Int. J. Inf. Secur. (2016) 15:131–143
DOI 10.1007/s10207-015-0281-1

REGULAR CONTRIBUTION

# Behavior-based approach to detect spam over IP telephony attacks

**Randa Jabeur Ben Chikha**[1] · **Tarek Abbes**[1] · **Wassim Ben Chikha**[2] · **Adel Bouhoula**[1]

Published online: 24 March 2015


**Abstract** Spam over IP telephony (SPIT) is expected to become a serious problem as the use of voice over IP grows. This kind of spam is appreciated by spammers due to its effectiveness and low cost. Many anti-SPIT solutions are applied to resolve this problem but there are still limited in some cases. Thus, in this paper, we propose a system to detect SPIT attacks through behavior-based approach. Our framework operates in three steps: (1) collecting significant calls attributes by exploring and analyzing network traces using OPNET environment; (2) applying sliding windows strategy to properly maintain the callers profiles; and (3) classifying caller (i.e., legitimate or SPITter) using ten supervised learning methods: NaïveBayes, BayesNet, SMO RBFKernel, SMO PolyKernel, MultiLayerPerceptron with two and three layers, NBTree, J48, Bagging and AdaBoostM1. The results of our experiments demonstrate the great performance of these methods. Our study, based on receiver operating characteristics curves, shows that the AdaBoostM1 classifier is more efficient than the other methods and achieve an almost perfect detection rate with acceptable training time.

✉ Randa Jabeur Ben Chikha
randa.jabeur@supcom.tn; jabeur_randa@yahoo.fr

Tarek Abbes
tarek.abbes@isecs.rnu.tn

Wassim Ben Chikha
wassimbenchikha@gmail.com

Adel Bouhoula
adel.bouhoula@supcom.rnu.tn

[1] Digital Security Research Unit, Higher School of Communication of Tunis (Sup'Com), University of Carthage, Cité El Ghazala, Tunisia

[2] SERCOM laboratory, Tunisia Polytechnic School, Carthage University, 2078 La Marsa, Tunisia

**Keywords** VoIP · SPIT detection · Behavior-based approach · Supervised learning methods · ROC

## 1 Introduction

VoIP is a family of technologies that can offer both voice communications and multimedia sessions over internet protocol (IP) networks. This technology is rapidly adopted by consumers and enterprises since it offers more functionalities and higher flexibility than traditional telephony. Two kinds of protocols are used in most VoIP calls, which are signaling protocol and media transmission protocol. Session initiation protocol (SIP) is the most adopted in signaling protocol and real-time transport protocol (RTP) is the most adopted in media transmission.

With the increase in VoIP applications, VoIP threats appear and become more and more a problem. SPIT, known as unsolicited and unwanted calls sent via VoIP networks, is one of these threats. Attacker prefers to make SPIT calls because it can be done quickly and with a low cost. In fact, each VoIP account has an associated IP address. Thus, SPITters can easily send their calls to thousands of IP addresses. As a result, SPIT can annoy VoIP users. In addition, VoIP network can be overloaded by intensive messages.

In light of this, countermeasures are proposed in order to identify and filter SPIT. These countermeasures are mostly derived from the experience of SPAM defense. They include the reputation-based [1], call frequency-based [2], dynamic blacklisting, fingerprinting [3], challenging suspicious calls by captchas [4] and the use of more sophisticated machine learning. Under this latter method, Nassar et al. [5,6] perform a supervised learning in order to detect attacks on SIP protocol. They apply the support vector machine (SVM) classifier on the calls history logfile. Moreover, Wu et al. [7]

propose a semi-supervised learning approach using the metric pairwise constrained k-means method (MPCK-Means) to discover SPIT calls.

In this paper, we propose a SPIT detection system through behavior-based approach. The main contributions of this paper are as follows: (1) the design of SPIT detection system including a large number of identification criteria, (2) the application of "sliding windows" strategy to properly maintain the callers profiles and (3) the investigation of ten supervised classification methods to recognize SPITters. Depending on the scope and criteria used in classification, each method has shown its effectiveness on many real-life data and has been applied to a wide range of applications. Moreover, it is too difficult to prior know the most efficient classifier in such scenario. Thus, the purpose of this comparison is to determine the characteristic of each classifier in terms of both SPIT detection and convergence speed. Here, through a comparative study, we prove that the AdaBoostM1 outperforms the other classification methods.

The rest of the paper is organized as follows. In Sect. 2, we mention some related works about anti-SPIT mechanisms and the motivations of using machine learning algorithms. The proposed system for detecting SPIT attacks is developed in Sect. 3. We describe our simulation scenario in Sect. 4, and then, we present our experiment results in Sect. 5. Finally, Sect. 6 summarizes this work and enumerates our future works.

## 2 Related works

### 2.1 Anti-SPIT mechanisms

SPIT becomes a severe threat for VoIP users because of the reduction in voice call costs, comparing to current infrastructures, and the lack of a global legal and regulatory framework. It affects the private life of the customer and his correspondences and becomes a source of noise for them. It is classified as social threats by Keromytis [8]. To resolve this problem, many anti-SPIT mechanisms and techniques have been proposed. Bai et al. [9] have classified these mechanisms into four categories: list-based filtering, reputation-based filtering, Turing test and pattern-based filtering.

#### *2.1.1 List-based filtering*

It identifies SPIT according to three types of filtering lists: blacklist, whitelist and graylist. The principle is to block spam calls from blacklist, accept user calls from whitelist and temporarily reject unclassified calls from graylist [10–13]; Mathieu [14] proposed a framework including blacklists and whitelists in order to apply statistical traffic analysis method using the number and duration of calls. Nevertheless, list-based filtering approach has some limits. Indeed, the spammers can easily change their addresses to elude detection. Therefore, the list-based filtering is vulnerable to Sybil attack [15], a threat against identity in which an individual entity masquerades as multiple simultaneous identities.

#### *2.1.2 Reputation-based filtering*

The main principle is to compute a reputation score in order to detect SPIT attacks [16]. The score is generated using buddy list and user ratings for buddies. In fact, the user can only trust people who have a high reputation score. In a similar work, Patankar et al. [17] propose using buddy lists inside the VoIP network in order to establish a trust chain between the caller and the callee. Besides, Balasubramaniyan et al. [18] employ the call durations to build social network relationships and global reputations for users [19]. This approach suffers from the information poisoning attack if it is set up to allow unknown but highly ranked persons to contact the user [20].

#### *2.1.3 Turing test*

Basically, the Turing test is a test of a machine's ability to exhibit intelligent behavior. Used to detect SPIT, this approach employs some parameters, such as call duration and silence period to detect and block spam calls. Related to this mechanism, Quittek [21] creates a modular SPIT detection platform in NEC VoIP SEAL. It is a Turing test that monitors the call communication patterns. However, Turing test has a limit because it is not applicable for human-initiated SPIT.

#### *2.1.4 Behavior-based filtering*

It monitors the call patterns and compares the current call pattern with the previous caller call pattern. If there is a significant difference, it blocks the call. Kusumoto et al. [22] proposed in a scoring system using call patterns and Naive Bayes approach to detect spammer. In this work, computed call patterns include different parameters such as user relationships, the rate of unsuccessful calls and the average call duration. The advantage of this system is that changing the values of these parameters is difficult for spammers.

In Table 1, a qualitative comparison of the techniques and the identification criteria supported by different previous approaches is presented. It is clearly shown that our proposed SPIT detection approach includes many techniques and large number of identification criteria. These latter will give our proposed approach and their benefits and hence can result a higher performance of SPIT detection.

As seen in Table 1, we have used only some identification criteria in our previous work [23]. Other important criteria such as variance of inter-call duration (VICD), variance of talk duration (VTD) and percentage of calls with

**Table 1** Qualitative comparison of the techniques and the identification criteria used by existing SPIT detection approaches

| | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 |
|---|---|---|---|---|---|---|---|---|
| *Techniques* | | | | | | | | |
| List-based filtering | | ✓ | ✓ | | | ✓ | ✓ | ✓ |
| Reputation-based filtering | | | | ✓ | | | | |
| Turing test | | | | | ✓ | | | |
| Behavior-based filtering | ✓ | ✓ | | | | ✓ | | ✓ |
| Sliding window | | | | | | | | ✓ |
| Boosting and bagging classifiers | | | | | | | | ✓ |
| *Identification criteria* | | | | | | | | |
| Percentage of answered calls | | | | | | | ✓ | ✓ |
| Percentage of rejected calls | ✓ | | | | | ✓ | ✓ | ✓ |
| Percentage of calls with invalid callees | | | | | | | | ✓ |
| Number of calls attempts | ✓ | ✓ | | | | | ✓ | ✓ |
| Call duration | ✓ | ✓ | | ✓ | | ✓ | ✓ | ✓ |
| Inter-call duration | | | | | | | | ✓ |
| Number of destination users | | | | | | | ✓ | ✓ |
| Silence length | | | | | | | ✓ | ✓ |
| Talk duration | | ✓ | | | | | | ✓ |

*P1* Nassar et al. [5], *P2* Wu et al. [7], *P3* Mathieu et al. [14], *P4* Balasubramaniyan et al. [18], *P5* Quittek et al. [21], *P6* Kusumoto et al. [22], *P7* Jabeur Ben Chikha et al. [23], *P8* the proposed SPIT detection

invalid callees (%IC) are not considered. Indeed, SPITters cannot give any importance to these criteria and initiate calls with constant inter-call duration and/or constant talk duration and/or great percentage of calls with invalid callees. In addition, for each call, we performed six tests which is costly in terms of processing time. Therefore, this algorithm has considerable complexity which is not desirable for a real-time system.

In this work, our main contribution is to propose a behavior-based approach to detect spam over IP telephony attacks. To this end, we have included VICD, VTD and %IC to the set of criteria. Thus, for each call, the proposed algorithm select nine identification criteria from the network traces of signaling and voice activities that allow for better distinguishability between legitimate and SPITter UAs. Since SPIT communication can be real-time and can cause significant network overload and considerable user annoyance, we have employed the strategy of "sliding window". The purpose of this strategy is to collect continuously the relative network traces window-by-window. Moreover, it can ensure the good storage of communication data flow in real time [24,25]. Contrary to [23], the classification process is done at each window using supervised classification (and not at each call). Consequently, the complexity of the proposed system is reduced compared to [23]. To determine the adequate classifier for the proposed system, we have carried out a comparative study in terms of SPIT detection and convergence speed between ten supervised classification methods (Naive Bayes, BayesNet, SMO RBFKernel, SMO PolyKernel, MultiLayerPerceptron with two and three layers, NBTree, J48, bagging and AdaboostM1).

## 2.2 Machine learning algorithms

Machine learning, a branch of artificial intelligence, allows a machine to evolve through a learning process and so performs tasks that are difficult or impossible to complete by traditional algorithms. Machine learning algorithms generally fall into two categories: supervised or unsupervised. We call unsupervised learning or clustering when the system or the operator has only examples, but not labels, and the number of classes and their nature were not predetermined. If classes are predetermined and examples are known, the system learns to classify according to a classification model, and it is called supervised learning.

Our work focuses on the supervised learning machine. In this context, we describe classification algorithms that we use to evaluate our framework.

### *2.2.1 Bayesian methods*

Bayesian methods are used as classification solutions in different domains such as data mining. In our work, we use two Bayesian methods called Naive Bayes and Bayesian networks.

*Bayesian networks method (BayesNet)* BayesNet is structured as a combination of a directed acyclic graph of nodes and links, and a set of conditional probability tables [26].

Thus, nodes correspond to classes or criteria, and links represent the relationship between nodes. The strength of these links is determined by the use of conditional probability tables. For each node, there is one probability table that represents the probability distribution of the node's parent. If a node is not connected (has no parents), the probability distribution is unconditional. If a node is connected to one or more parents, the probability distribution is a conditional distribution where the probability value of each node depends on the values of the parents.

*Naive Bayes method* Naive Bayes method is based on the Bayesian theorem [27]. A Naive Bayes network has a simple and unique structure with only two levels. The first level contains a single parent node, and the second contains several children with strong naive assumption of independence among child nodes. In simple terms, a Naive Bayes classifier assumes that the existence of a characteristic for a class is independent of the existence of other characteristics. Thus, a Naive Bayes network is a simple classifier, easy to program, and often effective. However, it is very sensitive to the presence of correlated attributes.

#### *2.2.2 Multilayer Perceptron (MLP)*

MLP network is the most widely used neural network classifier. Here, neurons are arranged in several layers. Each neuron of a layer is connected to all the neurons of the next layer. MLP is composed of an input layer, one or more hidden layers and an output layer. The addition of this hidden layer allows the network to model the functions of complex nonlinear decision between any input and output space. In a classification problem, each output neuron is dedicated to a given class. MLPs can resolve the problems when one has little information about the relationship between input vectors and their corresponding outputs. The most popular learning algorithms used for MLP is the backpropagation [28].

#### *2.2.3 Decision tree*

A decision tree is a decision support tool that uses a tree graph of decisions and their possible consequences. It is an excellent tool for helping to decide between several courses of action. In our work, we use two decision tree methods: C4.5 decision tree and Naive Bayes tree.

*C4.5 decision tree (C4.5)* C4.5 creates a model based on a tree structure [29]. In the tree, nodes and branches correspond to criteria and possible values connect criteria, respectively. A leaf that symbolizes the class terminates a set of nodes and branches. Thus, tracing the path of nodes and branches to the terminating leaf leads to determine the class of an instance. J48 algorithm is an implementation of the C4.5.

*Naive Bayes tree (NBTree)* NBTree adopts a hybrid model of a decision tree classifier with a Naive Bayes classifier [30]. Therefore, the NBTree model is a decision tree of nodes and branches where each leaf contains a Naive Bayes classifier.

#### *2.2.4 Support vector machines (SVM)*

SVM was first suggested by Vapnik [31] in 1960 for data classification. It is a supervised machine learning method based on the statistical learning theory. A more detailed description can be found in various literatures [32–34]. SVM classifies data in large data sets by identifying a linear or nonlinear separating surface in the input space of a data set. The separating surface depends only on a subset of the original data known as a set of support vectors. A support vector machine constructs a hyper plane or set of hyper planes in a high dimensional space, which can be used for classification. A good separation is achieved by the hyper plane that has the largest distance to the nearest training data points of any class, called functional margin. If this functional margin is large, then the generalization error of the classifier will be small. SVM models are built around a kernel function [31] that transforms the input data into an $n$-dimensional space where a hyper plane can be constructed to partition the data. The standard SVM algorithm implemented in Weka, a data mining tool that we use in our experiments, is called the sequential minimal optimization (SMO) [35]. Here, we employ SMO PolyKernel using the polynomial kernel function [36]. We also use the SMO radial basis function (RBF) kernel method [37].

#### *2.2.5 Bagging algorithm*

Bagging, which means *bootstrap aggregating*, is a "bootstrap" ensemble method that makes decisions from multiple classifiers [38]. In bagging, each bootstrap sample is obtained by randomly sampling training instances, with replacement, from the original training set. Hence, some examples may be selected repeatedly, while others may left out. The bootstrap sample and the training set have the same number of instances. The final decision in bagging is obtained by combining the decisions of the component classifiers using unweighted voting. Note that bagging enhances the classification performance mainly by minimization of the variance error [39]. Usually, this algorithm is applied to the decision tree algorithms. Thus, we employ the bagging algorithm combined with J48 classifier.

#### *2.2.6 Boosting algorithm*

Adaptive Boosting (AdaBoost) is a method for combining multiple classifiers by using a set of many weak or base classifiers in order to ameliorate the overall performance [40]. AdaBoost calls a given weak classifier repeatedly in a series of rounds. At each call, a distribution of weights is updated

indicating the importance of examples in the data set for the classification. On each round, the weights of each correctly classified example are decreased (or alternatively, the weights of each incorrectly classified example are increased), so that the new classier is forced to focus on those examples. At the end, the predictions of all weak classifiers are combined into a single prediction with weighted voting. AdaBoost is adaptive in the sense that subsequent classifiers built are tweaked in favor of those instances misclassified by previous classifiers. It is more effective than bagging at reducing both bias and variance of the basic classifiers and has good generalization properties. Also, this algorithm is less susceptible to the overfitting problem than many other learning algorithms. In this work, we use the AdaBoost.M1 algorithm [40] combined with J48 classifier.

## 3 SPIT attack detection system

### 3.1 SPIT identification criteria

In this subsection, we summarize the criteria used to identify SPITters.

*SIP address* A user agent (UA) caller has a SIP uniform resource identifier (URI). If this address appears in a blacklist database, the UA caller is considered as a SPITter and will be filtered.

*Calls messages rate* We analyze the number of calls made within a specific time period by each user. If this number exceeds a predefined threshold, then the UA can be considered as a potential SPITter.

*Constant calls duration* We analyze the duration of calls initiated by a single user. If it has a static duration, then the UA is a potential SPITter. Indeed, this caller may use an automated script in order to initiate VoIP calls.

*Constant silence length* We analyze the silence length of calls initiated by a single user. If it has a constant value, then the UA is a potential SPITter.

*Receivers' address pattern* If the receivers' addresses follow a specific pattern (e.g., alphabetical SIP URI addresses), then the call (message) is flagged as a potential SPIT. For example, the sender tries to find a valid address by sending many requests having similar URI with a unique difference in the first letter.

*Answered calls percentage* We compare the number of successful calls initiated by a caller with the number of failed ones within a predefined time period. If the percentage of successful calls is very small, then the caller is a potential SPITter.

*Errors number* If a user sends a large number of INVITE messages which are replied by the SIP proxy with a large number of error messages (e.g., 404 Not Found), then it is a sign of a potential SPIT attack.

*Size of SIP messages* By automating the process of sending messages, SPITters can employ a "bot" to solicit the same message (i.e., constant size of SIP message) to many UA subscribers. Thus, the size of SIP messages can be a good criterion to distinguish between human and "bot" UA and consequently help to detect SPIT. According to this parameter, we choose the call duration, speech duration and the silence length as criteria of SPIT identification.

In addition, SPITters can randomly generate many UA identifiers and try to communicate with them. Therefore, analyzing the number of destination users can be used to predict the behavior of each UA.

### 3.2 SPIT detection Process

According to the anti-SPIT mechanisms mentioned in the Sect. 2, we propose a SPIT detection model using behavior-based approach. Our system is consisted of two parts: the collection of UA's identification criteria and the classification of each UA as SPITter or legitimate.

Based on the identification criteria cited in Sect. 3.1, we select nine criteria for SPIT detection that are depicted in Table 2. SPITters can randomly generate many UA identifiers and try to communicate with them. Therefore, analyzing the number of destination users can be used to predict the behavior of each UA. These criteria have the ability to predict the legitimacy of calls. In fact, SPITter can try to change the message content in order to hide their menacing activities but without giving importance to these criteria.

Our proposed system collects the network traces of signaling and voice activities for identification criteria according to the diagram illustrated in Fig. 1.

For each initiated call, the SIP address of the caller and the addresses of the callees are collected. In fact, when the caller and callee addresses are valid, the proposed system checks if the address of the caller belongs to the blacklist of proxy server. If it is true and the caller address is not in the callee

**Table 2** Identification criteria of SPITter

| Criteria | Abbreviation |
|---|---|
| Percentage of answered calls | %AC |
| Percentage of rejected calls | %RC |
| Percentage of calls with invalid callees (NOT FOUND response) | %IC |
| %AC + %RC + %IC = 100 % | |
| Number of calls attempts | NCA |
| Variance of call duration | VCD |
| Variance of inter-call duration | VICD |
| Number of destination users | NDU |
| Variance of silence length | VSL |
| Variance of talk duration | VTD |

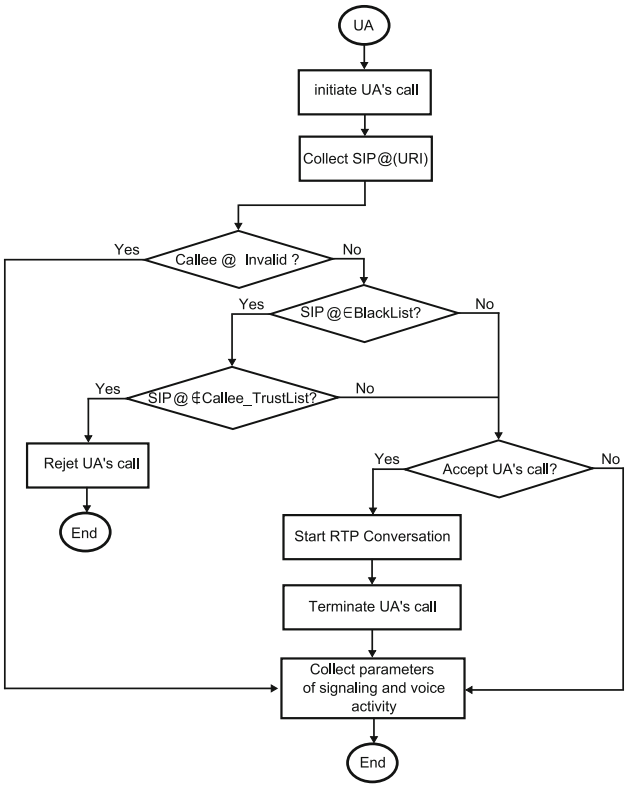


**Fig. 1** Collection of the network traces (signaling and voice activities)

**Table 3** Parameters of the network traces (signaling and voice activities)

| Parameter | Abbreviation |
|---|---|
| ID call | Id_Call |
| Callee address | @Callee |
| Caller address | @Caller |
| Success call | $SC\epsilon$ {0, 1} |
| Rejected call | $RC\epsilon$ {0, 1} |
| Invalid callee | $IC\epsilon$ {0, 1} |
| Call initiated | CI |
| Call end time | CET |
| Call duration | CD = CET − CI |
| Silence length | SL |
| Talk duration | TD |

trustlist, the call is rejected. Otherwise (i.e., SIP address is not in blacklist or SIP address in the callee trustlist), an RTP conversation is done. Thus, after the end of the conversation, the collection of the network traces of signaling and voice activities triggers. These traces are then processed for the SPIT detection. Table 3 shows the parameters of network traces that we must retrieve.

As communication data flow is potentially infinite, it will be impossible to store it completely. Besides, extracting properties for data set are more valuable when data is time correlated. Therefore, we apply the concept of the "sliding

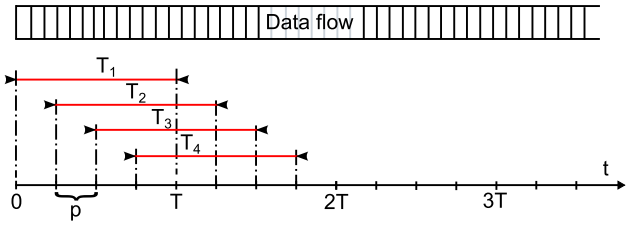


**Fig. 2** Sliding window

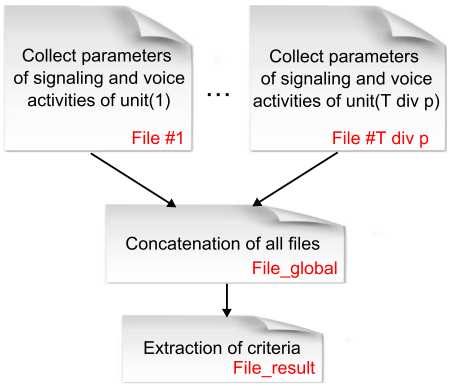


**Fig. 3** Extraction of criteria to identify SPITter

window." The role of this window is to maintain a flow of data as a finite set. The operating principle of this window is illustrated in Fig. 2.

The window is divided into p units ($p = 4$ in the Fig. 2). The first window $T_1$ is formed by four p units. Subsequently, each window $T_k$ ($k > 1$) is formed by inserting a new unit to its previous window and deleting the oldest one [41].

We explain our SPIT detection system by Algorithm 1. At the beginning, we create $T$ *div* p files [i.e., file(i) where $i = 1 \ldots T$ *div* p] to save the network traces of signaling and voice activities for each unit(i). According to the principal of "sliding window," our proposed system collects the relative network traces for each unit and concatenates all files in a global file. Thereafter, the system extracts the SPIT identification criteria from this file. The extraction process is illustrated in Fig. 3.

Finally, the classification phase is triggered, and the blacklist is updated. It is important to deduct a thread to continue both the collection of network traces for the next window and the operations of classification.

After the extraction of criteria, our proposed system employs a supervised classification given that our classes are previously known (labeled training data). We illustrate the mechanism of this classification in Fig. 4.

Indeed, this mechanism typically employs two phases of processing: The first one concerns learning. It involves building a classifier from a training database. The second phase is to predict the class of each UA based on the classifier already built.

In the following, we describe the simulation platform.

**Algorithm 1** SPIT detection system

**Input** T, p, parameters_signaling_voice_activities, file($i = 1..T\ div\ p$), file_BlackList
//Initialize the index i to collect the relative traces
//of the unit(1)
$i \leftarrow 1$;
**while** *(current_time* $\leq$ *time_of_simulation)* **do**
  //Open file relative to unit(i)
  Open(file(i));
  //Collect parameters of signaling and voice activities
  Write(parameters_signaling_voice_activities,file(i));
  **if** *(current_time mod p == 0)* **then**
    // Close the file relative to unit(i)
    Close(file(i));
    **if** *(current_time* $<$ *T)* **then**
      //Update the index i of the file to insert the new
      //data items (Sliding window)
      $i \leftarrow i + 1$;
    **else**
      //Concatenation of all files
      file_global $\leftarrow$ Concatenation(file($i = 1..T\ div\ p$));
      //Update the index i of the file to insert the new
      //data items (Sliding window)
      $i \leftarrow (current\_time\ div\ p)\ mod\ (T\ div\ p) + 1$;
      //Empty the file(i)
      Empty(file(i));
      // Extraction of criteria for identifying SPITter
      file_resultat $\leftarrow$ extraction_features(file_global);
      //Classification
      file_classification $\leftarrow$ classification(file_result);
      //Update of the BlackList
      Update(file_classification,file_BlackList);
      //Empty the file_global, the file_result
      //and the file_classification
      Empty(file_global);
      Empty(file_result);
      Empty(file_classification);
    **end**
  **end**
**end**

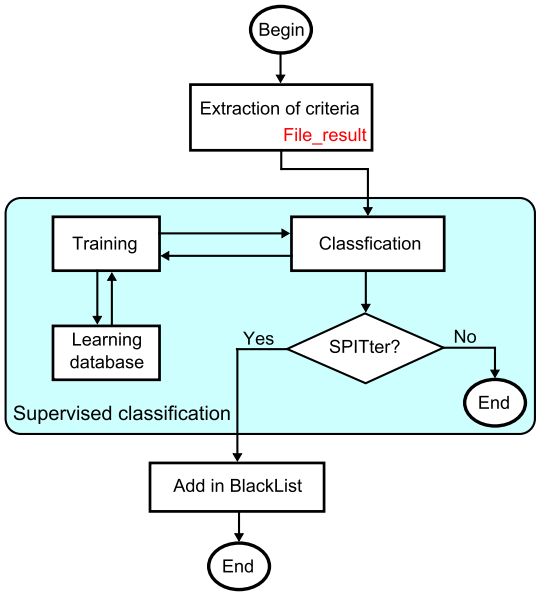


**Fig. 4** Supervised classification

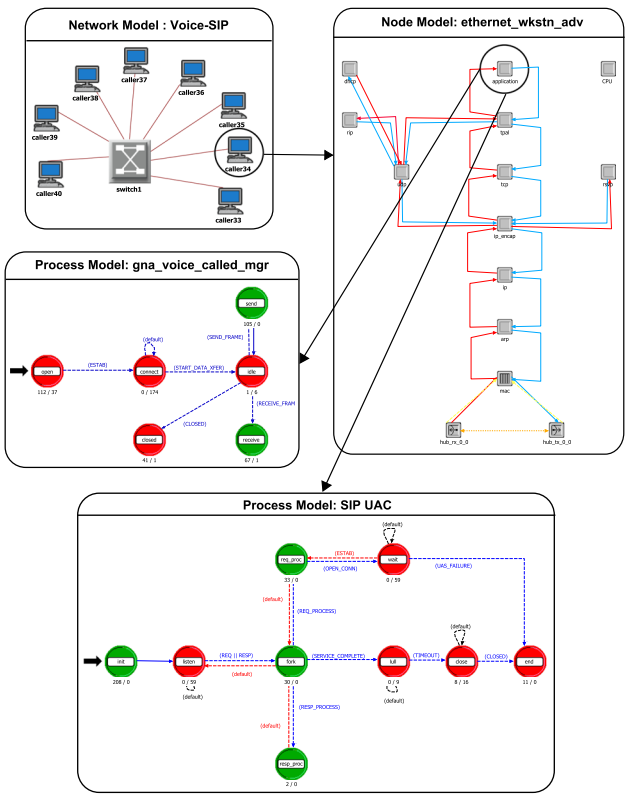


**Fig. 5** Hierarchical model of a VoIP network

## 4 Simulation platform

In order to evaluate the proposed system, we use OPNET simulator (Modeler environment 14.5) [42] to extract network traces (signaling and voice activities) and Weka [43] as a data mining software for classification.

### 4.1 Collection of SPIT identification criteria under OPNET

OPNET uses a hierarchical model describing, in a precise way, topologies and exchanged flows in a communication system.

We adapt the existing SIP model to extract SPIT identification criteria. As shown in Fig. 5, a hierarchical model has three description levels:

- Network model: It represents the physical topology including a set of nodes and interconnectors.
- Node model: It defines the behavior of the node and controls the flow of data among various functional elements within the node.

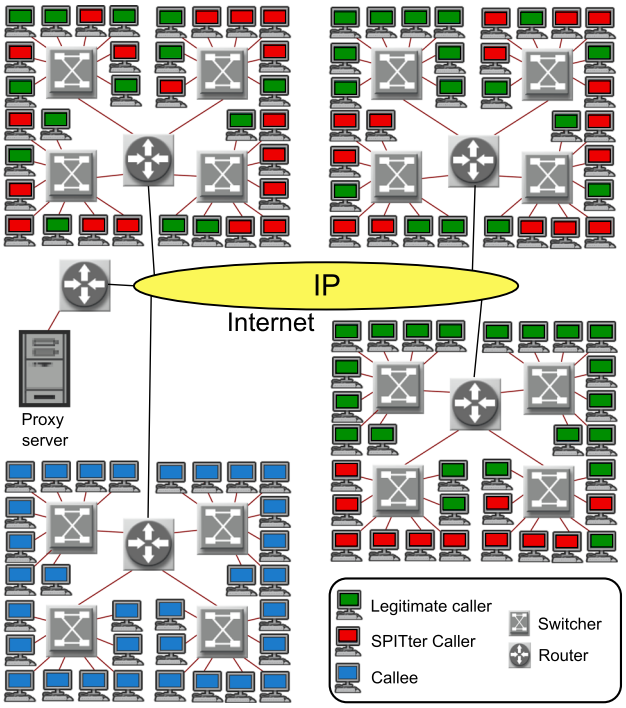


**Fig. 6** VoIP network topology

– Process model: It is represented by finite state machines (FSMs). It is created with icons and lines that represent states and transitions between states, respectively. Here, we can perform operations in each state or transition by means of embedded C or C++ code blocks.

At the OPNET network model, we establish a VoIP network topology illustrated in Fig. 6.

It is composed of stations (96 callers and 32 callees), switches, routers and proxy server. Here, there are 52 legitimate and 44 SPITter callers. Stations are geographically distributed on four regions. Communication between the UAs is ensured via a proxy server that handles the routing of any SIP session. The configuration of these stations is described in the following paragraph.

At the process model, our interventions occur in the exploitation of existing processes such as SIP_UAC process model and gna_voice_called_mgr to generate network traces including signaling and voice activities.

Note that each station has the parameters presented in Table 4.

In order to discriminate between the legitimate and SPITter UAs, we set the values of NCR, CD, ICD, SL and TD. These values should be exponentially or constantly distributed. Thereafter, we collect the parameters of network traces for all calls presented in Table 3 and hence extract the criteria of SPIT identification mentioned in Table 2.

**Table 4** Configuration of station parameters

| Parameter | Value |
|---|---|
| Client address: CA | Unique |
| Destination preferences: DP | List |
| Proxy server address: PSA | @proxy |
| Number of call repetition: NCR | NCR $\in \{cst_{NCR}, \exp(\alpha_{NCR})\}$ |
| Call duration (s): CD | CD $\in \{cst_{CD}, \exp(\alpha_{CD})\}$ |
| Inter-repetition call duration (s): ICD | ICD $\in \{cst_{ICD}, \exp(\alpha_{ICD})\}$ |
| Silence length (s): SL | SL $\in \{cst_{SL}, \exp(\alpha_{SL})\}$ |
| Talk duration (s): TD | TD$\in \{cst_{TD}, \exp(\alpha_{TD})\}$ |

### 4.2 Classification

The building of the training database is done by setting the thresholds. In order to discriminate each UA (i.e., legitimate or SPITter), we associate the thresholds $thr_j$ to the criteria $cr_j$ $(j = 1 \ldots 9)$, respectively. These criteria $(cr_j, j = 1 \ldots 9)$ correspond to %AC, VCD, VICD, VSL, VTD, %RC, %IC, NDU and NCA, respectively. We suppose that

$$\forall j \in \{1, \ldots, 5\},\ f\left(cr_j\right) = \begin{cases} 0 & \text{if } cr_j < thr_j, \\ 1 & \text{if } cr_j \geq thr_j; \end{cases} \tag{1}$$

and

$$\forall j \in \{6, \ldots, 9\},\ g\left(cr_j\right) = \begin{cases} 0 & \text{if } cr_j > thr_j, \\ 1 & \text{if } cr_j \leq thr_j. \end{cases} \tag{2}$$

Each UA is characterized by a set of functions $(S)$ given by

$$\begin{aligned} \mathcal{S} = (&f(cr_1),\ f(cr_2),\ f(cr_3),\ f(cr_4),\ f(cr_5), \\ &g(cr_6),\ g(cr_7),\ g(cr_8),\ g(cr_9)). \end{aligned} \tag{3}$$

We distinguish two classes of UA: the first is a legitimate $(\mathcal{LC})$ given by

$$\begin{aligned} \mathcal{LC} = \{&\mathcal{S}/f(cr_1) = f(cr_2) = f(cr_3) = f(cr_4) \\ &= f(cr_5) = g(cr_6) = g(cr_7) = g(cr_8) \\ &= g(cr_9) = 1\} \end{aligned} \tag{4}$$

and the second is a SPITter $(\mathcal{SC})$ given by

$$\mathcal{SC} = \overline{\mathcal{LC}} \tag{5}$$

### 4.3 Metrics for performance evaluation of UA classifier

The efficiency indicators used in the evaluation of classification applications are numerous. However, most of these indicators are built from the contingency Table 5. Cells abbreviations are as follows:

– TN (True Negative): legitimate UA classified correctly,
– FN (False Negative): SPITter UA classified as legitimate,
– TP (True Positive): SPITter UA classified correctly,
– FP (False Positive): legitimate UA classified as SPITter.

From Table 5, we can extract several relations that define efficiency criteria. We summarize some of these criteria in Table 6.

Indicators derived from a contingency table (Table 6) have the disadvantage of being specific to a threshold value and does not inform us about the effectiveness of the classifier to other threshold values. An effective way for the evaluation of detection performance is to plot the ROC curve [44]. The interpretation of the ROC curve is shown in Fig. 7.

By computing the ROC curve area, denoted area under the curve (AUC) [45], we can compare multiple classifier. Indeed, if the AUC of a classifier is greater than another classifier, then we can say that the first one is generally more efficient than the other.

The ROC curve is also used for fixing optimum threshold value. In fact, Wright [46] justifies the use of ROC in finding a filter's optimal threshold in signal detection theory (SDT). This author presents a discussion of ROC curve on how the optimum decision criterion can be achieved. According to Saber et al. [40], based on the respective ROC curve, an optimum threshold was computed for each class, which simultaneously maximizes the count of TP and minimizes the count of FP. Graphically, optimum threshold is known as point on curve closest to (0, 1). Indeed, if we denote $S_n$ et $S_p$ sensitivity and specificity, respectively, the distance between the point (0, 1) and any point in the ROC curve is $d^2 = (1 - S_n)^2 + (1 - S_p)^2$ [38]. So, to obtain the optimal cutoff point ( relative to optimum threshold), we must calculate the distance $d$ for each observed cutoff point and locate the point where the distance is minimum (see Fig. 7). Here, we pay considerable attention to the optimal threshold in order to show further that when dmin converge to 0 (i.e., AUC converge to 1) the classifier is considered as ideal.

We notice that there are difference between the threshold of this curve and the threshold used to distinguish between legitimate and SPITter UA.

**Table 5** Contingency table

| Classifier | Reality | |
|---|---|---|
| | Legitimate | SPITter |
| Legitimate | TN | FN |
| SPITter | FP | TP |

**Table 6** Indicators defined from the contingency table

| Indicator | Definition |
|---|---|
| Rate of false positive (RFP) | $\frac{FP}{TN+FP}$ |
| Rate of false negative (RFN) | $\frac{FN}{TP+FN}$ |
| Sensitivity, power | $(1 - TFN) = \frac{TP}{TP+FN}$ |
| Specificity | $(1 - TFP) = \frac{TN}{TN+FP}$ |
| Accuracy | $\frac{TP+TN}{TP+FP+TN+FN}$ |
| Rate of global error | $\frac{FP+FN}{TP+FP+TN+FN}$ |

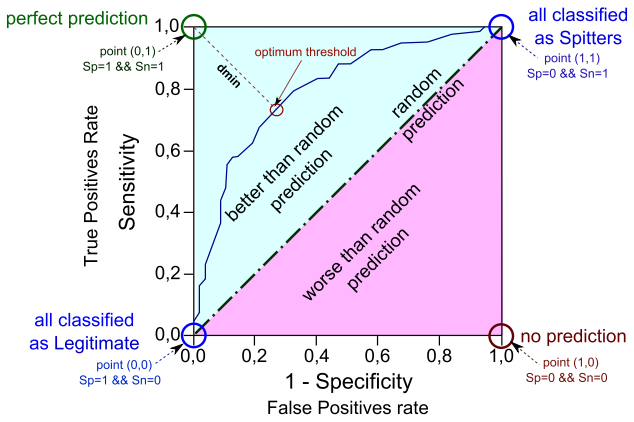


**Fig. 7** ROC curve

## 5 Simulation results

We execute our simulation during 52 min in order to evaluate the different aforementioned classification techniques. We set $T = 30$ min (duration of the sliding window) and $p = 2$ min (sliding unit). Under this configuration, we have fixed the thresholds as given by Table 7.

The thresholds of VCD, VICD, VSL and VTD are fixed through intensive simulation. In fact, when we select constantly distributed, the variances of these criteria do not exceed an upper bound. Note that the difference between the variance of exponential and constant distributions is

**Table 7** Fixed thresholds

| Criteria | Threshold | Threshold value |
|---|---|---|
| $cr_1$ | $th_1$ | 75 % |
| $cr_2$ | $th_2$ | 500 |
| $cr_3$ | $th_3$ | 500 |
| $cr_4$ | $th_4$ | 500 |
| $cr_5$ | $th_5$ | 500 |
| $cr_6$ | $th_6$ | 75 % |
| $cr_7$ | $th_7$ | 75 % |
| $cr_8$ | $th_8$ | 8 |
| $cr_9$ | $th_9$ | 10 |

**Table 8** Appearance of the 44 SPITter callers over 12 windows

| Window | Number of SPITters | Desired detection (%) |
|---|---|---|
| $T_1$ | 31 | 70.45 |
| $T_2$ | 1 | 72.72 |
| $T_3$ | 0 | 72.72 |
| $T_4$ | 1 | 75.00 |
| $T_5$ | 1 | 77.27 |
| $T_6$ | 0 | 77.27 |
| $T_7$ | 1 | 79.54 |
| $T_8$ | 2 | 84.09 |
| $T_9$ | 2 | 88.63 |
| $T_{10}$ | 2 | 93.18 |
| $T_{11}$ | 1 | 95.45 |
| $T_{12}$ | 2 | 100 |
| Total | 44 | |

remarkable. Thus, we choose the largest value of this latter as a threshold. For other criteria, we choose the thresholds appropriate to the sliding window period. For instance, it is reasonable that the behavior of UA becomes abnormal when the NCA exceeds 10 new UAs within 30 min. It is worth noting that the classification phase is triggered at each window and the blacklist is updated immediately. Hence, the UA, which identify as a SPITter, will be stopped immediately, and he can only communicate with UA that appear in the callee trustlist. Here, we can say that our proposed system can maintain good performance even when we change the values of thresholds. In fact, the choice of thresholds will not influence the detection of SPITters by machine learning algorithms. Thus, a security administrator can vary the appropriate choice of thresholds according to a pre-knowledge of VoIP network without losing detection performance. It is worth noting that our learning database contains 512 patterns (179 legitimate and 333 SPITter UAs). All simulations are generated based on this database. For the test phase, we generate 96 UAs where 44 become SPITters as seen in Fig. 6. The appearance of these 44 SPITter callers over the 12 windows is illustrated in Table 8. In Weka knowledge flow, we use the "TestSetMaker" as a test option since it allows for classifying all UAs. Note that the classifier with an ideal detection is one where its curve of true detection is the same as the curve of desired detection.

In our experimental results, we plot the ROC curves to evaluate the performance of the machine learning techniques relative to our framework.

Figure 8 shows the ROC curves from the Naive Bayes and the BayesNet methods.

By comparing the ROC curves of these two methods, we can conclude that the SPIT detection performance using the BayesNet method is much better than that of Naive Bayes.

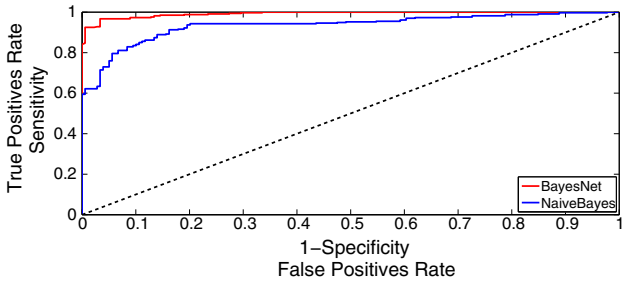


**Fig. 8** ROC curves of Naive Bayes and BayesNet methods

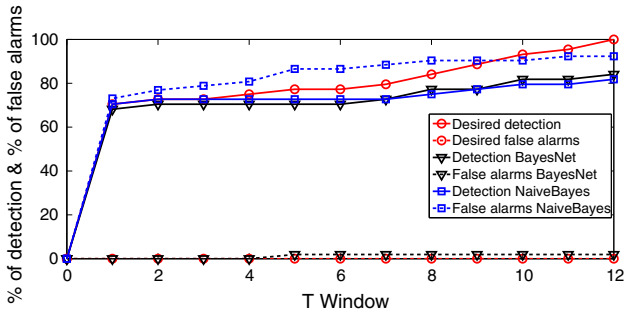


**Fig. 9** Curves of true detection and false alarms for BayesNet and Naive Bayes methods

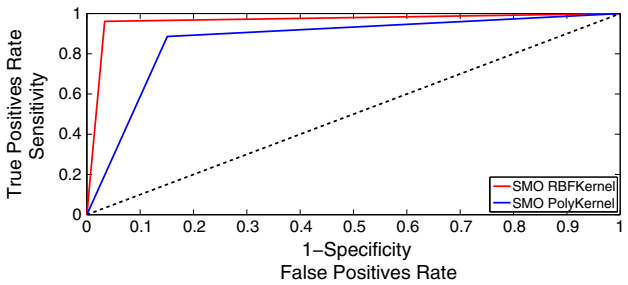


**Fig. 10** ROC curve analysis of SMO PolyKernel and RBFKernel methods

Indeed, the ROC curve of BayesNet method is above that of the Naive Bayes, and therefore, its AUC is larger.

To confirm these results, we schematize in Fig. 9 the rates of true detection and false alarms for the two classification methods: Naive Bayes and the BayesNet. We notice that Naive Bayes method produces a lot of false alarms. Regarding the BayesNet method, the percentage of desired detection (i.e., ideal detection) is not achieved although the percentage of false alarms is quite close to the desired case (i.e., ideal case).

Figure 10 depicts the ROC curves issued from the SMO PolyKernel and RBFKernel methods. We can conclude that SMO RBFKernel method outperforms SMO PolyKernel since it has a larger AUC.

Figure 11 shows the curves of true detection percentage and false alarms percentage of SMO PolyKernel and RBFKernel methods. These curves confirm that SMO RBFKernel

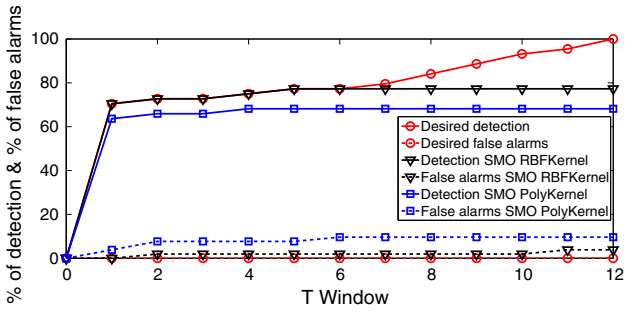


**Fig. 11** Curves of true detection and false alarms for SMO RBFKernel and SMO PolyKernel methods

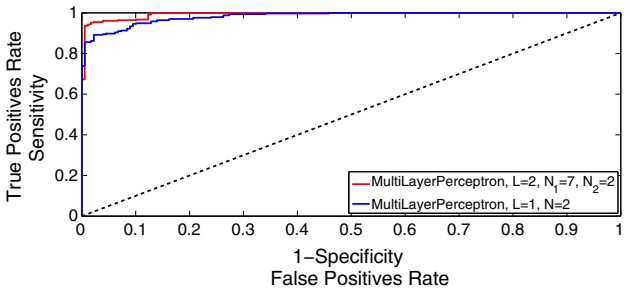


**Fig. 12** ROC curve analysis of MultiLayerPerceptron methods with two and three layers

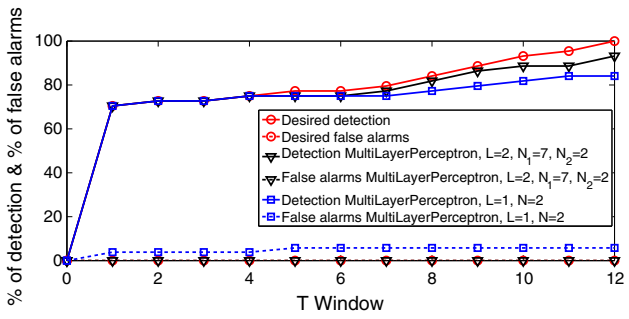


**Fig. 13** Curves of true detection and false alarms for MultiLayerPerceptron methods with two and three layers

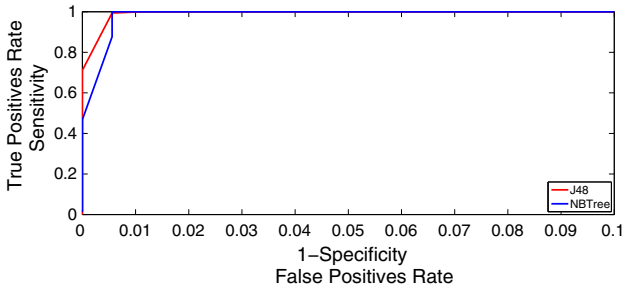


**Fig. 14** ROC curve analysis of J48 and NBTree methods

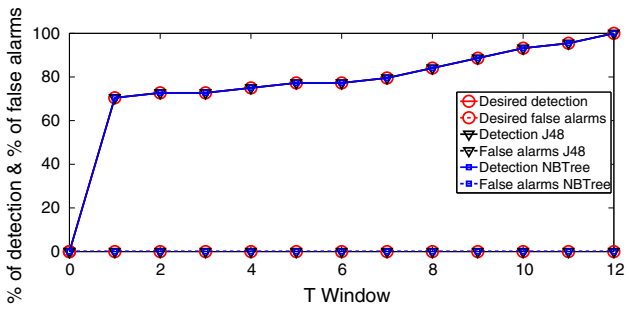


**Fig. 15** Curves of true detection and false alarms for J48 and NBTree methods

classifier exceeds SMO PolyKernel since it has a lower false alarms rate and a higher true detection rate.

We also compare the method of three-layer perceptron (seven nodes at the first hidden layer [$(N_1 = 7)$ and two nodes at the second hidden layer $(N_2 = 2)$] with the two-layer perceptron [(two nodes at the unique hidden layer $(N_1 = 2)$]. Through Fig. 12, it is clear that the AUC of the three-layer perceptron is larger than that of two-layer perceptron. Therefore, the first classifier is the most efficient.

Figure 13 presents the rates of true detection and false alarms for the two classification methods: three-layer perceptron and two-layer perceptron. These curves confirm the superiority of three-layer perceptron, mainly because of a less generation of false alarms. This characteristic is also maintained when comparing three-layer perceptron with the Bayesian and SVM methods.

Figure 14 shows the good performances of the J48 and NBTree methods. Indeed, these classifiers have the greatest AUC compared to the above classification methods.

Figure 15 shows the curves of true detection percentage and false alarms percentage of J48 and NBTree methods. These curves confirm that these two methods are efficient in our testing environment. Indeed, they provide an ideal true detection rate without any false alarm.

Comparing the ROC curves of AdaBoostM1, Bagging and J48 methods will reveal that the AdaBoostM1 classifier is more efficient as it offers better AUC $(AUC = 1)$ as shown in Fig. 16.

It is clear in Fig. 17 that the AdaBoostM1 and bagging methods can perfectly distinguish between legitimate and SPITter UAs since they successfully detected SPITters and did not produce any false alarms.

Through Table 9, we illustrate the duration of the training phase for the various classification methods. This period corresponds to time needed to build the classification model. We can notice that the time duration taken by the AdaBoostM1 method is acceptable, but it is too long for the MultilayerPerceptron method.

Table 10 summarizes the AUC of each classification method.

The first interesting insight is that AdaBoostM1 method gives the best detection rate since it has the largest $AUC = 1$.

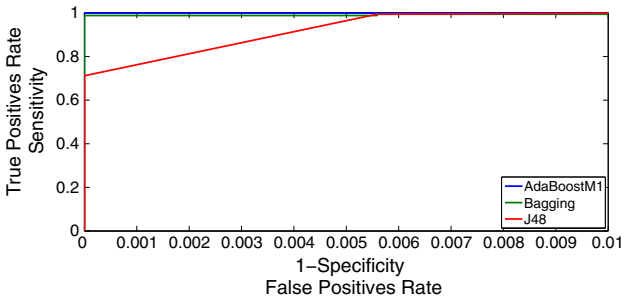


**Fig. 16** ROC curve analysis of AdaBoostM1, Bagging and J48 methods

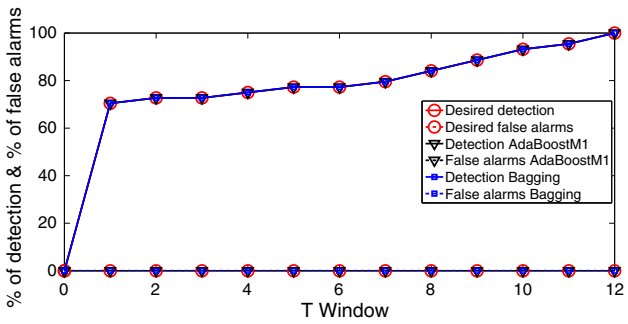


**Fig. 17** Curves of true detection and false alarms for AdaBoostM1 and bagging methods

**Table 9** Time model building

| Classifier | Time taken to build model (s) |
|---|---|
| Naive Bayes | 0.007 |
| BayesNet | 0.016 |
| J48 | 0.021 |
| Bagging | 0.090 |
| AdaBoostM1 | 0.130 |
| SMO (RBFKernel) | 0.277 |
| NBTree | 0.858 |
| MultilayerPerceptron $L = 2$ | 2.248 |

**Table 10** AUC of each classification method

| Classifier | AUC |
|---|---|
| AdaBoostM1 | 1 |
| Bagging | 0.9999 |
| J48 | 0.9992 |
| NBTree | 0.9982 |
| BayesNet | 0.9920 |
| MultiLayerPerceptron $L = 2$ | 0.9915 |
| MultiLayerPerceptron $L = 1$ | 0.9827 |
| SMO RBFKernel | 0.9637 |
| Naive Bayes | 0.9346 |
| SMO PolyKernel | 0.8675 |

It is closely followed by Bagging $= 0.9999$. On the contrary, Naive Bayes and SMO PolyKernel methods provide relatively low detection rates with $AUC = 0.9346$ and $AUC = 0.8675$, respectively.

In conclusion, we can say that the AdaBoostM1 method provides a compromise between detection performance and convergence speed in our detection system.

## 6 Conclusion

This work aims to develop an anti-SPIT mechanism using a behavior-based approach. To achieve this purpose, we deployed a VoIP network topology using OPNET environment. Through the simulation, we collected network traces of signaling and voice activities. These traces were useful to extract nine identification criteria of SPIT attacks on the one hand and to implement a sliding window mechanism on the other hand. Under our experimental assumptions and using the data mining tool Weka, we carried out a comparative study of ten supervised classification methods (Naive Bayes, BayesNet, SMO RBFKernel, SMO PolyKernel, MultiLayerPerceptron with two and three layers, NBTree, J48, Bagging and AdaBoostM1). We used the ROC curves and the learning period time to compare between the various classifiers. Our study showed the superiority of the AdaBoostM1 classifier. Another finding from our study was the presence of false alarms and false negatives (non detected attacks) with the different classifiers. In a future work, we plan to conduct a stateful multi-protocol analysis to have more identification criteria.

## References

1. Kolan, P., Dantu, R.: Socio-technical defense against voice spamming. ACM Trans. Auton. Adapt. Syst. (TAAS) **2**(1), 2 (2007)
2. Shin, D., Ahn, J., Shim, C.: Progressive multi gray-leveling: a voice spam protection algorithm. IEEE Netw. **20**(5), 18–24 (2006)
3. Yan, H., Sripanidkulchai, K., Zhang, H., Shae, Z.Y., Saha, D.: Incorporating active fingerprinting into spit prevention systems. In: Third Annual Security Workshop (VSW), 2006
4. Schlegel, R., Niccolini, S., Tartarelli, S., Brunner, M.: Spit prevention framework. In: Proceedings of IEEE GLOBECOM, pp. 1–6, Dec. 2006
5. Nassar, M., Dabbebi, O., Badonnel, R., Festor, O.: Risk management in voip infrastructure using support vector machines. In: Proceedings of International Conference on Network and Service Management (CNSM), pp. 48–55, Oct. 2010
6. Nassar, M., State, R., Festor, O.: Monitoring sip traffic using support vector machines. In: Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID), pp. 311–330, 2008
7. Wu, Y.S., Bagchi, S., Singh, N., Wita, R.: Spam detection in voice-over-ip calls through semi-supervised clustering. In: Proceedings of Dependable Systems Networks, pp. 307–316, 2009

8. Keromytis, A.D.: A comprehensive survey of voice over ip security research. IEEE Commun. Surv. Tutor. **14**(2), 514–537 (2012)
9. Bai, Y., Su, X., Bhargava, B.: Adaptive voice spam control with user behavior analysis. In: Proceedings of IEEE International Conference on High Performance Computing and Communications (HPCC), pp. 354–361, Jun. 2009
10. Rosenberg, J., Jennnings, C.: The session initiation protocol and spam. In: IETF Draft, Feb. 2007
11. Hasen, M., Hansen, M., Moller, J., Rohwer, T., Tolkmit, C., Waack, H.: Developing a legally compliant reachability management system as a countermeasure against spit. In: VoIP Security Workshop, Berlin, Jun. 2006
12. Dantu, R., Kolan, P.: Detecting spam in voip networks. In: Proceedings of the Steps to Reducing Unwanted Traffic On the Internet Workshop, Cambridge, pp. 31–37, Jul. 2005
13. Radermacher, T.A.: Spam Prevention in Voice over IP Networks. Master's thesis, University of Slazburg, Nov. 2005
14. Mathieu, B., Niccolini, S., Sisalem, D.: Sdrs: a voice-over-ip spam detection and reaction system. IEEE Secur. Priv. **6**, 52–59 (2008)
15. Levine, B.N., Shields, C., Margolin, N.B.: A survey of solutions to the sybil attack. In: Technical Report 2006–052, University of Massachusetts Amherst, MA, Oct. 2006
16. Rebahi, Y., Sisalem, D.: Sip service providers and the spam problem. In: Workshop on Securing Voice over IP, Washington, DC, Jun. 2005
17. Patankar, P., Nam, G., Kesidisand, G., Das, C.: Exploring anti-spam models in large scale voip systems. In: Proceedings of International Conference on Distributed Computing Systems, China, Jun. 2008
18. Balasubramaniyan, V.A., Ahamad, M., Park, H.: Callrank: Combating spit using call duration, social networks and global reputation. In: Proceedings of Conference on Email and Anti-Spam, USA, Aug. 2007
19. Soupionis, Y., Gritzalis, D.: Aspf: Adaptive anti-spit policy-based framework. In: Proceedings of International Conference on Availability, Reliability and Security (ARES), Aug. 2011
20. Johansen, A.J.: Improvement of Spit Prevention Technique Based on Turing Test. PhD thesis, Mahanakorn University of Technology, 2010
21. Quittek, J., Niccolini, S., Tartarelli, S., Stiemerling, M., Brunner, M., Ewald, T.: Detecting spit calls by checking human communication patterns. In: Proceedings of IEEE International Conference on Communications (ICC), pp. 1979–1984, 2007
22. Kusumoto, T., Chen, E.Y., Itoh, M.: Using call patterns to detect unwanted communication callers. In: Proceedings of International Symposium on Applications and the Internet (SAINT), 2009
23. Jabeur Ben Chikha, R., Abbes, T., Bouhoula, A.: A spit detection algorithm based on user's call behavior. In: 21st International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Sept. 2013
24. Datar, M., Gionis, A., Indyk, P., Motwani, R.: Maintaining stream statistics over sliding windows. In: Proceedings of 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 635–644, Jan. 2002
25. Golab, L., Garg, S., Ozsu, M.T.: On indexing sliding windows over on-line data streams. In: Proceedings of International Conference on Extending Database Technology (EDBT), pp. 712–729, 2004
26. Bouckaert, R.: Bayesian Network Classifiers in Weka. Technical Report, Department of Computer Science, Waikato University, Hamilton, 2005
27. John, G.H., Langley, P.: Estimating continuous distributions in bayesian classifiers. In: Proceedings of 11th Conference on Uncertainty in Artificial Intelligence, pp. 338–345. Morgan Kaufman, San Mateo (1995)
28. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
29. Kohavi, R., Quinlan, J.R.: Decision-tree discovery. In: Klosgen, W., Zytkow, J.M. (eds.) Handbook of Data Mining and Knowledge Discovery, Chap. 16.1.3. pp. 267–276. Oxford University Press (2002)
30. Kohavi, R.: Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In: Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD), 1996
31. Vapnik, V.N.: The Nature of Statistical Learning Theory, 2nd edn. Springer, New York (1999)
32. Cristianini, N., Shawe-Taylor, N.J.: An introduction to support vector machines. Cambridge University Press, Cambridge (2000)
33. Vapnik, V.N.: Statistical Learning Theory. Wiley, New York (1998)
34. Joachims, T.: Making large-scale svm learning practical. In: Schokopf, B., et al. (eds.) Advances in Kernel Methods-Support Vector Learning. MIT Press, Cambridge (1999)
35. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. In: Schokopf, B., et al. (eds.) Advances in Kernel Methods: Support Vector Machines. MIT Press, Cambridge (1998)
36. Scholkopf, B., Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond. The MIT Press, Cambridge (2002)
37. Scholkopf, B., Kah-Kay, S., Burges, C., Girosi, F., Niyogi, P., Poggio, T., Vapnik, V.: Comparing support vector machines with gaussian kernels to radial basis function classifiers. In: Proceedings of Signal Processing, pp. 2758–2765, 1997
38. Kumar, R., Indrayan, A.: Receiver operating characteristic (roc) curve for medical researchers. Indian Pediatr. **48**(4), 277–287 (2011)
39. Breiman, L.: Arcing classifiers. Ann. Stat. **26**(3), 801–849 (1998)
40. Saber, E., Tekalp, A.M., Eschbach, R., Knox, K.: Automatic image annotation using adaptive color classification. Graphical Models Image Process. **58**, 115–126 (1996)
41. Nori, F., Deypir, M., Sadreddini, M.H.: A sliding window based algorithm for frequent closed itemset mining over data streams. J. Syst. Softw. **86**(3), 615–623 (2013)
42. OPNET Technologies Inc. 2012. URL: http://www.opnet.com/
43. Weka, Machine Learning Group at the University of Waikato. URL: http://www.cs.waikato.ac.nz/ml/weka/
44. Fawcett, T.: An introduction to roc analysis. Pattern Recogn. Lett. **27**, 861–874 (2006)
45. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recogn. **30**, 1145–1159 (1997)
46. Wright, D.B.: Receiver operating characteristics curves. Encycl. Stat. Behav. Sci. **4**, 1718–1721 (2005)