Application of Evolutionary Algorithms in Detection of SIP based Flooding Attacks

M. Ali Akbar, Muddassar Farooq
Next Generation Intelligent Networks Research Center (nexGIN RC)
National University of Computer & Emerging Sciences (FAST-NUCES)
Islamabad, Pakistan
{ali.akbar, muddassar.farooq}@nexginrc.org

ABSTRACT

The Session Initiation Protocol (SIP) is the de facto standard for user's session control in the next generation Voice over Internet Protocol (VoIP) networks based on the IP Multimedia Subsystem (IMS) framework. In this paper, we first analyze the role of SIP based floods in the Denial of Service (DoS) attacks on the IMS. Afterwards, we present an online intrusion detection framework for detection of such attacks. We analyze the role of different evolutionary and non-evolutionary classifiers on the classification accuracy of the proposed framework. We have evaluated the performance of our intrusion detection framework on a traffic in which SIP floods of varying intensities are injected. The results of our study show that the evolutionary classifiers like sUpervised Classifier System (UCS) and Genetic clASSIfier sySTem (GAssist) can even detect low intensity SIP floods in realtime. Finally, we formulate a set of specific guidelines that can help VoIP service providers in customizing our intrusion detection framework by selecting an appropriate classifier–depending on their requirements in different service scenarios.

Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**]: General—*Security and protection*

General Terms

Experimentation, Security

Keywords

Denial of Service, Network Security, Session Initiation Protocol, IP Multimedia Subsystem

1. INTRODUCTION

The global communication market is rapidly moving towards the integration of cellular voice networks and the Internet. The driving logic behind this phenomenon is the easy availability of high-bandwidth Internet connectivity at much cheaper rates as compared to the cellular network. The demand for novel Internet-based multimedia applications is on the rise. To provide the cellular users with transparent access to these IP based services, the operators in the telecommunication sector are gradually adopting an all IP standard for multimedia applications known as *IP Multimedia Subsystem* (IMS) [15]. IMS provides an architectural framework for service control using some common Internet-based protocols. The main function of IMS is the control and signaling of multimedia sessions. The actual transport of the multimedia traffic is done independent of IMS.

To handle negotiation and control of user's sessions, IMS uses the Session Initiation Protocol (SIP). The core of the IMS framework consists of three different types of SIP based components. These SIP servers/proxies are known as the Call Service Control Functions (CSCFs) [6]. The three CSCFs are: 1)Proxy CSCF acts as a gateway to the IMS Core and simply relays incoming SIP requests to the I-CSCF; 2)Interrogating CSCF hides the internal topology of the IMS core, and 3)Serving CSCF handles the incoming SIP requests and performs signaling operation like a common SIP server.

SIP protocol–apart from IMS–is also the de facto standard for session signaling in Voice over Internet Protocol (VoIP) infrastructures. A reliable VoIP/IMS infrastructure must guarantee at least 99.9% uptime to stay competitive in the telecommunication market. Therefore, a smooth operation of SIP servers (CSCFs in IMS) is vital for any service operator. However, the popularity of the VoIP and IMS based services is now receiving an ever increasing attention of imposters or intruders towards these infrastructures. VoIP servers are among the **SANS Top 20 Security Risks** [20]. A successful Denial of Service (DoS) attack on a VoIP server can result in huge financial losses and seriously undermines the credibility of an operator. Moreover, it also adds to frustration of end users.

1.1 Problem Statement

The SIP protocol is vulnerable to the DoS attacks. The SIP based DoS attacks can be launched in two ways: (1) the attacker sends a huge burst of SIP messages that simply overloads the capacity of a SIP server subsequently forcing it to deny service to the legitimate requests, and (2) the attacker sends one or more SIP requests containing malformed SIP header fields. These malformed requests exploit vulnerabilities of the SIP servers. The authors in [12] have shown that the response time of an open-source SIP server increases significantly even for low intensity attacks. In this study, to maintain focus we detect SIP flood attacks only.

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.

1.2 Related Work

In 2003, a comprehensive security framework for IMS was released by the 3rd Generation Partnership Project 2 (3GPP2) [1]. Many of the known vulnerabilities in the SIP protocol are addressed by this framework. However, the framework specifies no standard security mechanism for the detection and prevention of flooding attacks. To fill this void, researchers in the recent past have proposed several defensive measures. Some security frameworks (such as [21], [4] and [17]) have been proposed for the detection of flooding attacks in IMS. Anomaly detection algorithms (see [2] and references therein) have been actively used for security in VoIP networks. Recently, the authors in [12] proposed a support vector machines based intrusion detection system for detection of SIP flooding attacks and spam threat.

The evolutionary algorithms have been applied to solve some real world problems in the domain of network security. However, only little work is done on the application of evolutionary algorithms in detecting DoS attacks on IMS. The authors of [4] have proposed an artificial immune system based algorithm for detection of flood attacks on IMS and compared it with a signature based algorithm.

1.3 Our Contributions

To the best of our knowledge, this is the first comparative study of evolutionary and non-evolutionary classifiers to solve the real-world problem of SIP flood attacks in IMS. We propose a generic intrusion detection system with flexible set of features and classifiers. In this study, we have used a subset of SIP features presented in [12]. We have selected four non-evolutionary and six evolutionary algorithms from various machine learning schemes. These classifiers are listed in Table 1. We evaluate these algorithms on ten synthetic SIP traffic datasets with different levels of attack intensities and durations. We formulate a number of guidelines on the basis of the results of our experiments that can help a service operator to choose the best classifier or set of classifiers for various application scenarios.

2. INTRUSION DETECTION FRAMEWORK

We present a generic intrusion detection framework for attack detection in IMS. The framework consists of six components that will be shortly introduced. This framework is the generic form of the intrusion detection system proposed by the authors of [12]. Our framework is modular in design that allows the operator to customize it by selecting appropriate set of features and classifier.

2.1 Components

Figure 1 shows the main components of the framework. These components are described below:

The packet sniffer captures the online SIP traffic and stores it in a packet buffer. The captured packets are stored in a fixed size buffer. The size of the buffer is chosen by making a trade-off between the processing delay and the significance of the traffic patterns in the collected data. The size of the buffer should be small because the buffered packets only gets service once the buffer is full. However, the number of packets stored should be large enough to depict the actual protocol traffic pattern.

As the buffer gets filled, the feature extractor component analyzes the packets and computes values of different features. Then the feature values are passed to the classifier.

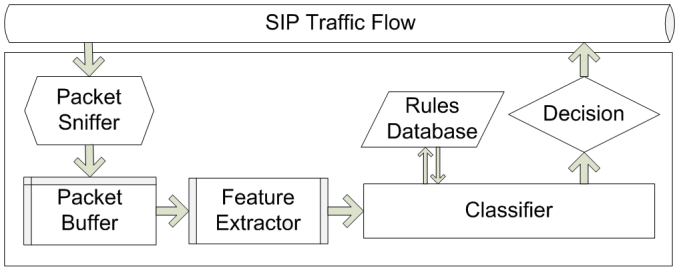

Figure 1: Intrusion Detection Framework

The idea of calculation of feature values on fixed number of packets instead of fixed time interval is proposed in [12]. The advantage of this approach is that the intrusion detection is performed more rapidly during an attack and less frequently in the absence of an attack.

The classifier takes the values of the feature set as input and solves a binary classification (benign vs malicious) problem with the help of rules (population) stored in the rules database. In this study, we have evaluated the performance of the intrusion detection framework for different classification algorithms. These algorithms are described in Section 3. The classification decision made by the classifier is used to decide among two alternatives: (1) the packets are dropped and alarm is raised if the classifier decides that the packets are part of an attack; (2) otherwise, the traffic flow continues uninterrupted.

2.2 Process

The process of intrusion detection consists of two phases. In the training phase, the labeled packets of benign and attack traffic are passed to the intrusion detection system. The classifier *learns* its rules or evolves its population based on the features of the labeled traffic. For accurate classification, the intrusion detection system should be periodically updated to cater for changing usage behavior. Once the training phase is complete, the intrusion detection system performs its operation of classifying the incoming traffic based on the learned behavior during the training phase.

3. CLASSIFICATION ALGORITHMS

In this paper, we study the feasibility of using evolutionary classification algorithms in our framework for detecting SIP flooding attacks in IMS. We also include well-known non-evolutionary classifiers in our study to get a better understanding about the role of classifiers in our framework. We study the performance of classifiers with respect to their classification accuracy, and efficiency in terms of processing speed. The objective is to find a classifier that provides a consistent detection accuracy under various attack rates in realtime.

We have selected six well-known evolutionary and four non-evolutionary classifiers for our study. The criterion for selecting algorithms is to cover a wide range of classification paradigms used in machine learning and evolutionary computing. We have summarized the paradigms and related classifiers in Table 1. We now provide a brief description of each classifier to make the paper self contained. An interested reader can follow the inline references to find more details about the algorithms.

Table 1: Machine Learning Paradigms and Selected Algorithms (Ev=Evolutionary, NE=Non-Ev)

	Classification Paradigm	Algorithms
Ev	Evolving Neural Networks	EvRBF
Ev	Statistical Learning + GBML	Fuzzy AdaBoost
Ev	Pittsburgh-Style GBML	GAssist-ADI
Ev	Michigan-Style GBML	XCS, UCS
Ev	Ant Colony Optimization	c-AntMiner
NE	Decision Tree Induction	C4.5
NE	Support Vector Machines	C-SVM
NE	Instance Based Learning	KNN
NE	Statistical Modeling	Naïve Bayes

3.1 Evolutionary Algorithms

3.1.1 Evolving Radial Basis Function(EvRBF)

Neural Networks attempt to model the biological neural networks to solve complex real world problems. The RBF neural network is a two layer, feed-forward network. In RBF NNs, the radial basis function of the distance of each hidden neuron's central point from the input is calculated. Afterwards, the weighted sum is calculated by the output layer neurons as a function of the outputs of hidden layer neurons and the weights of the links connecting them to the output layer neurons. Evolving Radial Basis Function Neural Networks or Ev-RBF [19] make classification process simpler by automatically determining the values of RBF-NN parameters using evolutionary algorithms.

3.1.2 Fuzzy AdaBoost (Fuzzy-AB)

A fuzzy rule based classifier is defined by a fuzzy relationship that assigns each instance a degree of membership to each class; therefore, an instance may belong to multiple classes with different degrees of compatibility. The fuzzy classifiers are well suited for classification problems having imprecise ('fuzzy') boundaries. In the Fuzzy AdaBoost algorithm [8], boosting is used to combine multiple weak fuzzy hypotheses through a genetic iterative learning process to evolve a hybrid classifier that performs significantly better than any of the component hypothesis [8].

3.1.3 Genetic Classifier System (GAssist-ADI)

Genetic clASSIfier sySTem (GAssist) [5] is an evolutionary classification algorithm belonging to the Pittsburgh style Genetic-Based Machine Learning paradigm. In Pittsburgh approach, each individual in the population represents a *'complete solution to the classification problem'* [5]. GAssist uses genetic algorithm to evolve the individuals–which represent rulesets–of the population. GAssist uses a fitness function based on the Minimum Description Length (MDL) principle to make optimal trade-off between complexity and accuracy of the rulesets. Generalization of the rulesets is improved by using the incremental learning with alternating strata (ILAS) windowing scheme. We have used the adaptive discretization intervals (ADI) rule representation because our dataset contains only real values.

3.1.4 eXtended Classifier System (XCS)

XCS [22] is a Michigan style, rule-based, learning classifier system. In the Michigan approach, each individual represents a single rule and the whole population defines the complete ruleset. The initial rules are generated using a subset of the training data. New rules are periodically evolved using a niche genetic algorithm. The prediction accuracy of each rule (or individual) defines the fitness of that individual. The expected payoff values for all possible actions are stored in a prediction array. XCS algorithm has the ability to solve real world problems through evolution of accurate generalizations and real values representation [23].

3.1.5 sUpervised Classifier System (UCS)

UCS [11] is also a Michigan style, rule based, learning classifier system. It has been derived from XCS. The basic operation of UCS algorithm is similar to the XCS algorithm. However, there are some differences which separate UCS from XCS. The major difference between the two algorithms is their learning schemes (reinforcement scheme in XCS and supervised scheme in UCS). UCS does not maintain a prediction array. Since the fitness is defined in terms of the classification accuracy, UCS guarantees evolution of accurate classifiers. UCS can learn and evolve rules online, hence it is well suited for realtime online system.

3.1.6 Continuous Ant-Miner Algorithm (c-AntMiner)

Ant-Miner [14] is a data mining algorithm based on Ant Colony Optimization (ACO) paradigm. Continuous Ant-Miner algorithm [13] extends Ant-miner to classify using continuous (real-valued) attributes. Ant-Miner algorithm works on the principle of an ant colony's foraging patterns. Ants choose random paths in search of food and deposit a chemical 'pheromone' on their paths which evaporates with the passage of time. The optimal path to a food source–in a steady state–has relatively greater concentration of pheromone because more ants traverse on that path. This leads to selection of the optimal paths. The Ant-Miner algorithm maps this phenomenon in the classification domain. It begins with an empty rule set and creates one rule at a time by adding terms to the partial rule in a probabilistic manner. The rules are updated and pheromone is added iteratively based on the quality of the rule. Finally, the rules above a certain pheromone threshold are selected as the optimal rules [13].

3.2 Non-Evolutionary Algorithms

3.2.1 C4.5 Algorithm

C4.5 classification algorithm [16] builds a decision tree to solve the classification problem. The algorithm splits the training dataset into smaller subsets on the basis of each feature. The feature with maximum information gain is selected as a decision node. The algorithm is then repeated on the subsets until all the features have been evaluated or no additional information gain is achieved by splitting data using the remaining features. After the decision tree is made, the tree is pruned to remove useless branches.

3.2.2 Support Vector Machine (C-SVM)

Support Vector Machines [10] classify instances by establishing decision boundaries (hyperplanes) between the instances belonging to different classes. SVM creates the decision hyperplane by iteratively minimizing the error function while conforming to certain constraints. Based on the error function and the constraints, SVM has different variations of algorithms. The most commonly used SVM based classification algorithm is C-SVM. The Support Vector Machines

transform complex problems with overlapping instances in to non-overlapping objects by transforming the instances from the input space to the feature space using mathematical functions known as kernels. Some commonly used kernels are Linear, Polynomial, Radial Basis Function (RBF) and Sigmoid kernels.

3.2.3 K Nearest Neighbors Algorithm (KNN)

In K-Nearest Neighbors algorithm [7], the training phase involves only storing feature values and corresponding classes. The classification is done through the voting of K training instances which are least distant from the instance under test. The least distant instances ('nearest neighbors') are identified by calculating distance of the instance under test from all training instances. The class of the majority of the nearest neighbors is assigned to the instance under test.

3.2.4 Naïve Bayes Algorithm (NB)

Naïve Bayes algorithm [18] creates a probabilistic model for classification. Even though all features contribute towards the overall probability of classification, Naïve Bayes algorithm assumes that the features are statistically independent of one another. Although this assumption may not hold true for all cases, Naïve Bayes algorithm has shown promising results compared with other well-known classification algorithms in real world applications [18]. In technical terms, if $X = (x_1, x_2, x_3, ...)$ is the feature vector and C is a class, then the probability that this feature vector belongs to a class C is given by $P(X|C) = \prod_{i=1}^{n} P(X_i|C)$.

4. PERFORMANCE EVALUATION

In this section, we present our strategy for performance evaluation and comparison of above-mentioned algorithms. Our primary objective is not to bias our strategy to a particular learning paradigm or classifier; therefore, we first provide performance metrics that help in unbiased evaluation of algorithms. We then describe our test bed that generates SIP traffic. We also show how we inject flooding attacks into the benign traffic. Finally, we show the results of our experiments.

4.1 Performance Metrics

To compare the algorithms, we use four metrics to evaluate their performance:

True Positive (TP) Rate: The percentage of attack instances correctly labeled as attack,

False Positive (FP) Rate: The percentage of benign instances incorrectly labeled as attack,

Training Time: Time (in seconds) taken by the classifier to train itself on the provided training dataset, and

Testing Time: Time (in seconds) taken by the classifier to classify testing datasets. The time is averaged on five datasets containing benign traffic and different intensities of attack traffic embedded in it. (An attacker can launch flood of SIP traffic by exploiting some hardware or software vulnerability of the SIP devices.)

The True Positive (TP) rate shows the attack detection accuracy of the classifiers. A higher TP rate indicates better attack detection. The False Positive (FP) rate indicates the false alarms raised by the classifier. A classifier with high FP rate will deny service to the legitimate users. The training time is important for an online classifier which periodically updates its rule-base. The testing time shows the additional delay introduced in the traffic by the classifier. These four parameters are effective indicators of the overall classification accuracy and the speed of classification.

4.2 Testbed Setup

We have decided to synthetically generate SIP traffic in our IMS laboratory because of unavailability of publicly available SIP traffic datasets. The testbed simulated user agents communicating with each other through a SIP server. The SIP server and user agents are simulated using an open-source SIPp tool [9]. It has the ability to simulate customized SIP scenarios written in eXtensible Markup Language (XML) syntax. The SIPp agents can be controlled at run time using a predefined UDP port. The simulated SIP server and clients follow a typical SIP call flow as shown in Figure 2. To instantiate a call, the SIP client sends an `INVITE` request to the server. When the bell starts ringing on the callee's device, the SIP server sends back `180 RINGING` response message. The server sends a `200 OK` response message when call is picked up. The caller sends an `ACK` response and the call has been successfully established. To terminate a call, the client sends a `BYE` request. The server responds with a `200 OK` message which means that the call has successfully ended. In our testbed, we use constant call duration of 60 seconds. We have also simulated random packet losses to simulate the characteristics of a lossy network.

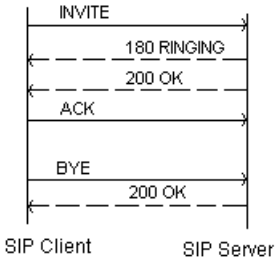

Figure 2: The Typical SIP Call Flow

4.3 Training and Testing Datasets

We used our test bed to generate five benign traffic datasets that have normal distribution call rates. The duration of each dataset is one hour. We define the server load as the mean of the normally distributed call rate. The server load for benign traffic datasets was set to 500 calls per min.

We divided flood attacks in to two categories based on the attack duration: 1) Chunk (prolonged) attacks, and 2) Harmonic (bursty) attacks. The chunk attacks model an attacker who launches a continuous flood of SIP requests for a long time in an attempt to overload the server. On the other hand, the harmonic attacks model an attacker who launches flood attacks in short bursts after frequent intervals.

For both chunk and harmonic attacks, we have also generated five flood attack scenarios with varying attack intensity. The duration of each chunk attack is on the average 11 min. The harmonic attacks contain multiple bursts each of 1 minute duration. The simulated attack rates are 10 cps (calls per second), 25 cps, 50 cps, 100 cps and 500 cps. The attacks are artificially injected in the benign traffic datasets after performing some time and user synchronization adaptations. In this way, we obtain ten datasets for our exper-

iments. Five datasets belong to chunk attack category and the remaining five datasets contain harmonic attacks. Each dataset is of one hour duration and contains attack of different intensities.

4.4 Feature Extraction

Our generic intrusion detection framework can be customized to use any feature set. We have chosen a subset of the list of features presented by authors of [12]. The selected features represent the complete distribution of the SIP messages, requests and responses, in a given traffic dataset.

The SIP messages can be classified in to two types: Request messages and Response messages. The request messages are used to bring some change in the session. For example, `INVITE` request is used to initiate a session. Similarly, the `BYE` request is used to signal termination of an established session. The response messages are used to notify of the result of the SIP request. For example, the `200 OK` message indicates success of the request. The list of features used in our experiments includes the number of `INVITE`, `REGISTER`, `BYE`, `ACK`, `OPTIONS`, `CANCEL`, `UPDATE`, `REFER`, `SUBSCRIBE`, `NOTIFY`, `MESSAGE`, `INFO` and `PRACK` requests. These values are normalized to the total number of SIP request messages. The features also include the number of `Informational`, `Success`, `Redirection`, `Client Error`, `Server Error`, `Global Error` response messages that are normalized to the total number of SIP response messages. The feature values are calculated over a buffer size of 500 packets. This buffer size is chosen as a function of the the average load of a server under normal no attack scenario. The selected features capture the complete pattern of SIP traffic. Any deviation in values of these features reflects change in pattern of SIP messages. This situation indicates an anomalous– possibly attack–scenario. The feature values are extracted using a custom code extraction module in `C++`. The instances are stored in ARFF format.

4.5 Classification

To evaluate the classifiers (all except cAntMiner), we use the KEEL software [3]. We have used the cAntMiner implementation of [13]. All algorithms in KEEL and cAntMiner are used with their standard parameter values. This is done to avoid any bias towards a particular classifier. The classifiers are trained using the traffic dataset that contains 100 cps attack. The testing is done in two steps. First, we perform experiments on the five datasets containing chunk attacks of different intensities embedded in the benign traffic each of one hour duration. Then, these experiments are repeated for harmonic attacks. For each set of experiments, we calculate the performance metrics for each classifier. The accuracy results are reported in the next section. The timing analysis is reported by taking average of the time taken by the classifier on the five datasets.

5. EXPERIMENTAL RESULTS

In this section, we present the results of our experiments. We discuss in detail the performance of classifiers in terms of the performance metrics (see Section 4.1). These metrics include *true positive rate*, *false positive rate*, *training time* and *testing time*. The classification accuracy (tp and fp) results have been summarized in Table 2 for chunk attacks and Table 3 for harmonic attacks. The efficiency of the classifiers (training and testing time) is given in Table 4.

5.1 Classification Accuracy

The results for TP and FP rates of the classifiers for the chunk and harmonic attacks with different variations of attack intensity are discussed below:

5.1.1 Training Accuracy

From Table 2, we can see that all algorithms show good training accuracy for chunk attacks. Most of the algorithms achieved 100% TP rate with 0% FP for the training dataset. The Naïve Bayes algorithm achieved the minimum training accuracy. However, this is an expected result because the SIP requests and responses have a certain correlation with each other. For example, an `INVITE` request is usually followed by a `180 Ringing` response. This correlation (or dependence) of SIP messages violates the Naïve Bayes' primary assumption of independence of feature values. Despite this *naïvety*, the TP rate of 99.50% is still quite high. It means that an attack of 100 cps has been reduced to only 0.5 cps.

The harmonic attacks are of much shorter duration as compared to chunk attacks. Moreover, they are distributed in the whole dataset. This diffused nature of harmonic attacks makes their detection a challenging task. Although all algorithms show good training accuracy for chunk attacks, there is significant difference in their training accuracy for harmonic attacks. Table 3 shows that C-SVM with (RBF kernel) has the best training accuracy (100% TP rate and 0% FP rate). GAssist-ADI and C4.5 also have 100% TP rate for the training dataset, they achieve this at the cost of 0.32% and 0.63% FP rate respectively. cAntMiner has the minimum TP rate of 95.60% on the training dataset. KNN has the highest FP rate of 0.95%.

5.1.2 Testing Accuracy

First we discuss the testing accuracy of classifiers for the case of chunk attacks. Table 2 shows the testing accuracy of the classifiers for the five datasets with chunk attacks. All of the non-evolutionary algorithms failed to detect a stealthy flood rate of 10 cps. In comparison, four evolutionary algorithms (Fuzzy-AdaBoost, GAssist-ADI, UCS and XCS) were able to evolve rules that filtered some of the stealthy flood attack packets. The detection accuracies of GAssist-ADI and UCS were significant enough to raise alarm of a suspicious activity. GAssist achieved the TP rate of 25.32% for 10 cps attack.

A further look at Table 2 shows a similar trend for attack rate of 25 cps. Three non-evolutionary algorithms (C4.5, C-SVM and KNN) and two evolutionary algorithms (Ev-RBF and c-AntMiner) failed to detect flood attacks as low as 25 cps. Naïve Bayes is the only non-evolutionary algorithm that partially (14.90%) detects the 25 cps attack. Both GAssist-ADI and UCS stay on top showing impressive increase (42.15% and 85.53% respectively) in their TP rate. UCS achieved the highest TP rate (98.19%) for 25 cps attack. The good detection accuracy of evolutionary classifiers (GAssist-ADI and UCS) for the chunk attacks indicates that the rules generated through evolution are general enough to capture all variations of attack. At the same time, the evolved rules are more accurate than any other algorithm.

Some people may argue that the attack rates of 10 cps and 25 cps are too low, hence there is no significance of detection at these rates. We agree that these attack rates are too low to cause significant DoS effect, yet such a stealthy attack may signify a suspicious activity e.g. beginning of

Table 2: Classification Accuracy (in %) of selected algorithms for Chunk Flood Attacks

	Training		Testing											
Attack Rate	**100 cps**		**10 cps**		**25 cps**		**50 cps**		**100 cps**		**500 cps**		**Average**	
	TP	**FP**	**TP**	**FP**	**TP**	**FP**	**TP**	**FP**	**TP**	**FP**	**TP**	**FP**	**TP**	**FP**
EvRBF	100	0	0	0	0	0	**87.06**	0	100	0	**0.12**	0	37.44	0
Fuzzy-AdaBoost	100	0	3.80	0	4.22	0	99.00	0	100	0	100	0	61.40	0
GAssist-ADI	100	0	**25.32**	0	67.47	0	100	0	100	0	100	0	78.56	0
UCS	100	0	12.66	0	**98.19**	0	100	0	100	0	100	0	**82.17**	0
XCS	100	0	2.41	0	3.80	0	100	0	100	0	100	0	61.24	0
c-AntMiner	99.51	0	0	0	0	0	99.26	0	99.51	0	99.87	0	59.73	0
C4.5	100	0	0	0	0	0	**0**	0	100	0	100	0	40.00	0
C-SVM (RBF)	100	0	0	0	0	0	99.50	0	100	0	100	0	59.90	0
C-SVM (Linear)	100	0	0	0	0	0	99.50	0	100	0	100	0	59.90	0
C-SVM (Poly)	100	0	0	0	0	0	96.52	0	100	0	100	0	59.30	0
C-SVM (Sigmoid)	100	0	0	0	0	0	**100**	0	100	0	100	0	60.00	0
KNN	100	0	0	0	0	0	99.50	0	100	0	100	0	59.90	0
Naïve Bayes	99.50	0	0	0	**14.90**	0	99.30	0	99.50	**0.04**	99.90	**0.04**	**62.72**	**0.02**

Table 3: Classification Accuracy (in %) of selected algorithms for Harmonic Flood Attacks

	Training		Testing											
Attack Rate	**100 cps**		**10 cps**		**25 cps**		**50 cps**		**100 cps**		**500 cps**		**Average**	
	TP	**FP**	**TP**	**FP**	**TP**	**FP**	**TP**	**FP**	**TP**	**FP**	**TP**	**FP**	**TP**	**FP**
EvRBF	97.80	0.32	0	0	0	0	**93.44**	**0**	97.80	0.32	**99.13**	**0.32**	58.08	0.13
Fuzzy-AdaBoost	98.90	0.32	0	0	31.82	0	95.08	0.32	98.90	0.32	99.13	0.96	64.99	0.32
GAssist-ADI	100	0.32	0	0	**79.55**	**0**	93.44	0.32	**100**	0.32	99.13	0.64	74.42	0.25
UCS	97.80	0.32	0	0	0	0	**4.92**	0	97.80	0.32	98.84	0.32	40.31	0.13
XCS	97.80	0.32	0	0.32	2.27	0	91.80	0	97.80	0.32	**99.13**	**0.32**	58.20	0.19
c-AntMiner	95.60	0.63	0	0.63	**95.45**	0.63	95.08	0.63	95.60	0.63	98.84	0.96	**77.00**	0.70
C4.5	100	0.63	0	0	**90.91**	0.32	95.08	0.63	100	0.63	99.42	0.96	**77.08**	0.51
C-SVM (RBF)	**100**	**0**	0	0	68.18	0.32	95.08	0.63	**100**	**0**	**99.71**	0.96	72.59	0.38
C-SVM (Linear)	97.80	0.63	0	0	31.82	0.32	95.08	0.32	97.80	0.63	99.42	0.64	64.82	0.38
C-SVM (Poly)	98.90	0.32	0	0	20.45	0	63.93	0	98.90	0.32	99.42	0.64	56.54	0.19
C-SVM (Sigmoid)	97.80	0.63	0	0	20.45	0.32	95.08	0.32	97.80	0.63	99.42	0.64	62.55	0.38
KNN	96.70	0.95	0	0	31.82	0	86.89	0	96.70	0.95	99.42	0.64	62.97	0.32
Naïve Bayes	97.80	0.60	0	0	56.80	0.30	**95.10**	0.60	97.80	0.60	99.40	1.00	69.82	0.50

a DoS or distributed DoS attack, brute force attempts to hack passwords etc. Therefore, the detection of such stealthy attacks may prove valuable in certain scenarios.

According to Table 2, C-SVM using sigmoid kernel function, GAssist-ADI, UCS and XCS detected 50 cps chunk attack with 100% TP rate. C4.5 algorithm was unable to detect the 50 cps attack. Although, TP rate of EvRBF improved significantly (87.06%), yet it is far lower as compared to other algorithms.

All algorithms performed well at chunk attacks of 100 cps because they are trained for this attack intensity. From Table 2, we note that all algorithms (except EvRBF) have good detection accuracy at 500 cps chunk attack. EvRBF gives poor (0.12%) TP rate for 500 cps attack. This is an excellent example of the overfitting problem. This means that the neural network that evolved as a result of training of RBF classifier, gives good accuracy at (and near) the attack intensity to which it is trained; whereas it gives poor accuracy when the attack intensity increases or decreases significantly.

Table 2 shows the average values of the TP rates of all classifiers for the chunk attacks. The top 3 algorithms are UCS (82.17%), GAssist-ADI (78.56%) and Naïve Bayes (62.72%). UCS stands out to be the most accurate evolutionary classifier for chunk attacks while Naïve Bayes turns out to be the most accurate non-evolutionary classifier for chunk attacks.

Now, we discuss the classification accuracy results of classifiers for the harmonic attacks. Table 3 shows that none of the classifiers were able to detect stealthy harmonic attack of 10 cps. This result is not surprising as very low rate harmonic attacks tend to diffuse in the benign traffic. The performance of the classifiers has a significant improvement for 25 cps harmonic attacks. From Table 3, we see that c-AntMiner has the highest TP rate (95.45%) with an FP rate of 0.63%. C4.5 is the best non-evolutionary algorithm at this attack rate with TP rate of 90.91% and FP rate of 0.32%. If we impose the requirement of zero false alarm rate, GAssist-ADI shows the best TP rate of 79.55%. EvRBF, UCS and XCS fail to detect 25 cps harmonic attacks.

All algorithms (except UCS) have good TP rate for 50 cps harmonic attacks. Table 3 shows that Naïve Bayes has the best TP rate (95.10%) for an FP rate of 0.60%. EvRBF is the best evolutionary classifier with 93.44% TP rate and 0% FP rate. Similarly, for 100 cps harmonic attacks, C-SVM (with RBF kernel) gives 100% TP rate with no false alarm. GAssist-ADI is the best evolutionary classifier in this case (100% TP rate and 0.32% FP rate). C4.5 algorithm also gives 100% attack detection for 100 cps harmonic attack at the cost of 0.63% FP rate. From Table 3, we see that all algorithms successfully detect the 500 cps harmonic attacks. C-SVM (with RBF kernel) achieved the best TP rate of 99.71% with FP rate of 0.96%. Both EvRBF and XCS show minimum FP rate of 0.32% and TP rate of 99.13%.

A comparison of the average TP rates of classifiers for the harmonic attacks in Table 3 shows that C4.5 and cAntMiner classifiers are the most accurate classifiers for detection of harmonic attacks. The two classifiers have very close average TP rates (77.08% for C4.5 and 77.00% for cAntMiner). GAssist-ADI is at the third position with 74.42% average TP rate and 0.25% average FP rate.

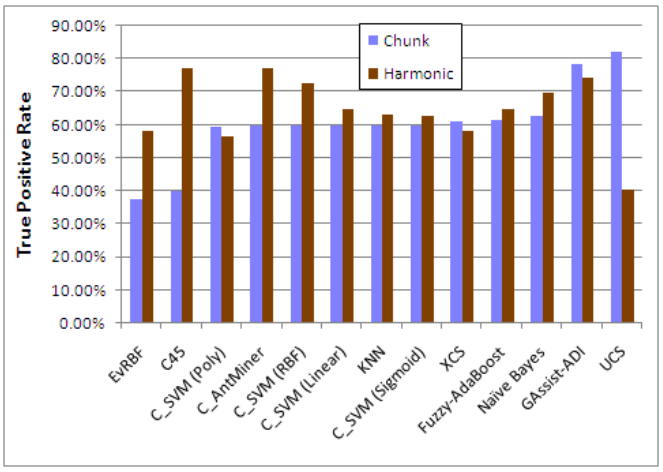

Figure 3: Correlation of Average TP Rates for Harmonic and Chunk Attacks

Figure 3 shows correlation of TP rates of classifiers for chunk and harmonic attacks. Note that most of the TP rates in the center of the figure are distributed around the 60% mark. The TP rates for chunk attacks are increasing towards the right and decreasing towards the left. Also note that the TP rates for harmonice attacks are increasing on both sides and decreasing in the middle and at the ends. Therefore, the optimal classification point lies towards the right side. Figure 3 shows that GAssist-ADI achieves the optimal accuracy for both chunk and harmonic attacks. If we restrict ourselves to non-evolutionary algorithms, Naïve Bayes classifier becomes the optimal choice for the problem at hand.

5.2 Efficiency (Speed) of Classification

The results for time taken by classifiers in training and testing phases are as follows:

5.2.1 Training Time

Table 4 shows that the non-evolutionary algorithms outperform evolutionary algorithms (except c-AntMiner) in training time. The training time of KNN is 0 sec because it performs all computations at the time of classification (i.e. testing time). The c-AntMiner algorithm has the lowest training time in the evolutionary classifiers. This is due to the fact that c-AntMiner uses no mutation and crossover operations which consume most of the time during the training of other evolutionary classifiers. The Pittsburgh style LCS (GAssist-ADI) consumes less training time as compared to the Michigan style LCS (XCS and UCS).

5.2.2 Testing Time

Table 4 shows that C4.5, C-SVM and Naïve Bayes algorithms have very small testing time similar to their smaller training times. However, KNN has a greater testing time as it performs all computations at testing time. The c-AntMiner algorithm has the lowest testing time in the evolutionary classifiers. Other evolutionary algorithms have relatively higher testing time. The Michigan style LCS (XCS and UCS) have smaller testing time as compared to the Pittsburgh style LCS (GAssist-ADI).

Table 4: Average Training and Testing Time (in seconds)

	Chunk Attacks		Harmonic Attacks	
Algorithms	**Training**	**Testing**	**Training**	**Testing**
EvRBF	9	2	9	1
Fuzzy-AdaBoost	43	6	17.4	3
GAssist-ADI	73	14	44	7
UCS	119.6	3.3	65.1	0.8
XCS	149.1	4.1	144.6	1.6
c-AntMiner	<1	<1	<1	<1
C4.5, C-SVM, NB	<1	<1	<1	<1
KNN	0	3.8	0	1.6

6. INFERENCES & GUIDELINES

It is imperative from the results that it is difficult to identify a single classifier as the *best classifier in all scenarios*. If we compare the top three classifiers for chunk attacks (UCS, GAssist-ADI and Naïve Bayes), UCS is the most accurate classifier and Naïve Bayes is the fastest classifier. Moreover, some of the algorithms that perform best for one category of attack (chunk or harmonic), they have less than ideal performance for the attacks of other category. UCS is an example of such algorithms. Although, GAssist-ADI is accurate for both categories of attack, its large testing time makes it unsuitable for servers handling heavy traffic load.

Now we focus our attention–from the perspective of a VoIP operator– to a number of real-world scenarios. We analyze the requirements of each scenario and then present valuable guidelines, inferred from the results of our experimental study, to VoIP operators that will help them in customizing our framework to their needs before deploying them as a security firewall in front of their VoIP servers.

6.1 Scenario 1:

We consider the scenario of a small enterprise that has a local SIP server. The average traffic load on the SIP server is expected to be relatively small. In this scenario, the system administrators are expected to identify a stealthy attack with the help of a classifier that processes incoming traffic within an acceptable delay.

Guideline 1. We recommend that UCS classifier is ideally suited to handle this scenario. It is clear from the results of our experiments that UCS can detect with relatively more accuracy the stealthy chunk attacks and it also has relatively low testing time as well. Consequently, it can raise alarms in realtime for suspicious activity.

6.2 Scenario 2:

We now consider the scenario of a SIP server that is being used in a large enterprise. In this case, as expected, the large volume of VoIP calls would be flowing through the SIP server. In this situation, the efficiency requirements are very strict compared with the previous scenario–the classifier should be able to identify stealthy attacks but now at a relatively higher speed.

Guideline 2: We recommend Naïve Bayes for this scenario because it provides good detection accuracy at high attack rates. It can detect a number of stealthy attacks as well as high rate attacks with a reasonable amount of accuracy. Moreover, its processing overhead is negligible Consequently, Naïve Bayes will not put additional processing load on this relatively highly loaded server and also raise alarms in real time for malicious activity.

6.3 Scenario 3:

The next scenario is of a backbone SIP server that is handling the traffic load of multiple enterprises. The traffic volumes are very large and now the classification should meet very strict timing requirements. The low rate stealthy attacks cannot degrade the performance of a SIP server; therefore, this operator is not concerned with detecting SIP attacks. However, medium and high rate attacks must be detected at a wire speed and efficiently mitigated in real-time to avoid overloading of the server to a level where it can result in a DoS attack.

Guideline 3: We recommend using either Naïve Bayes or C-SVM (with the RBF kernel) because both of them meet the strict timing and accuracy requirements of this scenario. However, our experience is C-SVM (RBF) provides almost 100% detection accuracy at high attack rates. Moreover, it also has the smallest training and testing times; therefore, we recommend C-SVM for this scenario.

6.4 Scenario 4:

Finally, we consider the scenario of an operator, which is carrying an online troubleshooting operation or the offline analysis of the dumped traffic. The system administrators routinely do such a forensic analysis to identify attempts of hacker to launch malicious attacks–both stealthy and high rate. Moreover, they also identify–using the dumped trace–malfunctioning device. In this scenario, the timing requirements are really not important; however, detection accuracy is the key requirement.

Guideline 4: We recommend that the forensic expert should use a team of UCS and GAssist-ADI classifiers for this scenario because these two classifiers have the best accuracy for stealthy attacks. Their combination will cover stealthy attacks at 10 cps to very high rate attacks upto 500 cps.

7. CONCLUSION & FUTURE WORK

In this paper, we have presented a generic intrusion detection system with a flexible set of features and classifiers for SIP floods attacks. We have done a comprehensive study of six evolutionary and four non-evolutionary classifiers. An important contribution of this work is a set of guidelines that VoIP operators–working in the telecommunication industry–can utilize to customize our generic intrusion detection systems for their SIP server. To the best of our knowledge, this is the first investigative study of evolutionary and non-evolutionary classifiers to develop an online realtime intrusion detection system for solving the SIP flood attacks problem in IMS. The focus of our future work is to study the effect of selection of feature on the classification accuracy. We also plan to extend this work for mitigation of other security threats in IMS.

Acknowledgments

This work is supported by the National ICT R&D Fund, Ministry of Information Technology, Government of Pakistan. The information, data, comments, and views detailed herein may not necessarily reflect the endorsements of views of the National ICT R&D Fund.

8. REFERENCES

[1] 3GPP2. IMS Security Framework. *http://www.3gpp2.org*, Dec. 2003.

[2] M. Akbar et al. A Comparative Study of Anomaly Detection Algorithms for Detection of SIP Flooding in IMS. In *IMSAA*, 2008.

[3] J. Alcala-Fdez et al. KEEL: A software tool to assess evolutionary algorithms to data mining problems. *Soft Computing*, 2008.

[4] A. Awais et al. Attack analysis & bio-inspired security framework for IP Multimedia subsystem. *GECCO*, pages 2093–2098, 2008.

[5] J. Bacardit. Pittsburgh Genetics-Based Machine Learning in the Data Mining era: Representations, generalization, and run-time. *PhD disertation*, 2004.

[6] A. Cuevas et al. The IMS Service Platform: A Solution for Next-Generation Network Operators to Be More than Bit Pipes. *IEEE Comm. Mag.*, pages 75–81, 2006.

[7] B. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. Los Alamitos, CA: IEEE Computer Society Press, 1991.

[8] M. del Jesus et al. Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms. *Fuzzy Systems, IEEE Trans. on*, 12(3):296–308, 2004.

[9] R. Gayraud. SIPp, 2007. http://sipp.sourceforge.net.

[10] P. Lewicki et al. *Statistics: Methods and Applications*. StatSoft, Inc., 2006.

[11] E. Mansilla et al. Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks. *Evo. Comp.*, 11(3):209–238, 2003.

[12] M. Nassar et al. Monitoring SIP Traffic Using Support Vector Machines. In *RAID*, pages 311–330. Springer-Verlag Berlin, Heidelberg, 2008.

[13] F. Otero et al. cAnt-Miner: An Ant Colony Classification Algorithm to Cope with Continuous Attributes. In *ANTS*, pages 48–59, 2008.

[14] R. Parpinelli et al. An Ant Colony Algorithm for Classification Rule Discovery. *Data Mining: a Heuristic Approach*, 208, 2002.

[15] Poikeselka et al. *The IMS IP Multimedia Concepts and Services*. John Wiley & Sons, Ltd., 2nd edition, 2006.

[16] J. Quinlan. Improved Use of Continuous Attributes in C4.5. *JAIR*, 4:77–90, 1996.

[17] Y. Rebahi et al. Detecting flooding attacks against IP Multimedia Subsystem (IMS) networks. *AICCSA, 2008.*, pages 848–851, 2008.

[18] I. Rish. An empirical study of the naive Bayes classifier. In *Proc. IJCAI-01 Workshop on Empirical Methods in AI*, volume 335, 2001.

[19] V. Rivas et al. Evolving RBF neural networks for time-series forecasting with EvRBF. *Information Sciences*, 165(3-4):207–220, 2004.

[20] SANS Institute. SANS Top-20 2007 Security Risks, 2007. http://www.sans.org/top20/.

[21] M. Sher et al. Secure Service Provisioning Framework (SSPF) for IP Multimedia System and Next Generation Mobile Networks. *IWWST'05*, pages 101–106, April 2005.

[22] S. Wilson. Generalization in the XCS classifier system. In *Proc. Genetic Programming*, pages 665–674. Morgan Kaufmann, 1998.

[23] S. Wilson. Get Real! XCS with Continuous-Valued Inputs. *LNCS*, pages 209–222, 2000.