International Transactions on Systems Science and Applications
Volume 8 ▪ December 2012 ▪ pp. 71-83
sai: itssa.0008.2012.020

ISSN 1751-1461 (print)
ISSN 2051-5642 (online)

Cloud Network Security Monitoring and Response System

Murat Mukhtarov, Natalia Miloslavskaya and Alexander Tolstoy

Information Security Faculty
National Research Nulcear University MEPhI
Moscow, Russia
Email: muhtarov.mr@gmail.com

Abstract: The public clouds network monitoring and response system, based on flow measurements, open source tools and CSMS (Cloud Security Monitoring System) module, is introduced in the paper. The main goal of the research is to develop a set of algorithms and to implement a system, which automatically detects and makes a response to network anomalies, occurring inside a Cloud infrastructure. An approach of anomaly detection inside the Cloud infrastructure based on a profiling method of IPFIX (IP Flow Information Export) protocol data is proposed. An idea of a negative selection principle is used for generating signatures of network anomalies, named detectors. The automatic response module makes a decision on a network anomalies origin based on several iterative checks and creates a record on the firewall rules table. The network traffic profiling process automatically generates the firewall rules set for all traffic classes, obtained during the learning process. Main results of the research are development of the algorithms and the way of the monitoring network attacks inside the Cloud. Implementation of the algorithms is python-based script and at present it is under a hard-testing phase.

Keywords: cloud computing; cloud infrastructure; virtual infrastructure; application hosting; network security.

1. Introduction

Cloud computing is a novel way to provide customers with Information Technology services, but with virtualization technologies in the background. Cloud computing uses networked infrastructure, software and computing power to provide resources to customers in an on-demand environment. With cloud computing, information is stored remotely in a centralized server farm and is accessed by the hardware or software thin clients that can include desktop computers, notebooks, handhelds and other devices. Typically, Clouds utilize a set of virtualized computers that enable users to start and stop servers or use compute cycles only when needed (also referred to as utility computing) [1]. In terms of information security (IS), the cloud computing threat model consists of three fundamental issues: availability, integrity and confidentiality violations. Availability is terminated via Denial of Service attacks. The likelihood and easiness of these attacks will increase as the volume of information exchanged between a user and a cloud provider increases. Integrity issues arise due to the fact that users must be sure that the information they retrieve is the same as that they store. This is a difficult task for one reason: information changes over time as do users themselves. But, also, it is important to separate users' information from the production information (for example configuration files, system files integrity, and so on). Finally, confidentiality issues may take place, for example over (accidental) disclosure of information to third parties or because of aggregation. Most computer compromises result in information leakage, so this is also an important issue [2].

1.1 Research Subject

In this research availability is a main issue and the other issues that arise from it, so they are subsidiary risks for us. One possible way for Cloud networks to be monitored is to use a network telemetry principal with such protocols as Cisco Netflow [3] or IPFIX (Internet Protocol Information Export) [4]. Design of the open source virtualization technologies provides an opportunity to use Netflow/IPFIX probes on a hypervisor without performance reduction. IPFIX protocol has some advantages while being compared with the Netflow; it is not proprietary, it is open-standard and has improvements [5] that can be used in open source systems such as Linux or BSD (Berkley Source Distribution) -derivate systems. IPFIX is a lightweight network monitoring protocol for the connection control and volume-based traffic estimation [6].

In this research an approach to profile IPFIX data in such environment as a Cloud infrastructure and also suggest ways to make an automatic response to the detected anomalies inside a network is proposed. Our solution is based on combination of two techniques – estimation of maximum entropy [12] and artificial immune system algorithms. This combined approach allows increasing accuracy of anomalies detection in distributed environments with big amount of different network traffic patterns, for example in hosted Cloud services [6].

The way described in the paper is applicable to the Cloud solutions that provide their customers with such services as Infrastructure as a Service, Platform as a Service. In other words a Cloud Infrastructure consists of large amount of virtual machines running inside a virtual infrastructure based on physical servers and network equipment. Implementation of several algorithms and principles of monitoring that serves as a background for the Cloud infrastructure monitoring system is introduced in the paper. Important aspect of any monitoring is an ability to perform response actions against registered deviation. An approach of making decision about malicious traffic based on heuristic triggers that reduces number false positive errors is proposed.

The described system monitoring approach is focused on network security monitoring and response actions inside a Cloud. The main advantages of the CSMS approach are compatibility with the majority of operating systems and network equipment due to IPFIX protocol, ability of an automatic response to a network attack and ability of identifying unknown network anomalies in some cases. Across all paper it is mention that main source of network traffic is IPFIX data. The way of obtaining these data described in [6] in detail.

1.2 State of Art and Related Works

The main focus of the paper is a network security monitoring approach in a Cloud infrastructure. Some network security threats and issues that may occur in the virtual infrastructure clouds are discussed. All of them use shared hardware, network [1] and hypervisor's resources [2]. Several authors suggest their own way to increase IS level inside the Cloud infrastructure.

Security threats related to hosting application in a Cloud Infrastructure are covered by Molnar and Schechter [7]. The researches compare traditional and cloud hosting focused on IS threats. The Virtual Local Area Network (VLAN) separation technique on a Cloud Infrastructure is mentioned by Berger's et al. [8]. They suggest a way of increasing virtual infrastructure security by using a strong security policy inside a cloud infrastructure – Trusted virtual data center (TVDC). Their idea is based on the research of Bussani, et al., Trusted Virtual Domains (TVD): Secure Foundations for Business and Information Technology Services [9]. The main idea of TVDC is a strong isolation and integrity guarantee in virtualized, cloud computing environments [8]. To achieve this isolation researchers use network separation techniques based on IEEE 802.1q [10], memory control techniques and "colorizing" each data flow inside a cloud. Another approach to a Cloud infrastructure monitoring called "Private Clouds MONitoring Systems" (PCMONS) was created by Chaves, et al. [11]. Their main goal was to develop a modular and extensible monitoring system for the private Clouds. PCMONS is implemented as a module for the open source monitoring system Nagios and is compatible with the open source IaaS platform Eucalyptus [11]. But, it has several disadvantages: as PCMONS is a Nagios module, it inherited Nagios performance and scalability issues that eliminate

applicability to the huge Cloud infrastructure; also it is compatible only with one solution. In [13] Monfared analyzes weakness in existing Cloud environment monitoring mechanisms and suggests way of mitigating them. He analyzes several commercial and open-source solutions of the Cloud infrastructure security monitoring systems and approaches. Author concludes that security model of highly distributed Cloud systems not matured and security monitoring mechanisms need sensitive development [13]. Researchers from Fujitsu security in [14] suggest security architecture for Cloud computing where they propose several improvements of the Cloud infrastructure to make it more secure and increase trust level.

Analysis of related studies highlights problem of demanding in security monitoring systems and solutions for the distributed Cloud infrastructure. In this work one of the possible ways to monitor Cloud infrastructure that satisfies several requirements mentioned in the paper is suggested. This approach is suitable for the range of private Clouds due to usage of an open-standard network traffic measuring protocol.

2. Classification and Detection Deviation in Network Traffic inside Cloud Infrastructure

2.1 Traffic Classification

A network traffic is suggested to be divided in groups according to its purpose and frequency. Proposed classification is an example based on analyzing of network traffic patterns in our campus datacenter network. Some of the classes are common for the rest of network environment; among them are http, dns and mail protocols traffic. Some groups are specific for given Cloud network environment, for instance RDP or VNC traffic. Figure 1 shows traffic classification being used in CSMS implantation for our campus datacenter.

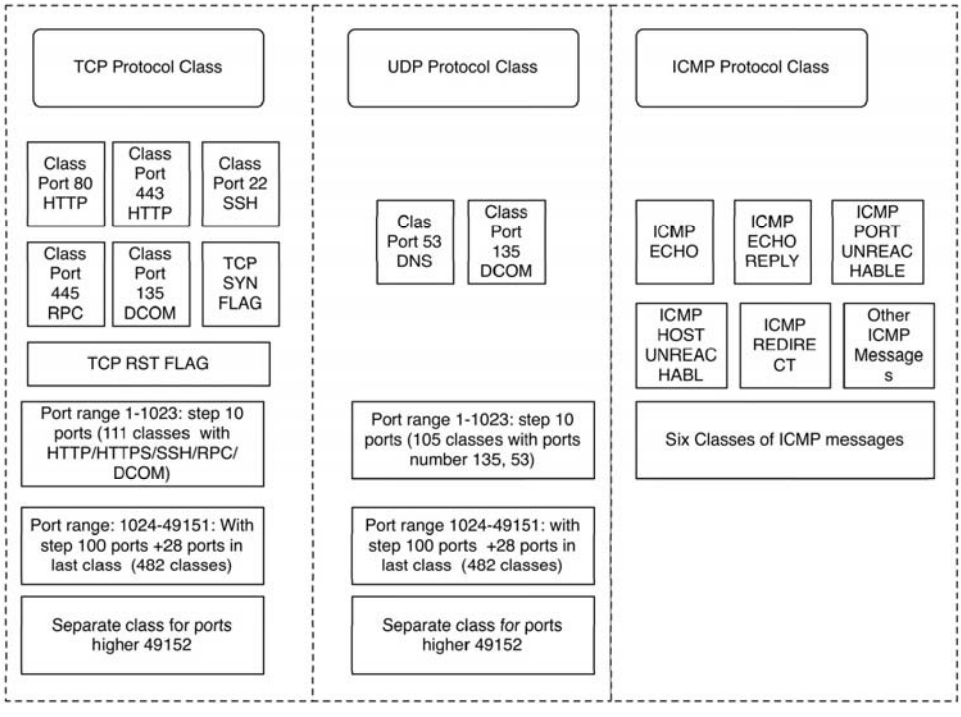

Figure 1. Network traffic classification.

All transport layer traffic is divided by transport protocols TCP and UDP and ICMP protocol in separate classes. Inside these three classes there are several subclasses: separate class for protocols that belongs to TCP: HTTP, HTTPS, DCOM, RPC, SSH; separate classes for UDP frequent protocols: DNS, DCOM. In addition, TCP packets with SYN flag were put in the separate class and with RST flag – as they indicate TCP session phase start and end. All other ports are divided in groups with step based on 10 ports (starting from 1 to 1023 server ports), step based on100 ports (starting from 1024 up to 49151) and separate class for all other ports. Such a division is made both for TCP and UDP ports. Resulting groups contain all network traffic obtained through network traffic collector. As described in our previous works information of a network traffic is obtained in real-time from IPFIX sensors. IPFIX sensors should be placed on each monitoring host and capturing is performed against interface of physical server where Virtual machine instances inside Cloud Infrastructure are hosted [6].

2.2 Network Traffic Profiling Based on Modified Maximum Entropy Approach

Maximum entropy approach as a method for analyzing network traffic was introduced by Gu and McCallum [12]. They used it to perform network traffic profiling and to identify deviations in a network traffic profile. This approach is suggested to be improved and to be actualized for modern traffic volumes. Authors noticed that method mentioned above is very suitable to be used for processing data obtained from flow-based measurement protocols like Netflow or IPFIX [6]. All data processed through the IPFIX collector should be classified according to the described above class map. Then profiling on obtained data set should be preformed. It could be done for the given period of time that depends on a network traffic behavior. For instance in an Internet service provider network one business customer network profile is very similar during all work days. So traffic profile of work hours of the day could be used as the baseline profile. During this phase given baseline traffic distribution should be profiled among classes of network traffic with maximum Entropy estimation approach.

To formalize our approach the following statements is used. Let Ω be an array of traffic classes ω, baseline set of network traffic obtained from IPFIX - $S=\{x_1,\ldots,x_n\}$. Consider that set of indicator functions are given. Indicator functions match traffic classes $F=\{f_1,\ldots f_i,\ldots\}$. Each traffic class describes indicator functions in the follow way:

$$f_i(x_i \in \omega) = \begin{cases} 1, if\ packet\ in\ class\ \omega \\ 0, if\ packet\ not\ in\ class\ \omega \end{cases} \quad (1)$$

It is obvious that a set of F is the set of indicator functions **1** (x), which equals 1 or 0, respectively. Observing distribution of IPFIX flows among network traffic classes will have a following form, where P'(ω) distribution of obtained network traffic, divided by traffic classes given above:

$$P'(\omega) = \sum \frac{1(x_i)}{n} \quad (2)$$

To profile network traffic distribution it is important to find distribution P(ω) which should satisfy equation (3) and should have a maximum entropy:

$$E_{P'}(f_i) = E_P(f_i),\ for\ \forall f_i \in F \quad (3)$$

In [12] it is mentioned that this problem is equivalent to maximum likelihood estimation problem of statement (4), where Z is the power of array |Ω|, P is the baseline distribution which should be found and $\lambda_i \in \Lambda$ are weights of the indicators $f_i(\omega)$:

$$P(\omega) = \frac{1}{Z} e^{\sum \lambda_i f_i(\omega)} \quad (4)$$

To find distribution which accurately explains profile it is required to add regular indicator g inside our model and to estimate whether it improves this model. To do it weight coefficients of the indicators should be modified according to the previous equitation:

$$P_{i,\lambda_i,g}(\omega) = \frac{1}{Z'} e^{\sum \lambda_i f_i(\omega)} e^{\lambda_g g} \quad (5)$$

After adding a regular indicator it is important to modify weights of the set of indicators. This problem is formalized as minimization of the divergence between observing distribution and sought-for network traffic profile:

$$\Lambda = \min_{\Lambda} \sum P'(\omega) \log \frac{P'(\omega)}{P(\omega)} \quad (6)$$

This problem could be solved with some of optimization algorithms. The comparison made in [16] is used, where it was concluded that two methods have best performance: quasi-Newton optimization and gradient descent (GD). After such a comparison two opportunities quasi-Newton optimization implementation of Broyden-Fletcher-Goldfarb-Shanno (BFGS) was done. To achieve best performance C-library liblbfgs is used [17]. The choice is based on possibility of usage library in a GNU C compiler.

Design of flow-based measurement protocols aggregates captured traffic in particular structures called flows [4]. Due to this feature of IPFIX protocol it is possible to improve performance of maximum entropy approach significantly. Profiling is proposed to do on a set of IPFIX flows instead of separate packets. Aggregation and sampling features allow to work with a set of flows and to share calculations between sources of IPFIX data (e.g. routers, firewalls or software sensors) and IPFIX data processing engine. Hence, amount of iterations depends directly on export frequency parameter defined in packets; a default value is set usually as 20 packets. It means that performance of algorithm increases at least 10 times (for outgoing and incoming traffic on interface, respectively).

To explain the idea of distribution and profiling an example of matching traffic class and indicators is shown in Table 1. Each captured flow in a particular traffic class is described in the table. The resulted distribution performs maximum entropy approach to obtain most significant traffic class that describes current network traffic profile.

Table 1. Matching traffic class to indicator function.

Protocol	TCP	TCP	TCP	TCP	UDP	UDP	ICMP
Port num, flag	80, ACK	443, ACK	80, SYN	161, ACK	53	53	echo
Indicator function	f_{HTTP}	f_{HTTPS}	$f_{HTTP,SYN}$	f_{TCP161}	f_{DNS}	f_{DNS}	f_{echo}

After putting traffic classes from given example into equation (4) the result will have the following distribution:

$$P = \frac{1}{2000} e^{f_{http}\lambda_{http}+f_{https}\lambda_{https}+f_{http,syn}\lambda_{http,syn}+f_{tcp161}\lambda_{tcp161}+f_{dns}\lambda_{dns}+f_{dns}\lambda_{dns}+f_{echo}\lambda_{echo}} \ldots \quad (7)$$

Due to the usage of flow-based measurement of a network traffic some improvements were obtained. Among them are unified source of network traffic data and increase of performance in maximum entropy approach at the same time of high accuracy of this method, which has ability to identify very tiny abnormal fluctuations in a network traffic profile.

2.3 Generating Anomaly Detectors and Anomaly Detection

Another approach proposed in the paper is a special way of generating a network anomaly detector. The idea of this approach lies in a negative selection algorithm, introduced by Forrest et al. [18]. According to the negative selection, a network traffic profile, which is returned by the IPFIX data-profiling (normal behavior profile), could be modified in the manner proposed below. Creation of the set of anomaly detectors based on negative selection is proposed. To create a set of potential anomalies detectors, the volumes of the traffic classes in a normal behavior profile with the random values in the range of Lower_Tr and Upper_Tr variables were increased. Also there are several settings for the detector generating process: the amount of detectors needed, the amount of affected positions in a profile and a threshold value of divergence. In other words, a network traffic profile is a vectors of values of volume of network traffic $Q=\{q_1,\ldots,q_n\}$, where n is an amount of classes that

were taken in the normal behavior profile in the step mentioned in the previous section and q_i is a volume value of a network traffic of given classes. The criteria of choosing resulting profile is based on divergence calculation (6). Minimum divergence value is a threshold variable that should be set according to the amount of a network traffic in the given profile:

$$D_i = \sum P(\omega) \log \frac{P(\omega)}{Q_i(\omega)} \quad (8)$$

where $Q_i(\omega)$ is a result of modifying values inside normal behavior profile and $P(\omega)$ is the normal behavior profile. An algorithm of generating anomaly detectors, based on this proposal described in a Figure 2, was developed.

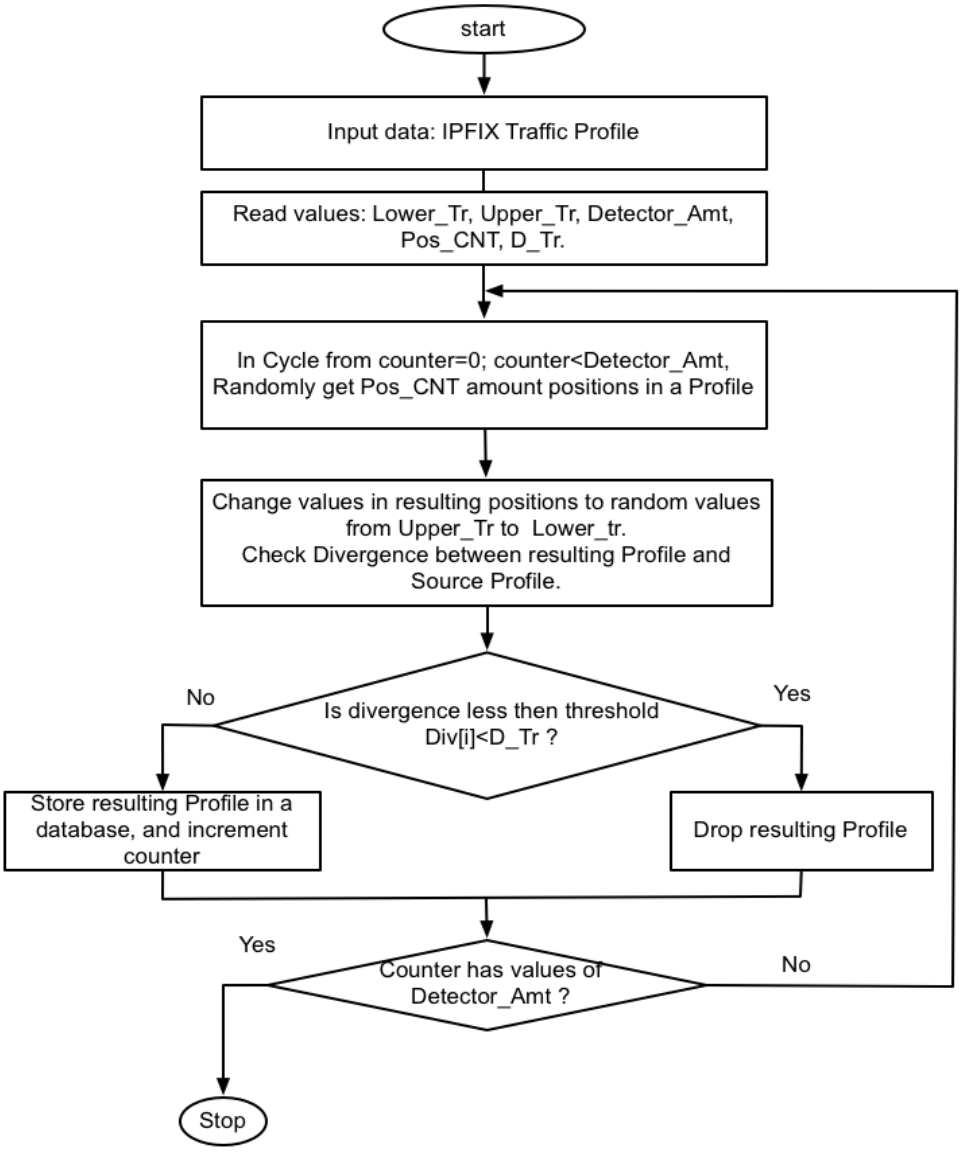

Figure 2. Anomaly detectors generation algorithm.

The initial data for the algorithm is a set of IPFIX flows that was chosen as the normal behavior profile. Algorithm should be given the following variables: Lower value of modification traffic

volume, Upper value of modification traffic volume, divergence threshold value, number of positions to be modified and amount of detectors. The algorithm randomly changes values of positions of the network traffic classes inside a profile according to the values of the Lower_Tr and Upper_Tr. The stopping criterion is an achievement of the required number of anomaly detectors. Detectors that are similar to the normal behavior profile should be dropped. All other detectors should be stored inside the database. Reasonable amount of detectors depends on the performance of a storage engine. In this research usage of non-sql databases such as mongodb [19] to store network detectors due to performance issues of RDBMS storages is suggested.

3. Core Algorithms of Cloud Monitoring and Response System

3.1 Network Traffic Profiling Algorithm

To monitor the network traffic anomalies, that in fact are the result of DDoS-attacks or abuse traffic, a way that will be applicable to the implementation inside a network of a Cloud Infrastructure should be found. Several requirements to this approach were worked out:

(1) To be informative enough to analyze network traffic volumes by traffic types;
(2) To be lightweight;
(3) To be easy to spread through a Cloud infrastructure network and
(4) To not impact production network performance.

As mentioned above the best way that satisfies all these requirements is to use flow-based measurement protocols like Netflow or IPFIX [6]. Here the IPFIX protocol is used as it is an open standard protocol.

To profile IPFIX data a maximum entropy estimation approach, introduced in [12] and [15], is used. Modification and improvement an algorithm of profile estimation are needed to make it applicable to IPFIX data analysis (Figure 3). For designing an algorithm classifying of a given pattern of a network traffic should be done. Network traffic classification process is needed because traffic patterns usually consist of large amount of the different traffic packets and storing profile of raw traffic data will require large amount of disk space. Therefore, large volumes of data will require more processor time for processing. So, usage of preprocessing classification algorithm is proposed, allowing to work with volume-based estimation of network traffic data divided by classes. The result of IPFIX flows preprocessing is a significant reduction of the size of data which should be processed by a monitoring system. Amount of traffic classes should be selected by user. Also, an expert should exclude “anomalies” if they are present in a given pattern.

This algorithm checks in a cycle each traffic class with maximum entropy approach and estimates weights of each traffic class in a model. The algorithm’s result is the network traffic profile in which only the most significant traffic classes in a given pattern are stored.

The algorithm stops when the next traffic class does not improve the profile enough, in other words the decrease of divergence should be less then threshold value.

To adapt the algorithm to a Cloud Infrastructure network, here is proposed to make some special traffic classes that are inherent to the network of a Cloud infrastructure. HTTP, HTTPS, DNS, SMTP, POP3, IMAP, POP3S, IMAPS, RDP, VNC, SQL ports are placed in the separate classes. After cycle pre-check with a given pattern an exclusion of a network traffic class from sampling process is proposed. This improvement allows to avoid additional checks of the network traffic patterns due to IPFIX protocol input data format – measurement based on flows.

3.2 Detecting Network Anomalies with Proposed Approach

Anomaly detection is based on the set of detectors, recorded in the database. Figure 4 shows anomaly detection algorithm with several heuristic checks that avoid number of false-positive errors. Functional features of particular Cloud infrastructure was taken into an account, especially in part of network traffic profiles. Probes are exporting flows of the IPFIX data to the collectors. A collector consists of two parts: the first one normalizes incoming data from the probes, and the second one

performs a comparison of a captured traffic pattern with an anomaly detector. Each anomaly detector has “time-to- live” (TTL) attribute. Normally this value set to one month. If a detector never matches any of the captured traffic patterns within one month it will be marked for deletion and it would be dropped at the end of the next month.

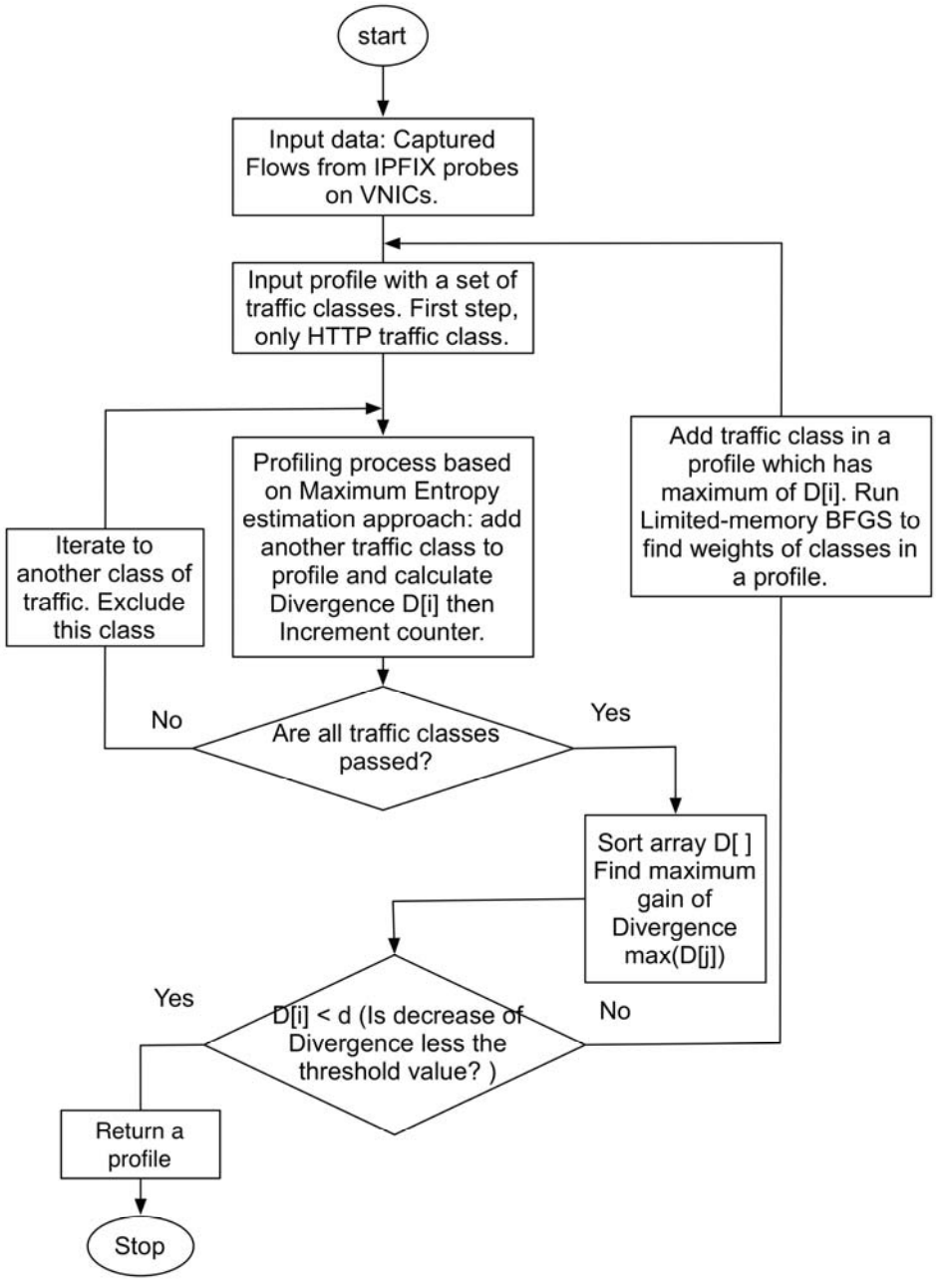

Figure 3. Block diagram of IPFIX data profiling.

One month period seems to be reasonable to reduce impact on the monitoring system performance and limits number of detector, but depends on user settings. Deletion of the detector

does not mean that network anomaly which should be covered with deleted detector would not be handled properly. Generating of the anomaly detectors is a pseudorandom process which allows the possibility of the collisions. So such kind of anomaly possible could be found with detectors from another generation with some probability.

Another case is when a detector matched some of the network traffic pattern. This detector changes its TTL ("time-to-live") attribute to one year and spreads it across all probes. Hence, it is possible to clean our detectors database from the patterns that would be never observed in the network traffic and collect patterns that are really useful for anomaly detection.

Figure 4 introduces our algorithm of anomaly detection and response actions. IPFIX information has several attributes referred to the IP packet header data. When a network anomaly is detected, a Cloud monitoring system could tell us what kind of traffic causes an anomaly. In this case, it is possible to find out a source IP address of anomaly traffic and block it inside a firewall.

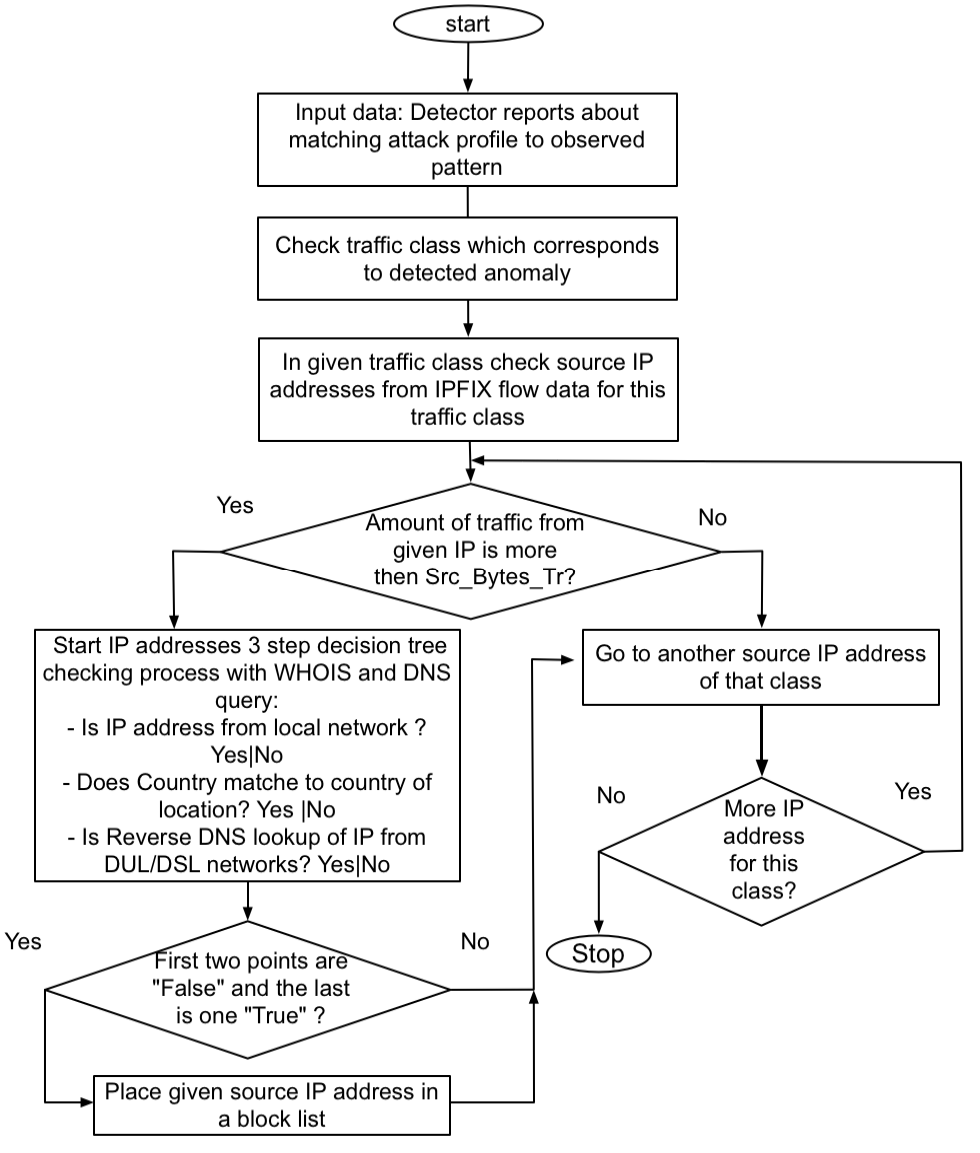

Figure 4. Block diagram of monitoring and response process.

To increase accuracy of the blocking system and to preserve a normal traffic there is used "Whois" Database queries to learn an origin of IP address, location and reverse DNS queries to estimate the purpose of IP address usage. Several checks are also performed: "*Is the IP address from our network?", "Is the IP address from our Country?", "Is the IP address from Digital Subscriber Line (DSL)/Dial Up Line (DUL) network?".* So, if the IP address is outside of a Cloud network and as well as a region, that it is probably could be a reason for an anomaly and it going to be blocked for an hour. Same action is performed if the IP address is from DSL/DUL networks.

4. CSMS Architecture and Deployment

4.1 IPFIX Sensor Placement

As mentioned above it is possible to capture a network traffic and to export IPFIX data on physical or virtual interfaces of deployment virtual machines inside the Cloud infrastructure. Usually several virtual servers are running on one physical server and each virtual machine is connected to the virtual switch (that is implemented via the bridge support in Linux kernel). A virtual interface usually is created if a virtual machine has access to a Cloud provider customer network (for example an Internet access connection). Sensors are proposed to be put on a physical or virtual interface in a manner shown in Table 2. The physical interface is putted in a tagged mode and the virtual local network interfaces spread across virtual interfaces of virtual machines.

Table 2. Probe placement example.

Virtual Machine	VM1	VM2	VM3	VM4
VLAN ID	VLAN100	VLAN200	VLAN300	VLAN400
Probe placement	*IPFIX probe*			
Physical interface	Hypervisor physical interface eth0			

IPFIX probe captures traffic that travels via these interfaces and exports data to the remote collector in a form of flows. Flows are stored in a relational database and are analyzed by a processing engine in real time. In this work a processing engine is called CSMS controller.

4.2 CSMS controller

CSMS Controller is a main component of CSMS system. It consists of several components: IPFIX collector, storage of IPFIX data and detectors and response module. To prepare CSMS controller for working in a new environment it should firstly be feed with sample IPFIX data that will be used to prepare normal behavior profile according to algorithm described above (Figure 5).

IPFIX collector receives IPFIX data from remote probes and listens UDP socket of operating system (default port used 9996). In collector all IPFIX data is preprocessed and is prepared in the form convenient for profiling. All IPFIX data is aggregated, summarized and submitted to a profiler. CSMS Controller main process should be run with "learn" and "feed" flag, as argument should be given captured IPFIX data in .pcap file format. Learning and profiling process could be done in a background mode parallel to the main process of anomalies monitoring. It allows to reconfigure and to adjust network traffic profile regarding to the current conditions of a network.

It is possible to store several profiles, for instance organized per network, per customer or per virtual server profile. Profiles are stored in database engine (currently only MySQL database is supported, but in future we are going to support PostgresSQL also). Detector generating process could be also performed in a background or it could be run on another server, hence it is possible to share recourses consumption. To start detector generating process csmsd daemon should be run with "detector" flag.

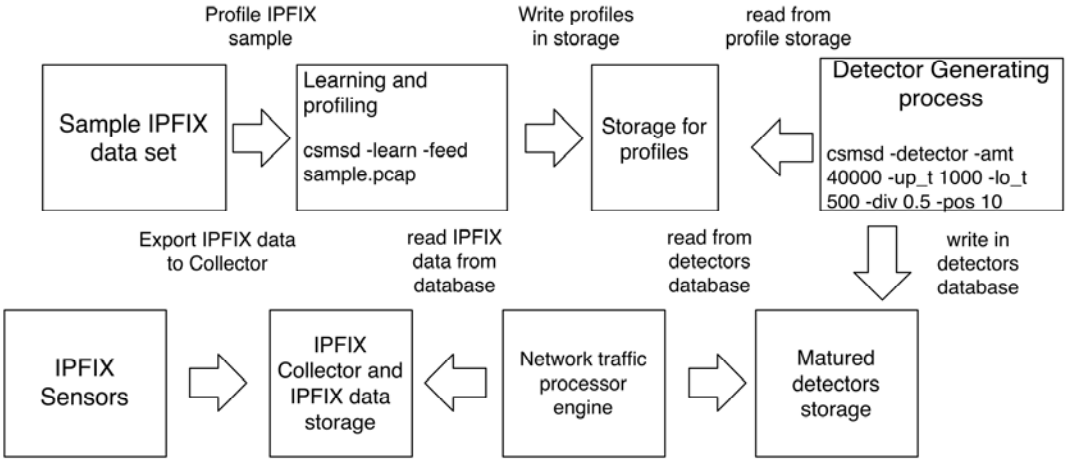

Figure 5. CSMS controller functioning scheme.

Matured detectors are stored in mongodb. They could be stored also in mysql database, but with performance reduction. MongoDB storage size depends directly on physical memory of a server where a controller process is executed.

The network traffic processor compares matured detectors with observing data. It increases TTL of detectors in case of successful detection of anomaly and drops detectors when TTL ended. In case of successful detection a response module make response action to the remote firewall. Currently any firewall is supported by means of an external shell script. It is possible to set external action in case of taking decision on anomaly traffic.

Scalability of a system depends on system recourses of a computer where csmsd process is executed and easily could be done by deploying new IPFIX sensor. Own implementation of IPFIX sensor is used, but our solution is fully compatible with any IPFIX probe and Cisco Systems Netflow v9 protocol also.

5. Integrating CSMS in Existing Cloud Infrastructure

Demonstrated deployment example shows integration process of CSMS module and IPFIX sensors inside typical cloud datacenters. For example, the Cloud infrastructure has been already deployed and based on the open source private Cloud Eucalyptus as shown on Figure 6.

Eucalyptus Cloud Controller usually runs on a Linux-based computer with two network interface cards (NIC). Cloud Controller is a front end of the Cloud Infrastructure and it divides network into two parts: public local area network (LAN) (on Figure 6. Public switch) and private LAN (on Figure 6. Private switch). It is suggested to deploy CSMS module on Cloud Controller as it is a central part of the Cloud Infrastructure and it connects both private and public networks. Also, it is suggested to deploy firewall equipment, which is connected to the Public switch and is able to block outside IP addresses in case of receiving corresponding command from CSMS.

IPFIX sensors is deployed in the Cloud Infrastructure Nodes (component of the Cloud where Virtual Machines of the End User runs) and it sends information to the IPFIX collector, which is also deployed on the Cloud Controller. In addition, IPFIX data is exporting from the border routers. CSMS module analyzes incoming data from the several sources (Nodes and Routers) and performs anomaly recognition actions – compares anomaly detectors patterns against observing network traffic data. In the case when anomaly is discovered, CSMS performs IP address check process to be sure that this traffic is not from own or trusted networks and then sends command to the firewall equipment in order to block malicious IP address. Advantage of this approach lies in possibility to deploy IPFIX sensors in every operating system which supports traffic capturing. It means that no

matter which kind of Cloud or Virtualization technology is going to be used, the most important is the ability to export IPFIX data from network equipment or from virtual network interfaces of the Cloud nodes.

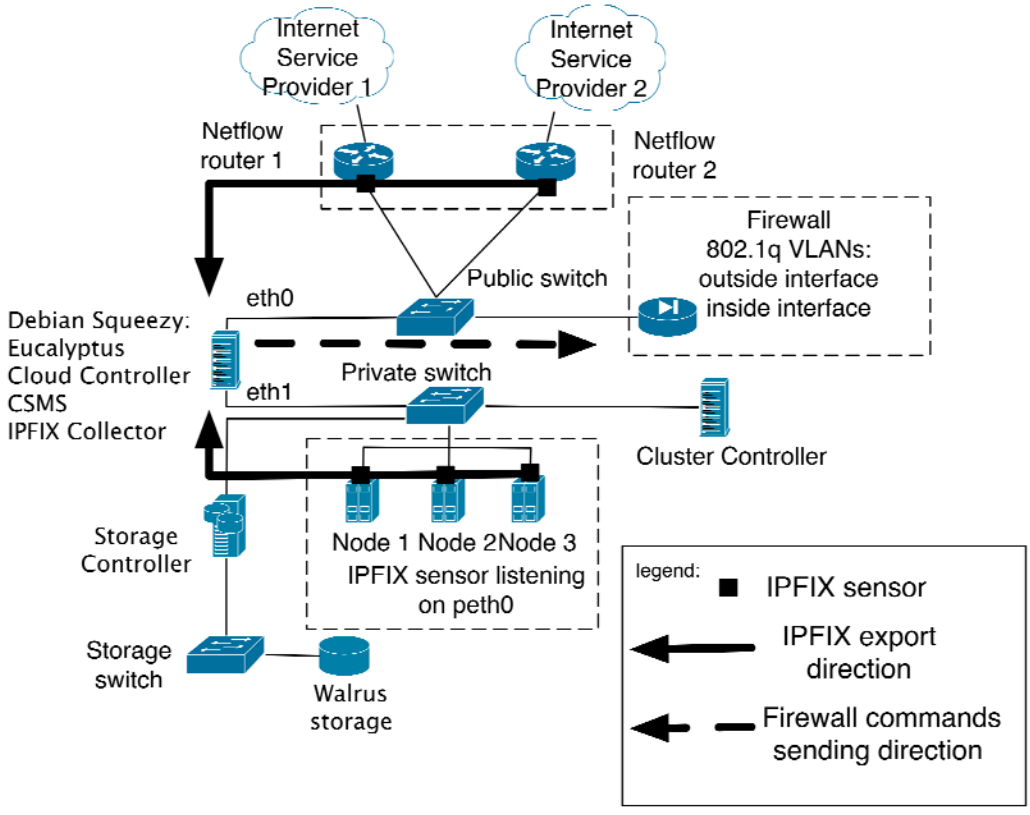

Figure 6. CSMS deployment inside Eucalyptus Cloud infrastructure.

6. Conclusion and Future Works

Flow-based measurement protocols such as IPFIX are an appropriate source of network traffic information, which allows to analyze traffic with statistical frameworks and approaches. In the paper the maximum entropy estimation approach is used to obtain the normal behavior network traffic profile based on IPFIX data. This way of monitoring network security is more productive and easy to implement in existing Clouds due to design and implementation of open source-based virtualizing software. The suggested approach of anomaly detection based on negative selection algorithm seems to be an appropriate way of monitoring in distributed environments such as a Cloud infrastructure network. It is ready to detect DDoS-attacks and other abuse traffic attacks, having an availability issue for Cloud computing as a main concern. Automatic response ability of the CSMS with the "Whois" and reverse DNS information, based on source IP address filtering, is a useful way to preserve customers from false-positive errors.

The future developments of this research are testing and implementing of proposed algorithms and approaches to find a suitable way of integrating them inside the existing open source Cloud infrastructures. Also an applicability of the described proposal to the network attacks should be analyzed. And issue of IPv6 protocol compatibility arises in last time due to exhausting of IPv4 address space. Currently researches should think about modifying IPFIX/Netflow templates to make

it compatible with IPv6 connections. Future work also should concentrate on IPv6 compatibility and support.

References

[1] Securing the Cloud: A review of Cloud Computing, Security Implications and Best Practicies http://www.savvis.com/en-us/info_center/documents/savvis_vmw_whitepaper_0809.pdf

[2] Schoo P., Fusenig V., Souza V. at el. Chanlanges of Cloud Networking Security. 2nd International ICST Conference on Mobile Networks and Management, September 2010 Santandar Spain (October 2010). HP Laboratories, HPL-2010-137.

[3] Claise B. RFC 3954 Cisco Systems Netflow Services Export Version 9, 2004.

[4] Claise B. RFC 5101 Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information, 2008.

[5] Boschi E. and Trammell B. Bidirectional Flow Measurement, IPFIX, and Security Analysis. 2006, pp. 8-10.

[6] Mukhtarov M., Miloslavskaya N., Tolstoy A. Netowrk security Threats and Cloud Services Monitoring. ICNS 2011 Venice/Mestre Italy, 22 May 2011, pp. 141-145.

[7] Molnar D., Schechter S. Self Hosting vs. Cloud Hosting: Accounting for the security impact of hosting in the cloud. WEIS 2010, pp. 149-164

[8] Berger S., Caceres R., Goldman K. and others Security for the cloud infrastructure: Trusted virtual data center implementation. IBM J. RES & DEV. Vol. 53, No. 5, paper 6, 2009, pp. 1-12.

[9] Bussani A., Griffin J. L., Jasen B., Julisch K., Karjoth G., Maruyama H., Nakamura M., et al., Trusted Virtual Domains: Secure Foundations for Business and IT Services, Research Report RC23792, IBM Corporation, November 2005.

[10] IEEE Standard 802.1Q for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks. IEEE 2005 see: http://standards.ieee.org/getieee802/download/802.1Q-2005.pdf

[11] Chaves, S.A.; Uriarte, R.B.; Westphall, C.B. Toward an Architecture for Monitroing Private Clouds, IEEE Communications Magazine Vol. 49, Issue 12, 2011, pp. 130-137.

[12] Gu Y., McCallum A. and Towsley, D. Detecting anomalies in network traffic using maximum entropy. Tech. rep., Department of Computer Science, UMASS, Amherst, 2005, pp. 345-350.

[13] Aryan Taheri Monfared, Martin Gilje Jaatun, Monitoring Intrusions and Security Breaches in Highly Distributed Cloud Environments, cloudcom, 2011 IEEE Third International Conference on Cloud Computing Technology and Science, 2011, pp. 772-777

[14] Masayuki Okuhara et al, Security Architecture for Cloud Computing, FUJITSU Sci. Tech. J., Vol. 46, No. 4, October 2010, pp. 397-402.

[15] PietraS.D., Pietra V.D. and Lafferty J. Inducing features of random fields. IEEE Transactions on Pattern Analysis andMachine Intelligence, Vol. 19, No. 4, 1997, pp. 380-393.

[16] Dao, Vu and V. Rao Vemuri, Computer Network Intrusion Detection: A Comparison of Neural Networks Methods. Differential Equations and Dynamical Systems (Special Issue on Neural Networks), Vol. 10, No. 1/2, 2002, pp. 201-214.

[17] libLBFGS: a library of Limited-memory Broyden Fletcher Goldfarb Shanno (L-BFGS). http://www.chokkan.org/software/liblbfgs/, copyright Naoaki Okazaki, 2002-2010.

[18] Stephanie Forrest, Alan S. Perelson, L. Allen, and R. Cherukuri. Self -nonself discrimination in a computer. In Proceedings of the 1994 IEEESymposium on Research in Security and Privacy. 1994, pp. 360-365

[19] MongoDB - a scalable, high-performance, open source NoSQL database. http://www.mongodb.org, 2012.