth International Conference on Measuring Technology and Mechatronics Automation, 2014.

[6] O. Cepheli, S. Buyukcorak and K. G. Kurt. *Hybrid Intrusion Detection System for DDoS Attacks*. Journal of Electrical and Computer Engineering, 2016.

[7] System Information Gatherer and Reporter. `http://support.hyperic.com/display/SIGAR/Home`

[8] Host and N/W IDS Documentation. `http://goo.gl/2ESffv`

[9] Java Serialization. `http://tutorialspoint.com/java/java_serialization.htm`

[10] SecureDB `https://securedb.co`

[11] jDrools Complex Event Processing Engine `http://drools.org`

[12] Java Matrix Package `http://math.nist.gov/javanumerics/jama`

[13] DDoS Datasets `http://si.edu/ant/traces/dataset_list.html`

[14] jNet Pcap `http://jnetpcap.com`

[15] CEP Module Documentation `http://goo.gl/wLxChh`

[16] Gentella Admin Template
http://colorlib.com/polygon/gentella
[17] Jersey Web Services
https://jersey.java.net/
[18] Maven Build Tool
https://maven.apache.org/
[19] GureKDD Dataset
http://www.aldapa.eus/res/gureKddcup/