Tunneling Activities Detection Using Machine Learning Techniques

Fabien Allard[1], Renaud Dubois[1], Paul Gompel[2] and Mathieu Morel[3]
[1]Thales Communications
160 Boulevard de Valmy – BP 82
92704 Colombes Cedex
FRANCE
firstname.lastname@fr.thalesgroup.com
[2]pgompel@gmail.com
[3]mathieu.c.morel@gmail.com

ABSTRACT

Tunnel establishment, like HTTPS tunnel or related ones, between a computer protected by a security gateway and a remote server located outside the protected network is the most effective way to bypass the network security policy. Indeed, a permitted protocol can be used to embed a forbidden one until the remote server. Therefore, if the resulting information flow is ciphered, security standard tools such as ALG[1], firewalls, IDS[2], do not detect this violation. In this paper, we describe a statistical analysis of ciphered flows that allows detection of the carried inner protocol. Regarding the deployed security policy, this technology could be added in security tools to detect forbidden protocols usages. In the defence domain, this technology could help preventing information leaks through side channels. At the end of this article, we present a tunnel detection tool architecture and the results obtained with our approach on a public database containing real data flows.

INTRODUCTION

Controlling flows going through network boundaries is a key point of information systems security. The filtering of these flows and the verification of their conformance to the network security policy is done in security gateways by ALG and firewalls. In particular, these tools enforce the restrictions on forbidden protocols over the network. This task is achieved by packets filtering techniques and deep inspection of carried payloads.

Nonetheless, firewalls and ALG may become completely ineffective in two cases: if a permitted protocol is used to embed a forbidden one or if the flow is ciphered. This enables a legitimate or malicious user to infringe the security policy of an information network, using covert application-layer tunnels to bypass security gateways (*cf.* fig. 1).

Tunnelling tools such as `HTTPHost` [1] or `STunnel` [2] are easily available on the Internet, and may be used by a legitimate user to establish a forbidden connection with an external Internet server. These connections consist in a protocol usually filtered by the gateway (*e.g.* ICQ, FTP, SSH, Skype, Gnutella, BitTorrent, etc.) embedded in an HTTP or HTTPS connection. The resulting data exchange is not controlled by the security

[1] Application Level Gateways
[2] Intrusion Detection System

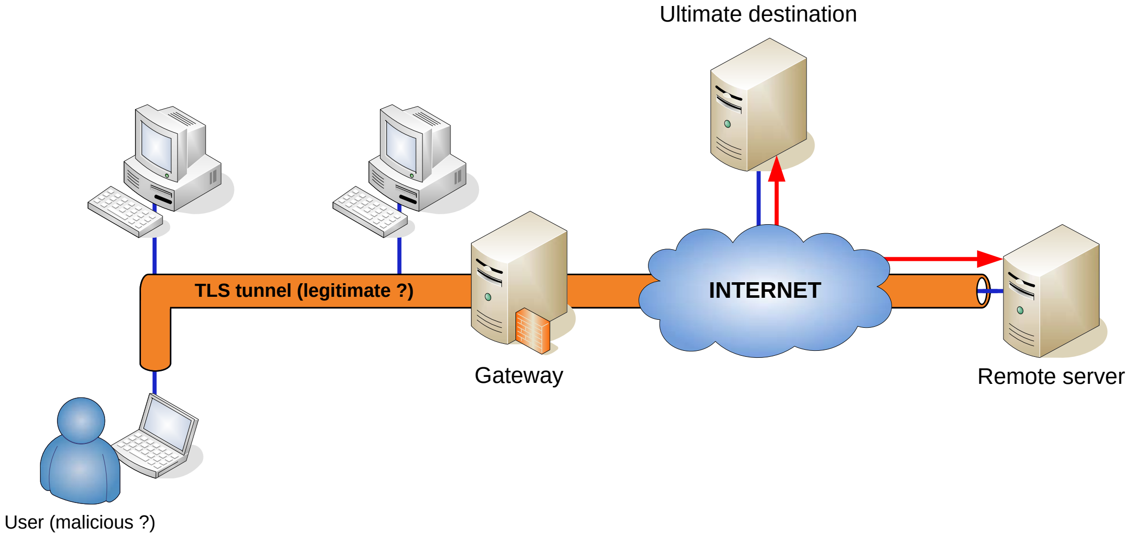

Figure 1: High level scheme of a TLS tunnel

gateway and may lead to critical information leaks or malware intrusions. For example, an invited participant to a meeting on a military vessel may use a hidden tunnel to leak out classified information via a VoIP protocol. Moreover, similar hidden tunnels are used by attackers on the Internet to communicate with local hosts that have previously been infected by a backdoor.

In this paper, we propose a solution to this problem based on machine learning techniques. Our system relies on a statistical analysis of ciphered flows enabling identification of the carried inner protocol, and therefore, detection of tunnelling activities. This solution consists in computing features for each flow and comparing these parameters to a statistical model previously built. The parameters used are derived from the size and the inter-arrival delays of the packets in the flow.

1.0 RELATED WORK

Many flow level classifiers have been presented in former works and applied to protocol identification [3, 4, 5, 6]. These studies use different parameters and machine learning techniques (Bayesian methods, support vector machine, etc.) to classify the flows into several categories (SSH, HTTP, P2P, GAMES, etc.), with promising results. However, none of these studies specifically address the security issues. Therefore, they use parameters easily tampered with by an attacker, such as port numbers or TCP flags.

To our knowledge, the methods presented in [7] and [8] are the only ones that share our goal to classify encrypted or encapsulated traffic. Nonetheless, both of these works use only the first packets of a connection to classify the entire flow. Thus, by simulating a legitimate flow using only the first packets, an attacker can easily bypass these systems.

Considering the security approach specifically, *i.e.* tunnels detection, we describe a classification method based on a decision trees forest. This method leads to better results than other machine learning algorithms. A study dealing with the impact of TLS encapsulation on flows features used for classification is also presented. Then, we present a tunnel detection tool architecture and the classification results obtained with our approach. Finally, we propose a means to decrease the false positive rate.

2.0 MACHINE LEARNING TECHNIQUES APPLIED TO TUNNEL DETECTION

Many different machine learning tools have been applied to the flow classification problem. A machine learning algorithm is used to classify a vector among several pre-determined classes. It consists in two phases :

- a learning phase, taking as input a set of vectors for each class and returning a classifying model. During this phase, the class of each vector is known ;
- a challenge phase taking as input a set of vectors, each belonging to a hidden class, the model and returning the class of each vector.

In our case, the classes are the protocols (HTTP, etc.), and the vectors are the flows (TCP, etc.) over the gateway. However, related studies were conducted on different databases, with different parameters, and results cannot be compared from one paper to another. An interesting qualitative survey of several methods is presented in [6], but no quantitative comparison is carried out.

In order to determine the most effective algorithm and the best parameters to use for classification, we conducted several experiments on a public database described in [9] and [10]. This database is composed of more than 20,000 flows captured on a real network. The distribution of the database flows by traffic classes are presented in table 1.

HTTP	Mail (POP, SMTP, IMAP, . . .)	FTP	Attack	Peer to Peer	Multimedia (WM player, Real player, . . .)	Services (X11, DNS NTP, . . .)	Interactive (SSH, Telnet)
5707	**3519**	**3107**	**1822**	**5717**	**649**	**2150**	**283**

Table 1: Distribution of the database flows by traffic classes

First, we selected the parameters that will be used to build statistical models. In order to classify the ciphered or encapsulated flows, these parameters must not be related to the packets payload. We thus kept only the parameters calculated from the sizes of exchanged packets and the inter-packets delays. In order to select the most discriminating ones, a correlation based feature selection with BestFirst search was applied, as decribed in [11]. A subset of 10 parameters was determined by this means :

- the number of transmitted packets, client to server direction ;
- the number of transmitted bytes, client to server direction ;
- the IP packets mean size, client to server direction ;
- the IP packets maximum size, client to server direction ;
- the minimum inter-arrival delay between two IP packets, client to server direction ;
- the maximum inter-arrival delay between two IP packets, client to server direction ;
- the number of transmitted bytes, server to client direction ;
- the maximum IP packets size, server to client direction ;
- the variance of the IP packets size, server to client direction ;
- the ’number of uploaded bytes / total number of exchanged bytes’ ratio.

Afterwards, we applied six different machine learning algorithms to the database, using a cross-correlation method to classify the entire database. These methods are: Support Vector Machine (SVM), Gaussian Mixture Model (GMM), K-Means, naïve Bayes method, C4.5 decision tree and RandomForest (a forest of random decision trees). For each algorithm, several criterions were evaluated, such as correct classification rate, false positive rate, computation time, etc. Figure 2 shows the correct classification rates obtained for each algorithm.

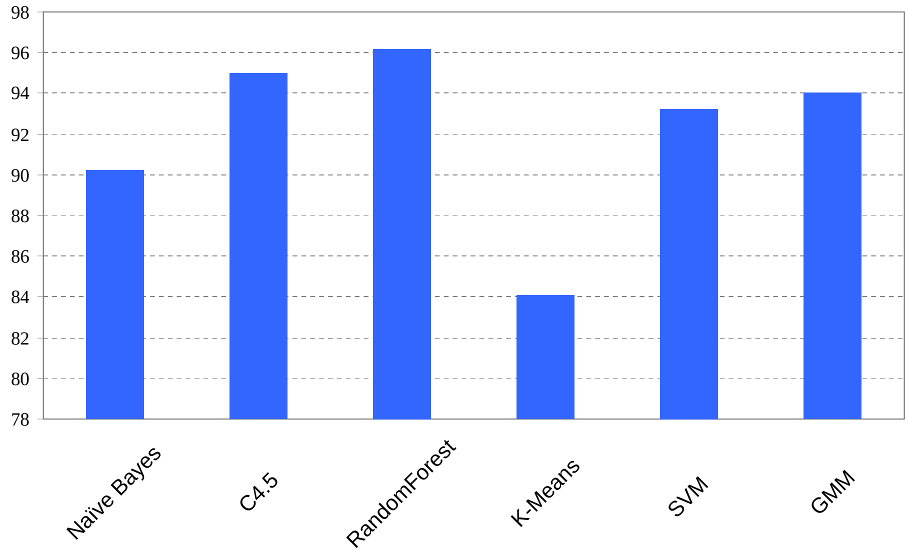

Figure 2: Correct classification rates for tested machine learning algorithms

It appears that *RandomForest*, a machine learning tool never applied before to flow classification, leads to the best performances in terms of correct classification rate and computation time.

3.0 IMPACT OF TLS ENCRYPTION ON CLASSIFICATION PARAMETERS

Previous experiments were carried out on a database made of clear flows. Unfortunately, there is no publicly available payload trace set composed of ciphered flows. Our work aims at demonstrating the feasibility of tunnel detection for ciphered flows, and thus it is necessary to prove that results similar to those mentioned above would be obtained on ciphered flows. We conducted a complementary study to evaluate the impact of encapsulation on classification parameters. In particular, we studied the effect of TLS encryption on the set of 10 parameters we use to classify a flow (note that TLS encryption is used to establish an HTTPs tunnel) following these steps :

- pairs of clear/ciphered flows and extracted are generated for different protocols (HTTP, SCP, SSH, etc.) ;
- the classification features are extracted for each flow ;
- an affine transformation function from clear to ciphered was estimated for each parameter ;
- the accuracy of these transformation functions was estimated by calculating the residual quadratic error of approximation.

The results obtained showed that the transformation induced by TLS encryption on classification parameters can be correctly approximated by affine functions for 8 features out of 10. On the opposite, two of them (minimum inter-arrival delays between packets from client to server and variance of the size of packets from server to client) were transformed in a more complex way.

We can reasonably conclude from this results that TLS encryption will not lead to a significant loss of performance for the classification algorithm mentioned above.

4.0 A TUNNEL DETECTION TOOL ARCHITECTURE

The biggest drawback of statistical methods is their high rate of false positive (*i.e.* legitimate flows classified as malicious). We propose a specific tunnel detection tool architecture designed to lower the false positive rate. Figure 3 describes this architecture.

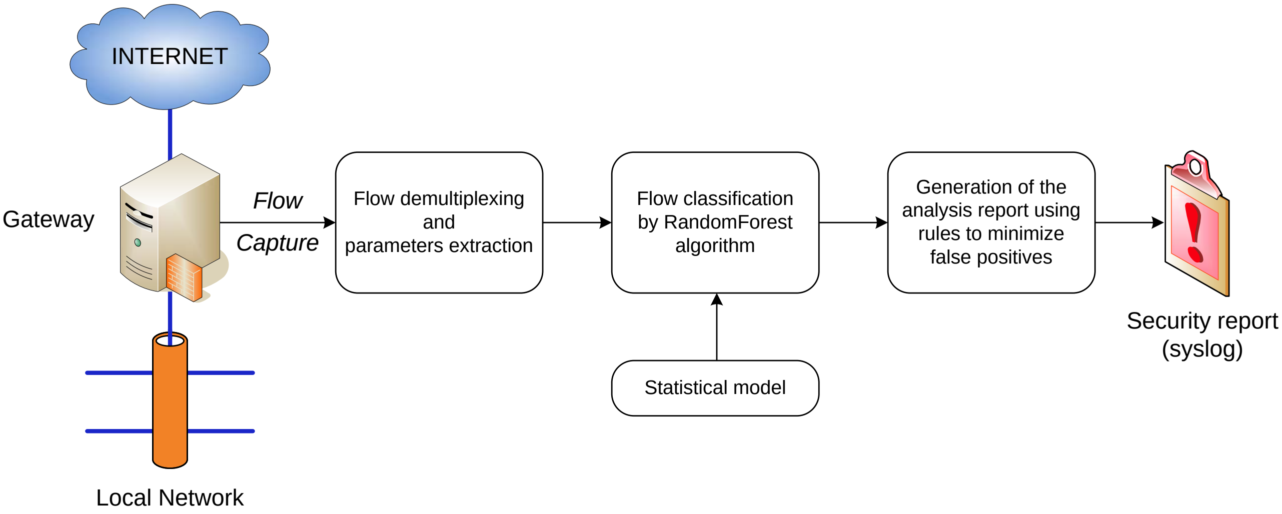
INTERNET
Gateway
Flow
Capture
Flow demultiplexing and parameters extraction
Flow classification by RandomForest algorithm
Generation of the analysis report using rules to minimize false positives
Security report (syslog)
Statistical model
Local Network

Figure 3: High level tunnel detection tool architecture

The system consists in a network capture tool (such as `TCPDump` [12]) combined with a flow demultiplexer. Classification features are then extracted from each flow, and a RandomForest model is used to determine the class of each connection. In order to minimize false positive cases due to errors of classification, a set of heuristic rules is applied to generate an analysis report composed of a list of alerts. These rules take into account past results of classification, and a level of confidence for each classification. No alert is raised if the confidence level is too low, if the IP address of the local or remote host is on a white list, etc. The analysis report generated by the application of this set of rules could have the *syslog* format, for future integration in a complex intrusion detection system.

The proposed architecture was implemented on an experimental platform and give very encouraging qualitative results. These results are presented in the next section.

5.0 QUALITATIVE RESULTS OF THE PROPOSED SOLUTION

5.1 Network simulation

At first, we implemented our detection tool on a network simulator. The simulator consisted in 3 machines, simulating respectively the local network, the gateway and the Internet. This simulator has been used to measure the TLS impact (section 3.0) and the efficiency of the detection tool. The resulting detection rates for the protocols shown in table 2 are close to 100%. However, this did not provide a convincing proof because the diversity of the flows is reduced compared to a real network :

- the topology of the network is too simple ;

- the behavior of the user is unique;
- the material is also unique (one OS, one hardware, etc.).

The results obtained for the TLS impact remain valid, but in order to evaluate the accuracy of the tool, a more complex set of flows had to be tested.

5.2 A flows database in order to evaluate our detection tool

The public database containing real data flows used for our experimentations is provided by the MAWI working group [13]. The database is a recording of the whole set of flows carried by a transpacific 150Mb/s network line between Japan and USA, during 96 hours. The payloads have been removed and the headers from layers 1 to 4 from the OSI model have been anonymised.

In order to illustrate the performance of our solution, we classified nine kind of network flows. For each protocol, the number of flows contained in the database and used for the experimentation is shown in the table 2.

HTTP	HTTPs	SSH	SMTP	DNS (over TCP)	FTP	Active Directory	POP3s	NetSteward
2500	**2500**	**2500**	**2500**	**2500**	**2500**	**1069**	**1503**	**1611**

Table 2: Distribution of the database flows according to the protocols

Note that the flows used for the experimentation are mostly clear flows, *i.e.* unciphered flows. Indeed, there is unfortunately no public database of ciphered flows precising for each flow which protocol is ciphered. Nevertheless, our analysis with this database is interesting and can be extended to ciphered flows for the following reasons :

- the flow classification features can be calculated with ciphered flows exactly as for the clear flows ;
- the impact of ciphering on the parameters is limited.

Parameters like the delay induced by the user behavior (as the password capture for an SSH session or the frequency of HTTP request while surfing) are not affected by the encryption.

5.3 Classification results

Table 3 shows the corresponding confusion matrix obtained with this algorithm. The procedure used to get the confusion matrix is :

1. For each flow, compute the features regarding the full connection ;
2. Train the classifying model (*i.e.* RandomForest) on a subset (*the learning set*) of flows ;
3. Challenge the model on the remaining vectors (*the challenge set*) ;
4. Report the results.

For example in this table :

- the number *93.08* in the first row indicates that 93.08% of *HTTP* flows have been correctly classified as *HTTP* ;
- the number *4.36* in the first row indicates that 4.36% of *HTTP* flows have been erroneously classified as *HTTPs*.

Therefore, the correct classification rates are on the table's diagonal. The average rate of correct classification is 95.81%.

HTTP	HTTPs	SSH	SMTP	DNS	FTP	Active Directory	POP3s	NetSteward	
93.08	4.36	0.0	1.08	0.04	0.24	0.08	0.36	0.76	HTTP
2.36	**91.56**	0.08	3.2	0.0	0.48	0.48	2.36	0.28	HTTPs
0.0	0.12	**99.44**	0.08	0.0	0.08	0.0	0.28	0.0	SSH
0.96	2.28	0.0	**91.12**	0.2	3.48	1.12	0.72	0.12	SMTP
0.0	0.0	0.0	0.32	**99.64**	0.0	0.04	0.0	0.0	DNS
0.08	0.6	0.0	3.0	0.0	**95.88**	0.2	0.24	0.0	FTP
0.19	0.09	0.0	0.47	0.0	0.0	**99.16**	0.0	0.09	ActiveDirectory
0.13	1.2	0.27	0.73	0.0	0.0	0.0	**97.67**	0.0	POP3s
1.37	0.06	0.0	0.0	0.0	0.0	0.0	0.0	**98.57**	NetSteward

Table 3: Confusion matrix obtained using the RandomForest method to classify the database

In a standard configuration, the only allowed protocol might be *HTTP* and *HTTPs*. Any flow classified in an other class (e.g. SSH, POP3, . . .) would then be considered as malicious. Hence, if we set this configuration, the tool detects 98.68% of illegitimate flows (corresponding to 1.32% of false negatives) with 4.72% of false positives (*i.e.* false alarms). This last rate is too high for an actual use, since most of flows are legitimate. In section 5.7, we propose a way to decrease the number of false alarms sent by the tunnel detection tool.

5.4 Classification computation time

As shown in table 4, the classification computation time is quite short. The implementation has been realized on a 3.06 Ghz PC platform running under a Debian distribution. The langage used is Java, therefore this computation time could be reduced using a faster langage such as C if needed.

Phase	Time
Learning Phase	**1143** ms
Challenge Phase	**223** μs

Table 4: Computation time with a 2500 flows database

5.5 Impact of the flows length

The procedure described in section 5.3 works with a full connection. Thus, it does not allow the gateway to take a real time decision such as ending a session as soon as an illegitimate flow is detected (the decision is *a posteriori*). In order to take a pro-active decision, a small number of packets can be used rather than the full connection. As a consequence, it increases dramatically the computation power required by the security gateway. Our study showed that the decision can be taken with only very few packets (about 3 packets). This could be explained by the fact that the considered protocols have different behaviors from the beginning of the connection, which helps to distinguish them with a small number of packets.

5.6 Impact of the database size

Another issue is the size of the learning database. Depending on the context, it may be hard to generate a large database for each flow. For example, the database built with our simulator had to be manually filled. The figure 4 illustrates the impact of the database size on the detection accuracy.

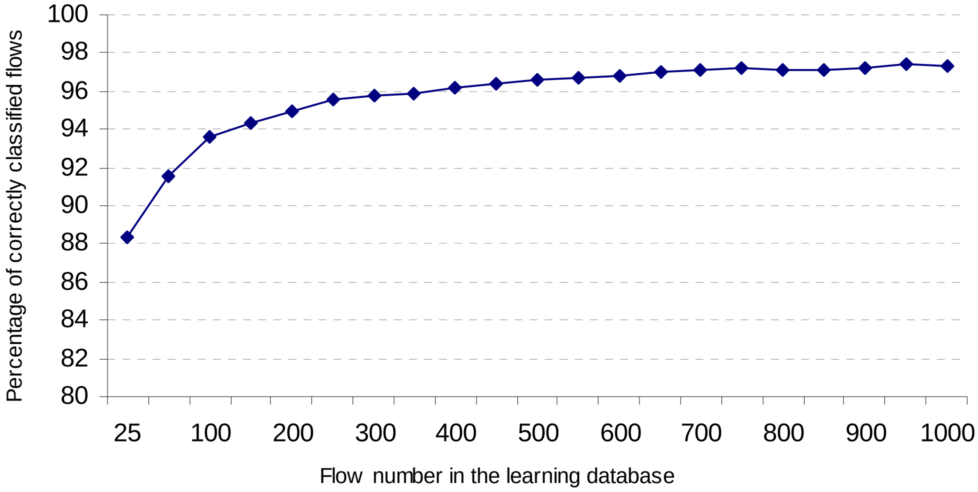

Figure 4: Impact of the database size on the detection accuracy

5.7 A simple method to lower the false positive rate

We saw in section 5.7 that the false positives rate (*i.e.* legitimate flows classified as malicious) is too high for an actual use while the illegitimate flows rate is, on the opposite, very good. Depending on the use case, it could be better to limit the number of false positives, because it could disturb most of the network users.

For this reason, we propose to set a confidence indicator. Therefore, a flow with a confidence indicator below a specific threshold will be automatically considered as legitimate. This rule can be added in the heuristic part of the tunnel detection tool architecture (*cf.* figure 3).

The figure 5 shows the rates of false positives and false negatives obtained by applying this simple heuristic, based on the confidence indicator set. We can see that such a rule can reduce the false positives rate. However, this method seems too 'naive', because the increase of false negatives rate (*i.e.* illegitimate flows allowed by the security gateway) is significantly faster than the decrease of false positives rate.

CONCLUSION

In this paper, we presented a solution to the key problem of encapsulated illegitimate flows detection across network boundaries. In a first part, we compared the performances of different machine learning algorithms and identified the best one in our specific case. In a second part, we conducted a complementary study showing that the effect of TLS encryption on classification features should not significantly affect classification performances. Finally, in a last part, we described a high-level tunnel detection tool architecture. We pointed out qualitative results using this tool with a public database and the impact of variation around the protocol on its accuracy. Finally we proposed, regarding the results obtained, a simple method to lower the false positive rate.

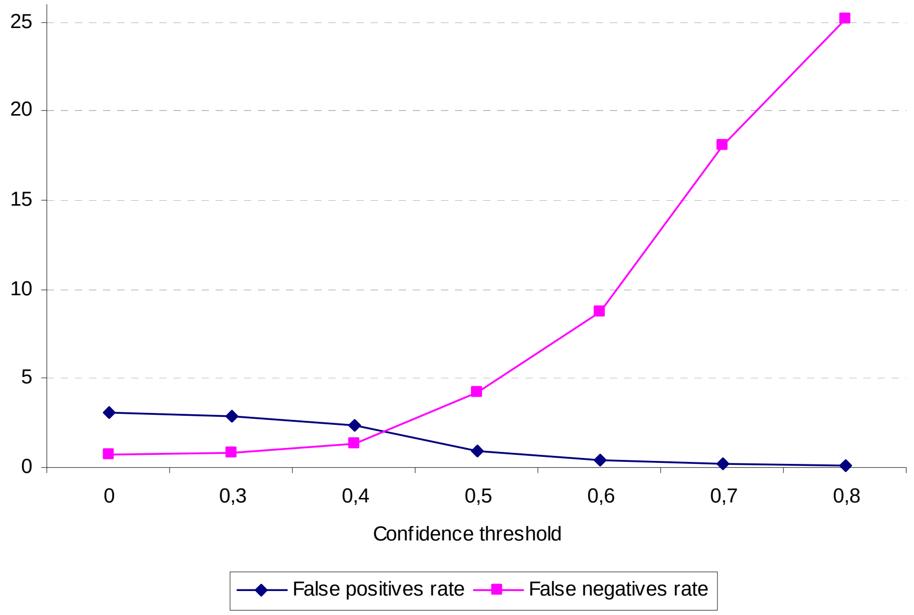

Figure 5: Impact of a rule based on a confidence indicator on the rates of false positives and false negatives

The construction of our solution is generic and can be tuned to be used for automatic classification, pro-active reaction or small learning database. In a global cyberdefense system, the proposed architecture could be efficiently used with a classical security tool, such as an IDS, in order to improve the security level.

REFERENCES

[1] *HTTPHost*, http://www.htthost.com

[2] *STunnel*, http://www.stunnel.org

[3] J. Erman, A. Mahanti and M. Arlitt, *Internet traffic identification using machine learning*, IEEE GLOBECOM'06, 2006.

[4] T. Karagiannis, K. Papagiannaki and M. Faloutsos, *BLINC: Multilevel traffic classification in the dark*, ACM SIGCOMM'05, 2005.

[5] A.W. Moore and D. Zuev, *Internet traffic classification using bayesian analysis techniques*, ACM SIGMETRICS'05, 2005.

[6] T.T. Nguyen and G. Armitage, *A survey of techniques for Internet traffic classification using machine learning*, IEEE Communications Surveys and tutorials, 2008.

[7] L. Bernaille and R. Teixeira, *Early recognition of encrypted applications*, PAM2007, 2007.

[8] M. Dusi, M. Crotti, F. Gringoli and L. Salgarelli, *Detection of encrypted tunnels across networks boundaries*, ICC'08, 2008.

[9] A.W. Moore, D. Zuev and M.L. Crogan, *Discriminators for use in flow-based classification*, Technical report, 2008.

[10] W. Li, M. Canini, A.W. Moore et R. Bolla, *Efficient application identification and the temporal and spatial stability of classification schema*, Computer Networks 53, 2009.

[11] N. Williams, S. Zander and G. Armitage, *A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification*, ACM SIGICOMM'06, 2006.

[12] *Tcpdump/Libpcap*, http://www.tcpdump.org

[13] *MAWI Working group traffic archive*, http://mawi.wide.ad.jp/mawi/