Application-Specific Traffic Anomaly Detection Using Universal Background Model

Hassan Alizadeh
IEETA, Univ. of Aveiro
3810-193 Aveiro, Portugal
hassan.alizadeh@ua.pt

Samaneh Khoshrou
INESC-TEC, Univ. of Porto
4200 - 465 Porto, Portugal
skhoshrou@inescporto.pt

André Zúquete
DETI/IEETA, Univ. of Aveiro
3810-193 Aveiro, Portugal
andre.zuquete@ua.pt

ABSTRACT

This paper presents an application-specific intrusion detection framework in order to address the problem of detecting intrusions in individual applications when their traffic exhibits anomalies. The system is based on the assumption that authorized traffic analyzers have access to a trustworthy binding between network traffic and the source application responsible for it. Given traffic flows generated by individual genuine application, we exploit the GMM-UBM (Gaussian Mixture Model - Universal Background Model) method to build models for genuine applications, and thereby form our detection system. The system was evaluated on a public dataset collected from a real network. Favorable results indicate the success of the framework.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection (e.g., firewalls)*; C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network monitoring*; I.5.1 [**Pattern Recognition**]: Models—*Statistical*

General Terms

Algorithms, Measurement, Security

Keywords

Network Anomaly, Intrusion Detection, Universal Background Model, Gaussian Mixture Models, Traffic Flows, Malware, Web Applications

1. INTRODUCTION

The world is increasingly getting more dependent on the Internet, which is being used by millions of persons that use vulnerable machines. Those machines are natural targets of attackers, which may compromise them in order to perform illegal activities, such as information theft or machine exploit in botnets. Intrusion detection systems (IDS) aim at mitigating this problem.

Although there is a long record of work in intrusion detection in the last three decades, the research field is far from exhausted. Intrusion detection is quite often a very complex task, since the configuration of IDS and their intrusion reporting is not straightforward.

For an intrusion to have any benefit, it must produce some activity, namely network activity. Thus, a machine compromised by an intrusion will most probably produce network traffic related with the intrusion, i.e., traffic that would not happen if the intrusion had not happen. Therefore, a straightforward way to detect evidences of intrusions is with traffic inspection.

Detection of intrusions from traffic inspection can be performed by matching observed traffic with large databases of well-known intrusion signatures created by experts knowing the traffic related with known intrusions (signature-based IDS), or by noticing deviations from pre-defined descriptions (profiles) of normal traffic (anomaly-based IDS). Unlike signature-based, anomaly-based IDS may detect new types of intrusions (unknown) and tackle zero-days attacks, although there is a difficulty in their adaptation to constantly changing profiles. Due to the advantages of anomaly-based relatively to signature-based paradigms regarding zero-day attacks, most existing research studies have focused on anomaly-base IDS.

1.1 Problem Definition and Motivations

Most proposed anomaly-base IDS's ordinarily make use of aggregated network traffic patterns for building profiles. Their main drawback is that, given the enormous number of different traffic types of interest, building a well-defined profile encompassing all normal traffic is extremely difficult. Consequently, they suffer from high false alarm rates.

The general trend observed in the literature for reducing such rates is that "*anomalies must be identified within each separate traffic class of interest*" [15, 29, 32]. The main problem of such systems is that the production and exploitation of traffic profiles for each class of interests are performed without knowing and considering which applications/services are responsible for the traffic. Consequently, they are not able to detect deviations caused by applications generating abnormal traffic whenever such traffic fits into the network normality profiles.

For example, Thunderbird uses mail related protocols (SMTP, POP, IMAP, etc.). But under the influence of a sophisticated intrusion, it may generate some other kinds of applicationlayer protocols, such as well-formed HTTP. Although such intrusion causes Thunderbird to deviate from its normal behavior, this deviation may be within the normal behavior of the network, since HTTP is a quite common protocol. As a consequence, this application anomaly cannot be detected by anomaly-based IDS equipped solely with network normality profiles, which leads to an increased false negative alarm rate. To detect intrusions in such scenarios, where a malware exploits an application yet generating well-formed traffic, the process of anomaly detection should be separately applied on each application's traffic. Hence, IDS need to be equipped with accurate applications' traffic profiles, as well as tools to bind applications to observed traffic. To the best of our knowledge, this problem has not been addressed in the literature.

Profiling individual application is not a new concept, since researchers in the field of Traffic Identification and Classification (TIC) make use of them as a baseline of their systems to identify and categorize traffic flows by application type. But it has received less attention in the scope of IDS research. Much of the history of TIC has focused on design systems to answer the following question: *what kind of traffic this packet (flow) belongs to?* Herein, we want to answer a more fine-grained question: *is this packet/flow normal for its source application?* To answer this question, the system requires to: (i) identify the claimant (source application) for being responsible for the traffic and (ii) model the genuine traffic of each application present in the network [3].

The methods presented in [37, 38, 30] provide a trustworthy binding between network packets and source applications that can be applied to fulfill the first requirement. Exploiting such binding, it is possible to check if a packet/flow conforms to a given network profile, besides being able to check if an application conforms to a given host profile (i.e., if it is normal or acceptable to have the application running in the host).

In our scenario we need to design a framework that decides whether or not traffic generated by an application conforms to its (genuine) application-specific profile. Such goal reminds a typical authentication verification algorithm in the field of biometric (using personal templates), where individual models need to be trained for all the classes. Modeling multiple classes has been addressed in the classic traffic classification methods. Since our scenario is a verification process, such methods are not directly applicable in our scenario. To the best of our knowledge, this is the first attempt to validate the legitimacy of the traffic (in the sense of occurrence likeliness) produced by an application.

Various methods have been applied for training computational models in the literature. GMM-UBM has become a standard in voice biometrics literature due to its widely use. However, it is a less explored method in the field of network security. Given flow features generated by each applications, we exploit GMM-UBM in order to model individual application, and thereby form our detection system.

In this paper we propose an application verification framework in order to improve the safety of the network. The system is designed to detect abnormality in individual applications rather than in the whole network. When the application claims the responsibility for aggregated traffic, the system determines whether the claim is true or false (the aggregated traffic conforms to the genuine application profile). Once the claim is false, it is most likely that the application is working under the influence of an intrusion.

1.2 Contributions

The main contributions of this paper are the following:

- Propose an application-specific traffic verification framework as a new research trend in network anomaly-based intrusion detection systems.
- Propose the use of GMM-UBM as an effective learning method in the literature in order to build a profile for genuine traffic of individual applications.

1.3 Paper Structure

The rest of the paper is organized as follows. Section 2 presents a literature review. GMM-UBM is briefly introduced in section 3. Section 4 presents the details of experimental setups and evaluation results. Section 5 concludes the paper with suggestions of possible future work.

2. LITERATURE REVIEW

Network anomalies are inferred from patterns in network traffic data that deviate from (or do not conform to) the normal network behavior (profiles) [5]. The process of finding such suspicious patterns is referred as network anomaly detection. Intrusions are the main cause of network anomalies and a framework that is able to address this problem is referred Anomaly-based Network Intrusion Detection System (ANIDS). An overview of various ANIDS was presented in [7].

The main assumption behind most existing ANIDS approaches is that the statistical characteristics extracted from traffic sources differ between normal and intrusion-infected traffic. Hence, they exploit statistical characteristics of normal network traffic in order to build their profiles. The statistical characteristics can be extracted from one or more different information sources of traffic such as: packet header, payload, flow, bandwidth usage or packet distribution.

Payload-based approaches [33, 34, 18, 12, 16, 36, 31] are probably the most promising way of detecting traffic anomalies created by malware-infected applications installed upon intrusions. In fact, most intrusions meet their own illicit purposes through the propagation of infected payloads. These methods have the ability to provide a near real-time detection with high accuracy levels. However, they suffer from high computational cost and problems when dealing with encrypted or otherwise securely encapsulated traffic. Moreover, privacy assurance is one of the main challenges when using these methods.

In order to alleviate these problems, alternative approaches relying on statistical characteristics of flows have been proposed. A flow is the fundamental object of the TIC research field. It has also received some attention in IDS literature recently [6, 9, 10]; in [28], the authors provided an overview of flow-based IDS.

According to the TCP/IP model, network traffic can be grouped into one or more data flows. Each flow is associated with an individual application running on the particular host and defined as a (unidirectional or bidirectional) sequence of IP packets sharing typically a 5-tuple identifier (source IP, destination IP, source port, destination port, IP protocol type) within a certain period of time. A flow can be described by a number of statistical properties parameterizing its behavior. Such features are extracted from the packet information regardless of their payloads. This can be computed directly from some parts of packet headers (such as average packet size, total transferred bytes) as well as indirectly from the time when a packet arrives (i.e. inter-packet timings such as min/max/average/variance packet inter-arrival time, flow duration). In [17], the authors described comprehensively 248 flow features.

Flow-based and payload-based ANIDS's may be able to detect some intrusions in applications in their current form. However, they suffer from detecting intrusion-infected applications generating well-formed traffic of other known applications. To the best of our knowledge, such possible intrusions, and consequently the systems for detecting them, have not yet been addressed in the literature. To achieve this goal, an ANIDS needs to first identify applications responsible for the traffic and then be equipped with a model for each individual, genuine application. In [37, 38], we have developed a framework for tagging the traffic with the exact source application. In this paper we make use of this assumption in order to build per-application models using statistical characteristic of traffic flows, and thereby form our detection system.

Machine Learning (ML) techniques are highly efficient in dealing with statistical data. Moreover, most of them are available as off-the-shelf components. For these reasons, they have been widely applied in ANIDS. In [27], the authors highlighted some challenges of applying ML to ANIDS and provided some guidelines to overcome them.

ML-based anomaly detection algorithms are typically grouped into *discriminative* and *generative* approaches. A generative approach (e.g. [4]) builds a model solely based on training examples of the normal class, whereas a discriminative approach (e.g. [14]) attempts to learn the distinction between the normal and abnormal classes [11]. Thus, discriminative approaches require a big set of abnormal data at the training phase, which makes them less practical. On the other hand, once a test record is received, generative algorithms are able to provide a robust notion of normality by checking how well it fits to the normal behavior model. This characteristic, as well as good ability of generalization, makes them the proper choice for anomaly detection scenarios. GMM-UBM is one of the most used generative approaches in voice biometrics. However, it is a less explored area within ANIDS.

3. UNIVERSAL BACKGROUND MODEL

The universal background model (UBM) [23] is an effective framework that achieved a great success in the field of voice biometrics [19]. Conceptually, it is a large pool of data that covers all space to represent person independent feature characteristics. To make an accepting or rejecting decision it is compared against person specific features.

A block diagram of the proposed system, Inspired by voice biometric systems, is shown in the figure 1.

Given a segment of traffic, T, and a claimed application, A, we define the problem of abnormality detection in traffic of specific application as the determination of whether or not T was generated by A. It can be easily understood if this problem is restated as a basic hypothesis test between

H_0: T <u>was</u> generated by hypothesized (A)

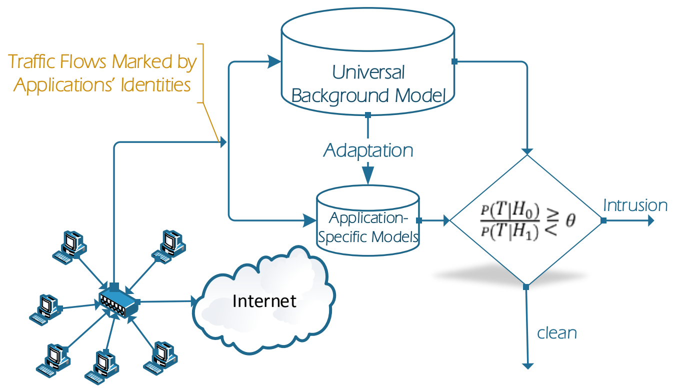
Traffic Flows Marked by Applications' Identities
Universal Background Model
Adaptation
Application-Specific Models
$\frac{P(T|H_0)}{P(T|H_1)} \gtrless \theta$
Intrusion
Internet
clean

Figure 1: The block diagram of the proposed system.

and
H_1: T <u>wasn't</u> generated by hypothesized application (A)

The optimal decision test between these two hypotheses is taken by a likelihood-ratio test:

$$\varphi(T|H_0) = \frac{p(T|H_0)}{p(T|H_1)} \begin{cases} \geq \theta & \text{accept } H_0 \\ < \theta & \text{reject } H_0 \end{cases} \qquad (1)$$

where $p(T|H_i), i \in \{0,1\}$ is the *likelihood* of observing sample T under hypothesis i and θ is the decision threshold for accepting or rejecting the claim. H_0 should characterize the hypothesized specific application, whereas, H_1 should be able to model *all the alternatives to the hypothesized specific application*. Hence, it is inevitable to have and use a model that successfully covers the space of alternatives to the hypothesized application. Such model must be trained with a large pool of data, covering a representative user space as well as significant amount of sources of variability. The most common solution in the literature that fulfill these requirements is the *universal background model* (UBM) [22].

Gaussian Mixture Models (GMM) are typically chosen to model both the UBM, i.e. H_1, and the hypothesized application, i.e. H_0. Such models are capable of capturing the empirical Probability Density Function (PDF) of a given set of feature vectors, so as to faithfully model their intrinsic statistical properties [23]. Gaussian mixtures display both the robustness of parametric unimodal Gaussian density estimates, as well as the ability of non-parametric models to fit non-Gaussian data [21]. This duality, alongside the fact that GMM have the noteworthy strength of generating smooth parametric densities, confers to such models a strong advantage as generative models [13].

3.1 H_1: UBM Parameter Estimation

To train the Universal Background Model a large amount of data, i.e. a set composed of data from all the enrolled applications, is used, so as to cover a wide range of possibilities in the individual search space [26]. The training process of the UBM is simply performed by fitting a k-mixture GMM to the set of feature vectors extracted from all the data. The model parameters can be obtained using the Maximum Likelihood (ML) estimation.

If we interpret the UBM as an "impostor" model, its "genuine" counterpart can be obtained by adaptation of the UBM's parameters using individual specific data. For each enrolled individual (application), an application specific model (APSM) is therefore obtained.

3.2 H_0: Application Specific Model (APSM)

Construction of hypothesized application model can be obtained in two ways: As a classic solution, a GMM can be trained using specific individual application data. However, Maximum Likelihood GMM (ML-GMM) training for individual class requires an adequate amount of data for each individual class which is unfavorable in real-world scenarios [20]. As a practical alternative, the model can be generated by the tuning of the UBM parameters in a maximum a posteriori (MAP) sense, using individual application data. This strategy provides clear advantages over classic ML-GMM. Derivation of APSM parameters from a well-trained ubm via adaptation provides a tight coupling between APSM and UBM, resulting in better performance and faster scoring than uncoupled methods [35], as well as a robust and precise parameter estimation, even when only a small amount of data is available [26]. Furthermore, the UBM constitutes a robust initialization for the application-specific models. Since the APSM is trained only using specific individual data, it is more prone to a poor convergence than the GMM for the UBM, learnt from a big pool of data [13].

3.3 Decision Making

Once the models of both UBM and individual applications are trained, making decision on new observations is quite straightforward. As referred in previous sections, the normality check is performed through the projection of the new test data, X_{test}, onto both the UBM and the claimed application model. The normality score is obtained as the likelihood-ratio.
We stress that, the ratio between the application-specific and the UBM probabilities of the observed data is a more robust decision criterion than relying solely on the application specific GMM probability. The use of a likelihood ratio with a universal reference works as a normalization step, mapping the likelihood values according to their global projection. Without such step, finding an optimal value for the decision threshold, θ, presented in Equation 1, would be a far more complex process.

4. EXPERIMENTAL METHODOLOGY

We first describe in detail the dataset used in the experiments. We then describe the experimental setup and evaluation metrics. Finally, we present the experimental results.

4.1 Dataset

All experiments are conducted on *"measurement"* dataset [1]. The dataset consists of three files: a *PCAP* file, a *flowlog*, a *readme* file. The *PCAP* file contains 6 Gigabytes data volumes from 12 million packets generated by various type of applications (23 applications) operated in 6 hosts within a network during 43 hours. The *flowlog* file contains all records of TCP and UDP flows, their starting and ending times as well as identity of applications (i.e. the first two characters of their corresponding executable file names as identifiers) responsible for the flows. The *readme* file lists identities used for each applications.

Before capturing network data, a specific network driver proposed in [30] was installed into all hosts in order to mark each particular flow with identity of responsible application. The identity is placed in the first packet of the flow by extending the IP option in the Router Alert option field.

Our experiments were conducted with TCP data flows. Table 1 shows the number of TCP flow records for each applications operating in different hosts. Due to scarcity of such data for some applications, only 9 applications were considered in the experiments. Note that each individual application can be run by one or more host(s) and also each host can execute one or more application(s). We stress that we do not use information of IP addresses and Ports in order to build per-application profiles.

Table 1: Structure of dataset used in this study [1]: Number of TCP flows generated by 23 applications operated in 6 hosts. The sign "-" is used if an application is not being executed on a particular host.

Applications	IP Add. of Hosts running Apps						Total No. of TCP Flows
	172.16.2.2	172.16.2.3	172.16.2.11	172.16.2.12	172.16.2.13	172.16.2.14	
wpc55agv2:Linksys Monitor	-	-	-	3504	3432	-	6936
eMule	-	-	2	-	3544	-	3546
Azureus	-	-	-	1834	-	-	1834
Internet Explorer	563	-	229	213	145	213	1363
MS Outlook Express	-	-	-	1077	-	-	1077
FilePlanet download manager	535	-	-	-	-	-	535
Skype	15	22	70	-	207	-	314
MSN Messenger	82	-	-	-	110	-	192
Limewire	-	-	-	-	-	131	131
Windows Media Player	-	-	-	-	32	25	57
Winamp	-	-	45	1	-	-	46
?? (unidentified)	4	-	-	2	-	30	36
StrongDC	-	-	-	33	-	-	33
Google Earth	15	-	1	4	-	-	20
uTorrent	10	-	-	-	-	-	10
TA Spring	3	-	-	-	-	-	3
Silkroad Online	3	-	-	-	-	-	3
Putty	1	-	-	-	-	-	1
Total Commander	-	-	-	-	-	1	1
WinSCP	-	-	-	1	-	-	1
Lsass	-	-	-	-	-	-	0
Svchost	-	-	-	-	-	-	0
Tremulous	-	-	-	-	-	-	0

Feature extraction: Applying Netmate tool [2], the PCAP file was processed in order to group the network traffic into separated TCP flows as well as to calculate their static values as flow features. A timeout mechanism (60 sec) was used to determine the end of a flow when a termination is not observed. Given a PCAP file, the Netmate tool in its current form is able to compute 42 features. Some of these features are as follows: the total number of packets/bytes, the minimum/ mean/ maximum/ variance of packet length (in bytes) as well as of inter-arrival time (in microseconds) in each direction, flow duration, etc.

Feature Selection: Feature selection can be formulated as finding the best possible subset of the feature space that possibly yields the highest performance with the lowest computational cost. The Plus ‘L’ and Take away ‘R’ approach, we adopted for selecting the best features is similar to [24]. The selection method keeps adding 2 and taking away one feature until the best performance is achieved. The selected feature set is then used in the test stage.

4.2 Experimental Setup and Evaluation Metrics

Our experiments consisted of three main stages: training, evaluation and test. Hence, the dataset was randomly divided in three subsets: 20% used for training the models, 40% for evaluation on which the system can be adjusted and calibrated, and the remaining part (40%) was used to assess the system performance.

The universal background model (UBM) was built using a pool of unlabeled data. Application specific models were obtained by adapting GMM, built from individual application’s data using UBM parameters. The likelihood test outputted the normality score. Scores in evaluation step are employed to find an optimal value for the threshold (θ).

The evaluation set was selected to produce the genuine application (true) and infected application (false) scores, which are used to find a threshold that determines if a specific application is clean or not. The threshold (θ) can be defined in two ways: 1) global thresholding, where a single threshold is defined; 2) application-specific thresholding, employing a different threshold for each class (application). In this study, the threshold was determined based on the Equal Error rate criterion, i.e. by the operating point where the False Rejection Rate (FRR) is equal to False Acceptance Rate (FAR). False acceptance happens when an infected application is not detected as such. False rejection happens when a clean application is considered infected. Given the assumptions of our framework, where infected applications generate well-formed traffic of other applications present in the network, we simulated the infected traffic of a specific application using the normal traffic of others claiming its identity. Half Total Error Rate (HTER) is a standard metric in the field of biometrics that combines both FAR and FRR into a single measurement. The HTER is formulated as:

$$HTER(\theta) = \frac{FAR(\theta) + FRR(\theta)}{2} \tag{2}$$

The evaluation set was also used to find the optimal set of features using floating search methods.

Finally the test set was selected to simulate realistic anomaly detection tests.

4.3 Experimental Results

The choice of the number of components is a fundamental aspect in the design of our GMM-UBM based system. Since UBM is trained on a large pool of data, one expects that a higher number of components could lead to a more precise model. However, when the number of components rises tight, we may have no access to enough training data for some of the components. Besides, increasing the number of components may lead to over fitting GMM to the training set and low system accuracy for the independent test dataset. Figure 2 shows the HTER in evaluation and test sets using a different number of GMM mixtures. We observe that setting where 16 components are trained yields significantly lower errors. In the next step, the adopted search method (Plus ‘L’ and Take away ‘R’ algorithm) was used

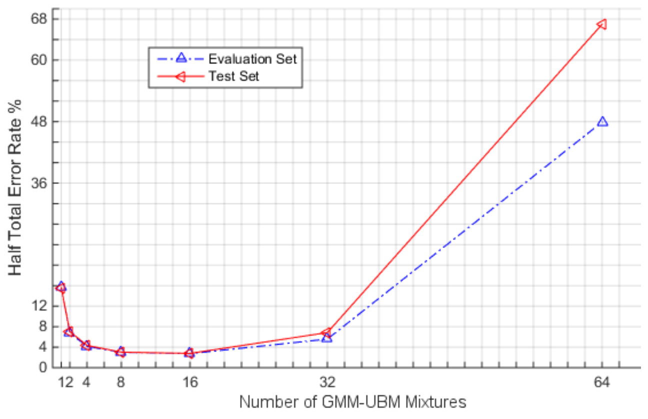

Figure 2: HTER for both evaluation and test sets in different GMM mixture sizes.

for selecting an optimal subset of features giving the minimum HTRE in the evaluation step. In the search algorithm we used L=2 and R=1. The optimal subset including 11 features was selected in evaluation step.

Then, the feature space was transformed onto an orthogonal space using Local Principle Component Analysis (LPCA). LPCA [8] improved the accuracy of UBM-based methods in the field of speaker recognition, where UBM space is partitioned into k disjoint regions (k is the number of classes). PCA algorithm is performed on the set of vectors for each class to obtain the transformation matrix. The transformation matrix of each region is stored in order to use in the evaluation and test phase of the same region. An application model is constructed via MAP adaptation with the set of transformed featured. We used the MSR toolbox [25] to train both UBM and application models.

Table 2: Results obtained by global and application-specific threshold techniques. FAR: False Acceptance Rate. FRR: False Rejection Rate. HTER: Half Total Error Rate.

	Global Threshold			Application-Specific Threshold		
	FAR	FRR	HTER	FAR	FRR	HTER
Evaluation Set	2.73	2.74	2.73	3.16	1.73	2.44
Test Set	2.75	2.83	2.79	3.18	1.73	2.45

In the next step, scores outputted by GMM-UBM were properly normalized. The normalized scores were then considered to define the proper threshold. Table 2 provides a summary of the overall FAR, FRR, and HTER measures using global and class-specific thresholds. The results were reported with 3-fold cross-validation. Results indicate that using class-specific thresholds is preferable. Hence, the results of this method are detailed in Table 3, where the performance of this method for multiple applications is presented.

5. CONCLUSION & FUTURE WORK

In this paper we presented an application-specific intrusion detection framework inspired by biometric verification methods. The framework exploits GMM-UBM in order to build robust and precise models for genuine traffic records.

Table 3: Results obtained by the application-specific threshold technique for different applications.

Applications	Evaluation Set			Test set		
	FAR	FRR	HTER	FAR	FRR	HTER
wpc55agv2	0.79	0.78	0.78	0.85	1.00	0.92
eMule	2.49	2.5	2.49	2.64	1.78	2.21
Azureus	2.73	2.73	2.73	2.52	2.87	2.69
Internet Explorer	2.94	2.94	2.94	2.95	3.18	3.06
MS Outlook Exp	0.16	0.16	0.16	0.21	0.85	0.53
FilePlanet download	2.34	2.35	2.35	2.37	2.03	2.2
Skype	4.06	4.06	4.06	4.48	3.51	3.99
MSN Messenger	4.95	5.26	5.11	4.85	9.65	7.25
Limewire	6.64	6.38	6.51	6.37	9.1	7.74

The system detects abnormality in traffic generated by specific applications in a multi-application network. However, this is a supervised setting where the genuine classes (applications) need to be properly identified in advance. Extending our framework in order to be applicable in a non-stationary environment, where both class evolution and concept drift are available, is the main direction for our future work.

6. ACKNOWLEDGMENTS

This work was funded by FCT (Foundation for Science and Technology), in the context of the project PEst-OE/ EEI/ UI0127/ 2014, and the PhD scholarship SFRH/ BD/ 84037/ 2012.

7. REFERENCES

[1] *The measurement used in this article*, 2008. Available at `http://www.crysys.hu/~szabog/index_files/measurement.tar`.

[2] Netmate, 2009. Available at `http://sourceforge.net/projects/netmate-meter/files/netmate-meter/`.

[3] H. Alizadeh and A. Zúquete. Traffic classification for managing applications' networking profiles. *submitted to Security and Communication Networks*, 2014.

[4] M. Bahrololum and M. Khaleghi. Anomaly intrusion detection system using hierarchical gaussian mixture model. *International journal of computer science and network security*, 8(8):264–271, 2008.

[5] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[6] J. Copeland. Flow-based detection of network intrusions, Feb. 27 2007. US Patent 7,185,368.

[7] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1):18–28, 2009.

[8] C. Hanilci and F. Ertas. VQ-UBM based speaker verification through dimension reduction using local pca. In *19th European Signal Processing Conference (EUSIPCO), Barcelona, Spain*, pages 1303–1306, 2011.

[9] L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras. Sshcure: A flow-based ssh intrusion detection system. In *Dependable Networks and Services*, volume 7279 of *Lecture Notes in Computer Science*, pages 86–97. Springer Berlin Heidelberg, 2012.

[10] R. Hofstede and A. Pras. Real-time and resilient intrusion detection: A flow-based approach. In *Dependable Networks and Services*, volume 7279 of *Lecture Notes in Computer Science*, chapter 13, pages 109–112. Springer Berlin Heidelberg, 2012.

[11] W. Hu, Y. Liao, and V. R. Vemuri. Robust anomaly detection using support vector machines. In *Proceedings of the international conference on machine learning*, pages 282–289, 2003.

[12] A. Jamdagni, Z. Tan, X. He, P. Nanda, and R. P. Liu. Repids: A multi tier real-time payload-based intrusion detection system. *Computer Networks*, 57(3):811–824, 2013.

[13] S. Khoshrou, J. S. Cardoso, and L. F. Teixeira. Learning from evolving video streams in a multi-camera scenario. *submitted to Machine Learning Journal*, 2014.

[14] D. Kim and J. Park. Network-based intrusion detection with support vector machines. In *Information Networking*, volume 2662 of *Lecture Notes in Computer Science*, chapter 73, pages 747–756. Springer Berlin Heidelberg, 2003.

[15] C. Krügel, T. Toth, and E. Kirda. Service specific anomaly detection for network intrusion detection. In *Proceedings of the ACM Symposium on Applied Computing*, SAC '02, pages 201–208, New York, NY, USA, 2002. ACM.

[16] Z. Like and G. B. White. Anomaly detection for application level network attacks using payload keywords. In *IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA'07)*, pages 178–185, April 2007.

[17] A. Moore, D. Zuev, M. Crogan, and Q. Mary. *Discriminators for use in flow-based classification.* Queen Mary and Westfield College, Department of Computer Science, 2005.

[18] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee. Mcpad: A multiple classifier system for accurate payload-based anomaly detection. *Computer Networks*, 53(6):864–881, 2009.

[19] D. Povey, S. M. Chu, and B. Varadarajan. Universal background model based speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'08)*, pages 4561–4564, March 2008.

[20] K. S. Rao and S. Sarkar. *Robust Speaker Recognition in Noisy Environments.* Springer, 2014.

[21] D. Reynolds. Gaussian mixture models. In *Encyclopedia of Biometrics*, book section 196, pages 659–663. Springer US, 2009.

[22] D. A. Reynolds. An overview of automatic speaker recognition technology. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'02)*, volume 4, pages 4072–4075, May 2002.

[23] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10(1- 3):19–41, 2000.

[24] M. Sadeghi, S. Khoshrou, and J. Kittler. Colour feature selection for face authentication. In *MVA'07*, pages 49–52, 2007.

[25] S. O. Sadjadi, M. Slaney, and L. Heck. MSR identity toolbox v1.0: A MATLAB toolbox for speaker-recognition research. *Speech and Language Processing Technical Committee Newsletter*, November 2013.

[26] K. Shinoda and N. Inoue. Reusing speech techniques for video semantic indexing [applications corner]. *Signal Processing Magazine, IEEE*, 30(2):118–122, March 2013.

[27] R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *IEEE Symposium on Security and Privacy (SP)*, pages 305–316, May 2010.

[28] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller. An overview of ip flow-based intrusion detection. *Communications Surveys & Tutorials, IEEE*, 12(3):343–356, Third 2010.

[29] J. W. Stokes, J. C. Platt, J. Kravis, and M. Shilman. Aladin: Active learning of anomalies to detect intrusions. Technical Report MSR-TR-2008-24, Microsoft Research, March 2008.

[30] G. Szabó, D. Orincsay, S. Malomsoky, and I. Szabó. On the validation of traffic classification algorithms. In *Passive and Active Network Measurement*, volume 4979 of *Lecture Notes in Computer Science*, chapter 8, pages 72–81. Springer Berlin Heidelberg, 2008.

[31] S. A. Thorat, A. K. Khandelwal, B. Bruhadeshwar, and K. Kishore. Payload content based network anomaly detection. In *First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT'08)*, pages 127–132, Aug 2008.

[32] A. Valdes. Detecting novel scans through pattern anomaly detection. In *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, volume 1, pages 140–151 vol.1, April 2003.

[33] K. Wang, G. Cretu, and S. Stolfo. Anomalous payload-based worm detection and signature generation. In *Recent Advances in Intrusion Detection*, volume 3858 of *Lecture Notes in Computer Science*, chapter 12, pages 227–246. Springer Berlin Heidelberg, 2006.

[34] K. Wang and S. Stolfo. Anomalous payload-based network intrusion detection. In *Recent Advances in Intrusion Detection*, volume 3224 of *Lecture Notes in Computer Science*, chapter 11, pages 203–222. Springer Berlin Heidelberg, 2004.

[35] Z. Xiong, T. F. Zheng, Z. Song, F. Soong, and W. Wu. A tree-based kernel selection approach to efficient gaussian mixture model - universal background model based speaker identification. *Speech Communication*, 48(10):1273–1282, 2006.

[36] L. Zhang and G. B. White. Analysis of payload based application level network anomaly detection. In *40th Annual Hawaii International Conference on System Sciences*, pages 99–99, Jan 2007.

[37] A. Zúquete, P. Correia, and H. Shamalizadeh. Packet tagging system for enhanced traffic profiling. In *IEEE 5th International Conference on Internet Multimedia Systems Architecture and Application (IMSAA)*, pages 1–6. IEEE, Dec 2011.

[38] A. Zúquete and M. Rocha. Identification of source applications for enhanced traffic analysis and anomaly detection. In *IEEE International Conference on Communications (ICC)*, pages 6694–6698. IEEE, June 2012.