NBA of Obfuscated Network Vulnerabilities' Exploitation Hidden into HTTPS Traffic

Ivan Homoliak*, Daniel Ovsonka*, Matej Gregr* and Petr Hanacek*
* Department of Intelligent Systems
Faculty of Information Technology, BUT
Brno, Czech Republic
{ihomoliak, iovsonka, igregr, hanacek}@fit.vutbr.cz

***Abstract*—This paper examines the detection properties of obfuscated network buffer overflow attacks by selected IDS and NBA. The obfuscation was performed by tunneling the malicious traffic in HTTP and HTTPS protocols with the intention of simulating the usual legitimate characteristics of the HTTP traffic's flow. The buffer overflow vulnerabilities of four services were used: Samba, BadBlue, Apache, DCOM RPC. Exploitation was performed in a virtual network environment by using scenarios simulating real traffic's conditions as well as legitimate traffic simulations which were performed. Captured data were examined by SNORT and by ASNM network features of the AIPS representing statistically and behaviorally based NBA. The achieved results show an obfuscated attacks transparency for SNORT detection and low detection performance of the AIPS trained by direct attacks and legitimate traffic only in contrast with high classification accuracy of the AIPS trained with an inclusion of obfuscated attacks. Data mining analysis was performed by using both bi-nominal and poly-nominal classifications, resulting into better performance of poly-nominal classification. At the summary, we emphasize the necessity of training the statistically and behaviorally based NBAs with divergent obfuscation techniques to strengthen their detection capabilities.**

***Keywords*—traffic obfuscation; protocol tunneling; buffer overflow; network vulnerabilities; NBA; IDS**

I. Introduction

Buffer overflow attacks exploiting the network vulnerabilities continues to be the one of the most dangerous threads in the domain of information security.

This class of attacks form a substantial portion of all security attacks simply because buffer overflow vulnerabilities are very easy to perform [1], [2], [3]. Buffer overflow attacks dominate in the class of remote penetration attacks because a buffer overflow vulnerability presents the attackers exactly what they need: the ability to inject and execute an attack code. The injected attack code runs with the privileges of the vulnerable application and allows the attacker to bootstrap whatever other functionality in order to control the compromised PC [4].

The impact can be very crucial either in commercial or personal network environments nearly every time. This fact encourages many researchers and developers to design new methods and approaches for detection of known and unknown (zero-day) network attacks. All designers have to deal with poly-morphism (code encryption) and meta-morphism (code obfuscation) enabling the malware to avoid positive signature matching of the malware's behavior or the malware alone. This is the reason why the most of the intrusion detection systems (IDS) fail in detecting such malware and their consequences or behavior. Therefore, researchers tend to use various artificial intelligence (AI) and data mining techniques or their combinations in order to achieve as accurate result as possible.

The authors of the paper [5] described a method called Automated Intrusion Prevention System (AIPS) which uses extended behavioral and statistical meta-information extracted from networks communications called Advanced Security Network Metrics (ASNM) originally designed in [6]. The AIPS uses honeypot systems as a source of expert knowledge for AI models learning. Their next work [7] performs an analysis of the results achieved by various data mining methods using data captured by simulated attacks in laboratory conditions. A related paper [8] describes a formal definition of the ASNM extraction process and performs experiments comparing performance of the ASNM with state-of-the-art network features set designed by A. Moore [9] using a publicly available dataset CDX 2009 [10]. The presented results of this paper show the similar NBA detection properties of the both features sets (offering high attacks' detection capability).

The authors demonstrated the applicability of their approach by performing the experiments which detect network attacks with consideration: an attack executes 'dangerous' communication directly from its source machine laying outside of the attacked network. They did not consider that network vulnerability can be exploited from the inside of the attacked network, using a previously exploited machine without any alert of IDPS or NBA. This consideration became the inspiration for our actual research. We consider the previously exploited machine which serves as a mediator between attacker and new vulnerable target machines. This machine performs obfuscation by the tunneling of every communication with attacker. The obfuscation we decided to use in this work was previously designed in [11].

The paper is organized as follows. Section II discusses related work in the field of network traffic obfuscation with an emphasis on malicious traffic and statistical analysis of the communication's behavior. Section III describes the method of obfuscation we used and our network architecture. Section IV describes simulations of our experiments on specific network services with various testing scenarios as well as mentioning techniques used for IDS and NBA detection. Section V describes data processing and analysis of captured data and section VI presents the summary of achieved data mining results. The conclusion is presented in section VII.

II. Related work

The authors Fogla et al. realize the obfuscation of network attacks by proposing a new class of polymorphic attacks, called polymorphic blending attacks (PBA) [12], which can effectively evade byte frequency-based network anomaly IDS. The attacks carefully match the statistics of the mutated attack instances to the normal profiles. They demonstrate the efficiency of PBA attacks on PAYL. In the next paper [13] they show that in general, generating a PBA that optimally matches the normal traffic profile is a hard problem (NP–complete), but can be reduced to SAT or ILP problems. They present a formal framework for PBA attacks and also propose a technique to improve the performance of an IDS against PBAs.

The paper [14] shows how obfuscated application layer protocols, such as BitTorrent's MSE [15] or Skype [16], can be identified by an analysis of statistically measurable properties of TCP and UDP sessions. The authors depict that many of the analyzed protocols have statistically measurable properties in payload data, flow behavior, or both. Based on their insight, they propose a few techniques for improving protocol obfuscation which inhibits traffic identification by statistical analysis. These techniques include better obfuscation of payload data and flow properties as well as hiding inside tunnels of well-known protocols. The purpose of this work is not to provide more effective intrusion classification of NBA systems, but rather to provide feedback to protocol creators who want to contribute on the network neutrality of the Internet.

Dusi et al. presented a mechanism called Tunnel Hunter, which can successfully identify protocols tunneled inside tunneling protocols such as HTTP, DNS and SSH [17]. It is performed by the statistical analysis of simple IP level flow features (i.e. packets sizes, inter–arrival time and packet order). Their technique suffers from the problem of sensitivity to packet-size and timing value manipulation.

Bar-Yanai et al. presented a method for real-time classification of encrypted traffic [18]. The proposed statistical classifier is based on a hybrid combination of k-means and k-nearest neighbor geometrical classifiers and is shown to be very robust, even to obfuscated traffic such as Skype and encrypted BitTorrent. The statistical feature set is composed of 17 parameters based on packet and payload byte counts, packet sizes and packet rates for each direction.

The authors of the work [19] deal with buffer overflow attacks by an internal analysis of exploits content. They propose method scanning network traffic for the presence of a decryption routine, which is characteristic for obfuscated malware. The method uses static analysis and techniques for emulated instruction execution. It has been implemented and tested on polymorphic exploits, including ones generated by state-of-the-art polymorphic engines [20], [21] and Metasploit [22]. They achieved a false positive rate close to 0%. This method supposes deciphering of communication data and thus it can be used only as a source of expert knowledge from the view of NBA.

Sommer et al. [23] examine the surprising imbalance between the extensive amount of research on machine learning-based anomaly detection pursued in the academic intrusion detection community, versus the lack of operational deployments of such systems. They claim that the task of finding attacks is fundamentally different from other tasks, making it significantly difficult for the intrusion detection community to employ machine learning effectively. They support this claim by identifying challenges particular to network intrusion detection and then, they provide a set of guidelines meant to strengthen future research on anomaly detection.

III. Obfuscation of attacks

The obfuscation of malicious communications was created with the aim of similarity maximization of obfuscated exploiting data and real network traffic [11]. The major requirement of the obfuscated traffic is obfuscation's transparency for upper network layers. Based on these requirements, the Hyper Text Transfer Protocol (HTTP) and HTTP Secure (HTTPS) protocols were selected as carrying protocols, thus the obfuscation is based on the encapsulating of suspicious data into standard HTTP or HTTPS packets. This makes the obfuscated data hard to detect by classic signature based approaches and even by statistical and anomaly based methods. Another reason why these protocols were selected is because they are heavily spreaded in nearly all computer networks, so it gives a higher probability of obfuscated communications that would not be detectable. The advantage of these protocols is also their challenge-response character which is ideal for our testing scenarios.

In our approach, we assume a private network connected to an outer network through the gateway which uses the Network Address Translation (NAT) and is monitored by the Intrusion Detection System (IDS) and by the Network Behavioral Analysis system (NBA). The network architecture is shown in figure 1. On the right side of the picture the private network with vulnerable machines and a previous exploited machine is illustrated. The left side of the picture represents the outer network with the legitimate user and the attacker. In the middle of the picture is situated the gateway which interconnects inner and outer networks.

The gateway is running on Debian Linux with a 2.6.32-5-686 kernel. The attackers station and the exploited machine are running on Ubuntu Linux with a 3.11.0-12 kernel. The Windows server is running on the Windows 2000 with SP4 and the Linux server is running on the Red Hat Linux with a 2.4.7-10 kernel.

The obfuscation system is divided into two separate modules. When the first one – named the Callback, has to be deployed into the internal network, the second one – called the Fake HTTP Server, acts as the remote HTTP server. The installation of the Callback module into the internal network can be done by the exploitation of a target machine laying at the internal network without detection of IDS or NBA system.

The Fake HTTP Server module of the obfuscation waits for a connection. When the connection is established from the Callback module, all traffic affected by obfuscation is tunneled through the carrying protocol. This module represents the main logic of the system because it has to apply advanced filters to all traffic and select only relevant packets for obfuscation.

The Callback module acts as a proxy because the main function of this module is to translate incoming encapsulated communications, restore their original content and then

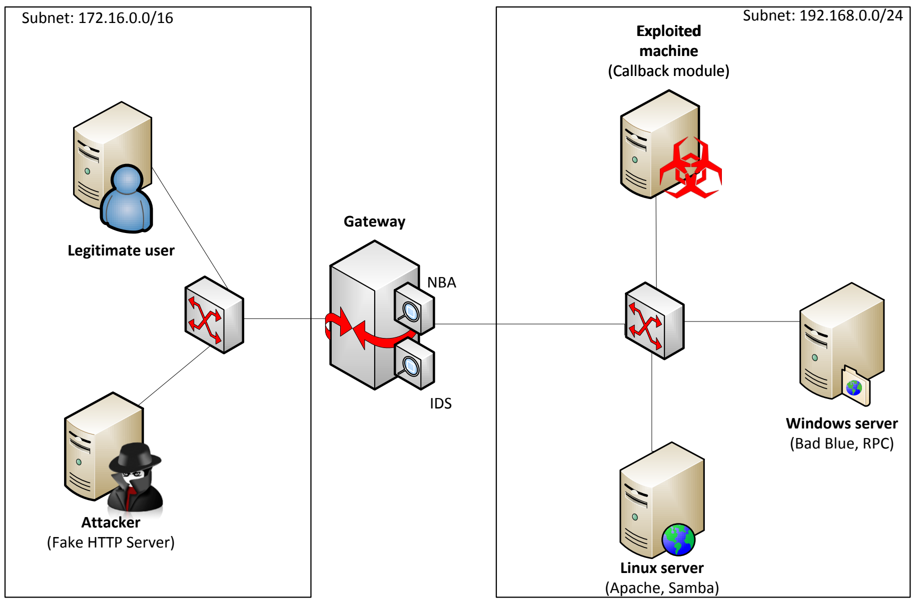

Figure 1: Scheme of virtual network's architecture.

distribute it into the internal network. The Callback module also forges IP addresses inside of processed packets, thus the responses of internal network can be caught by the Exploited machine. The caught packets are then encapsulated and sent as a callback message to the external Fake HTTP Server machine.

A. *Implementation notes*

All obfuscation routines are implemented at a lower network level. Transmitted encapsulated packets are bypassed using standard `iptables` with custom dynamic rules. Dynamic rules depend on a character of obfuscated communication, therefore it is important not to process all packets due to performance impact on host system. Next, packets are processed through the `libnetfilter_queue` library which provides the interface for working with packets in the user space. The packets affected by obfuscation are encapsulated here or restored from carrying packets before they reach the kernel space for regular processing. This solution is transparent for upper network layers and the host kernel cannot detects whether packets were modified or not. The solution is also protocol independent and no other configuration of host system is necessary.

IV. Experiments' description

For the purpose of our work we created a set of experiments in a virtual network environment. In these experiments we simulated both legitimate and malicious traffic. The legitimate traffic's experiments were performed manually because of adding some human factor and diversity to data and furthermore, we want to reach maximal authenticity of users behavior. Malicious content was created using Metaspolit [22] and some widespread public exploits. Well-known attacks were selected because they are easily detectable by classic IDSs, thus we can demonstrate functionality of our obfuscation technique.

Traffic was dumped and saved on the gateway node because traffic seen on the gateway contains all transmitted packets during our experiments. Packets were dumped using the Wireshark tool [24] directly from a software network bridge interface which interconnects internal and external networks. Data obtained here serves as input for further post-processing and analysis.

A. *Testing scenarios*

Each experiment was executed four times in different conditions for every single vulnerability and legitimate communication. Our testing scenarios should improve the relevance of the experiments' outputs and make our synthetic data more similar to real network traffic. We used a different configuration of virtual network environment for each testing scenario. Testing scenarios are divided into four categories:

1) The first scenario represents reference output without any modification to configuration. All experiments ran on the same host machine to minimize deviations among different tests.
2) The second scenario is dedicated to simulate traffic shaping. Therefore, all packets were forwarded with higher time delays. For this purpose, the special gateway machine with limited processor's performance was used. This machine was also fully loaded to emulate slower packets processing than in the first scenario.
3) The third scenario is supposed to simulate traffic policing when some of packets were dropped during the processing on the network gateway node. In this case, a custom packet dropper was used on the gateway node and 25% of packets were dropped, resulting in output which contains re-transmitted packets.
4) The fourth scenario represents transmission on an unreliable network channel, thus 25% of packets were

corrupted during processing on the networks gateway node.

B. Vulnerabilities

The vulnerabilities were selected with the aim of getting the variety of different attacks. Our experiments were executed on four vulnerabilities – two against a Linux based machine and the next two against a Windows based machine. These machines correspond to the PC stations in the right segment of Fig. 1. Each attack exploits well-known vulnerability and its execution leads to full privilege escalation, thus the attacker can get the root's permission. The details about each vulnerability and its exploitation are the following:

- **The Apache web service** – this attack exploits buffer overflow vulnerability in `mod_ssl` plugin of an Apache web server (old versions up to 1.3.22), so the obfuscated communication had to be transparent to higher network layers. All packets of this attack are sent to port number 443 of a vulnerable web server. This attack is not detectable by standard signature based approaches (including SNORT) due to the encryption of all packets. The CVSS score of this vulnerability is 7.5 and it is described by CVE-2002-0082 [25].
- **Bad Blue web service** – the second attack exploits a stack-based buffer overflow vulnerability in a Bad Blue web server (version 2.72b). In this case port number 80 was used. In the attack performing phase, the special crafted packet with a long header is sent which leads to an overflow of processing buffer. This attack is easily detectable by signature based detection systems. The CVSS score of this vulnerability is 7.5 and it is described by CVE-2007-6377 [26].
- **Microsoft DCOM RPC** – the third attack uses vulnerability in Microsoft Windows DCOM Remote Procedure Call (DCOM RPC) service of old versions of Windows 2000 (up to Service Pack 4) and Windows XP. A standard RPC service runs on port 139. This vulnerability allows a remote attacker to execute an arbitrary code based on buffer overflow in DCOM interface. The vulnerability is well documented and it was used, for example, by Blaster worm. The CVSS score of this vulnerability is 7.5 and it is described by the CVE-2003-0352 [27].
- **Samba service** – the last selected exploit is aimed to a buffer overflow in `call_trans2open` function for the same versions of **Samba**. This vulnerability allows a remote attacker to execute an arbitrary code. In our case public exploit was used which sends malformed packets to a remote server in batches. Packets differ in a one shell-code address only because the return address depends on versions of Samba and host operating systems. The CVSS score of this vulnerability is 10 and it is described by CVE-2003-0201 [28].

C. IDS detection

For the purposes of IDS detection, we used the SNORT 2.9.4 with rules distributed as snapshot 2940. There were performed direct attacks on previously described vulnerable services. All attacks were captured by the SNORT except the attack on the Apache service. In the Apache service case, the communication was encrypted, thus the signatures could not be matched in the payload. Captured SNORT alert messages are listed below and are delimited by newline in a particular vulnerability exploitation:

- **BadBlue**:
```
ET SHELLCODE Rothenburg Shellcode
INDICATOR-SHELLCODE x86 OS agnostic fnstenv
    geteip dword xor decoder
http_inspect: LONG HEADER
```
- **DCOM RPC**:
```
ET SHELLCODE Rothenburg Shellcode
INDICATOR-SHELLCODE x86 OS agnostic fnstenv
    geteip dword xor decoder
OS-WINDOWS DCERPC NCACN-IP-TCP
IActivation remote activation
    overflow attempt
```
- **Samba**:
```
GPL NETBIOS SMB trans2open buffer
    overflow attempt
GPL NETBIOS SMB IPC$ share access
```

Next, we performed obfuscated exploitation of each vulnerable service and there were not generated any alerts by the SNORT. Therefore, the tunneling obfuscation of attacks was successful in the breaking of the signature based IDS.

D. NBA detection

NBA detection was performed by the AIPS system, which uses ASNM metrics as connections' features [8]. The ASNM features are used to describe properties of an analyzed connection based on its statistical and behavioral characteristics. It contains 167 network features divided into five categories. The ASNM uses the context of an analyzed connection to compute some of the features. The context represents time bounded TCP connections set with the same destination and source IP addresses as an analyzed connection has.

We do not used the whole architecture of the AIPS, but only two parts of the component Network Detector [7]: Connections extractor and ASNM extractor. The AIPS works with TCP connections only because of its unambiguous packets association to connections. The summary of achieved results will be discussed in section VI.

V. Data processing and analysis

The whole process of data processing and analysis stage is illustrated in Fig. 2. There are 3 segments and data flow direction is shown from the top to the bottom of the scheme. Empty boxes represent data as input or output of some processes and filled ovals represent working components which perform some action. A working component takes input data and outputs output data. The upper segment represents the input of the whole experiment process and includes:

- **TCP dump files** – collected during the simulation of attacks and legitimate communications.

TABLE I: TRAFFIC CLASS DISTRIBUTION (TCP CONNECTIONS AND PACKETS)

Service	Legitimate traffic		Direct attacks		Obfuscated attacks	
	connections	packets	connections	packets	connections	packets
Apache	268	4246	101	1741	74	3868
BadBlue	170	2035	4	39	10	198
DCOM RPC	222	2077	4	50	8	184
Samba	20	531	20	357	8	712

- **Directory structure** – it contains information about a kind of simulation (attack or legitimate), identification of a network service, identification of vulnerability (CVE [29]).
- **Mapping of services to hosts** – it contains the IP addresses of hosts relevant to our simulations with mapping of the analyzed services to hosts. It included information about obfuscation tunnel's endpoints too.

The middle segment of the scheme represents a process of ASNM network features extraction. Two upper components of this segment serves for parsing TCP dump files with parallel expert knowledge processing. These two components take all inputs and exports processed data ready for storing into persistent database storage. The next component DB importer achieves persisting of processed data. Next, an active component Connection extractor performs the identification of all TCP connections in database. It produces a list of TCP connection objects with embedded expert knowledge information. Then, ASNM features extraction is performed for each TCP connection by the Metrics extractor component and the results of this step are ASNM features values for each TCP connection object in the CSV representation.

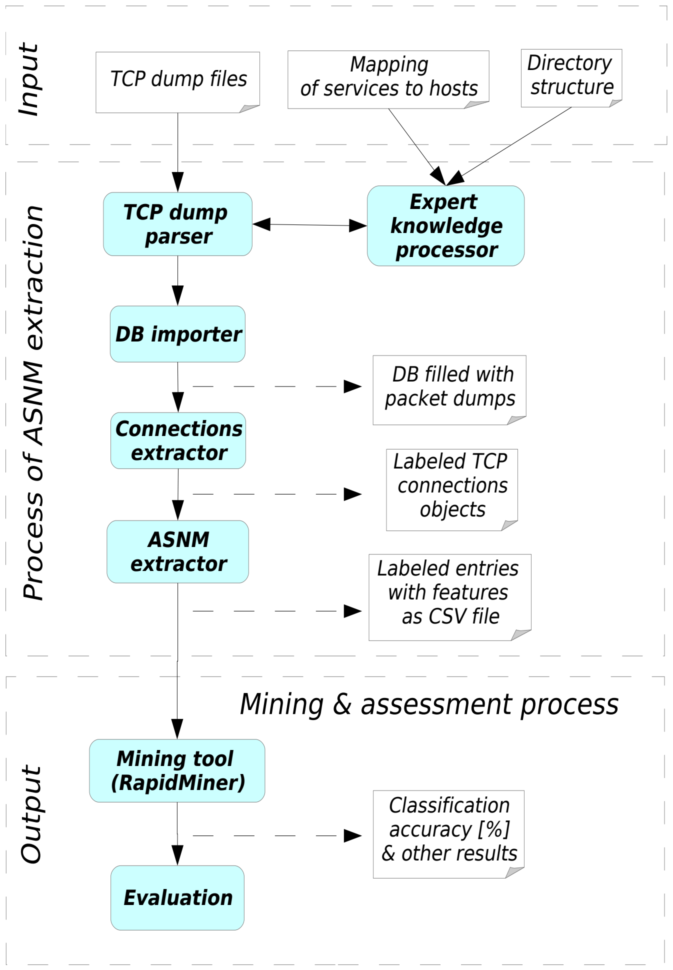
Input
TCP dump files
Mapping of services to hosts
Directory structure
Process of ASNM extraction
TCP dump parser
Expert knowledge processor
DB importer
DB filled with packet dumps
Connections extractor
Labeled TCP connections objects
ASNM extractor
Labeled entries with features as CSV file
Mining & assessment process
Output
Mining tool (RapidMiner)
Classification accuracy [%] & other results
Evaluation

Figure 2: Scheme of data processing and analysis.

The last segment of the scheme illustrates the mining and assessment process, which produces the output results of the analysis. The RapidMiner [30] tool for the mining purposes was used there. The class distribution of packets and connections is depicted in Table I.

VI. SUMMARY OF THE RESULTS

In this section we describe data mining experiments from two perspectives: bi-nominal and poly-nominal classification according to the interpretation of processed data. The communication of collected and processed data can either be interpreted as attacking and legitimate communications in binary representation or they can be interpreted as attacking and legitimate communications at the specific network service in the poly-nominal representation. Our poly-nominal classification includes three classes for each service (resulting in 12 classes) and one class for other captured traffic.

Every result is interpreted as a performance vector and a confusion matrix with specific recall and precision of classes. Each table representing a confusion matrix contains a performance vector representing an overall classification accuracy with validation precision, which is situated in the left top cell of the table.

The names of classes in the bi-nominal representation are True or False indicating a malicious or a legitimate behavior of communications respectively. The identification of classes in a poly-nominal case consists of two parts. The first part represents the subclass of the connection specifying its maliciousness and may contain three labels with an intuitive representation: legitimate traffic, direct attacks and obfuscated attacks. The second part of the identification indicates the specific acronym of a service. Every presented experiment uses the Naive Bayes classifier. Most of the experiments employ the 5–cross fold validation. The exceptions are the first bi-nominal and the first poly-nominal classifications experiments. Both categories of classifications use a forward feature selection to find the best features with emphasis on the selection of time independent features (inspired by problematic issues of [17]).

A. *Bi-nominal classification*

In the experiments of bi-nominal classification were all numeric attributes discretized into five bins due to a better

TABLE II: REFERENCE CONFUSION MATRIX

Accuracy: 97.64% (± 0.45%)	True True	True False	Class precision
Predicted True	151	3	98.05%
Predicted False	16	635	97.54%
Class recall	90.42%	99.53%	

classification performance. First, we performed an experiment which performs a malicious obfuscated traffic detection by a classifier trained by direct attacks and legitimate traffic only. Many modifications of classifier's settings and preprocessing phases to improve it were performed and as a result we achieved classification accuracy and recall of the attack class which were both equal to 24%.

The result of our first bi-nominal classification experiment was an impulse for the second one – we proposed the obfuscated and the direct attacks class unification. Therefore, we validated the classifier by direct attacks and obfuscated attacks labeled as one class for the purpose of this experiment (True label). The resulting classification accuracy and the confusion matrix are shown in Table II. The outcome of this experiment indicates a fine classification accuracy with respect to the class precision of the model trained by explicit information about obfuscated attacks.

The third experiment describes a validation of classifier using different subsets of training data by means of obfuscated or direct attacks exclusion. Therefore, in the first case we validated it by direct attacks and legitimate traffic only and in the second case we validated it by obfuscated attacks and legitimate traffic only. The results we achieved are depicted in Table III.

TABLE III: OBFUSCATED/DIRECT ATTACKS' EXCLUSION

(a) OBFUSCATED ATTACKS EXCLUDED

Accuracy: 99.18% ± 0.50%	True True	True False	Class precision
Predicted True	101	0	100.00%
Predicted False	6	628	99.05%
Class recall	94.39%	100.00%	

(b) DIRECT ATTACKS EXCLUDED

Accuracy: 98.09% ± 0.75%	True True	True False	Class precision
Predicted True	42	2	95.45%
Predicted False	11	625	98.27%
Class recall	79.25%	99.68%	

The outcome of this experiment shows a better classification accuracy for both cases in comparison with the reference accuracy from the second experiment, and it also indicates more accurate results of separated obfuscated and direct attacks' representation for malicious traffic detection. In the other words, the outcome indicates the different behavior of obfuscated and direct attacks because there it was more difficult to represent direct attacks and obfuscated attacks together as one class which resulted in a lower classification accuracy and lower class precision in the case of obfuscated and direct attacks unification.

The fourth binary classification experiment differs from the previous one in assumption, where obfuscated attacks or direct attacks can be undetected by NBA and therefore, they are labeled as legitimate communications in the training phase. We performed this assumption by neutralization of the target malicious class – labeling it as legitimate. Accordingly, we performed two validations: the first contained legitimate labels for the direct attacks' class and the second contained legitimate labels for the obfuscated attacks' class. The results we achieved are depicted in Table IV.

The accuracy results of the fourth experiment are lower in the cases of neutralized direct attacks and neutralized obfuscated attacks in comparison with originally labeled data (Table II), which stands for various statistical and behavioral characteristics of these classes. This fact can be observed when we compare the results of the previous experiment with the result of this one. In the actual experiment there is achieved a lower classification accuracy in both cases than in the previous experiment.

B. Poly-nominal classification

Our first experiment classifies obfuscated malicious traffic by the classifier which was trained by direct attacks and legitimate traffic only in order to test obfuscated attacks' detection. Various experiments' modifications to optimize classification accuracy were tried, but it was not possible to achieve a better accuracy than 0.00%. Modifications with sampling, discretization and grid optimization of Naive Bayes classifier's settings were performed too. This result shows the impossibility of statistically and behaviorally based NBA (ASNM of AIPS) to detect our obfuscated attacks without any previous information about them.

The next experiment was performed with all kind of training data (including obfuscated attacks). The greedy forward features selection of ASNM set was used in order to maximize performance of the Naive Bayes classifier. The best

TABLE IV: OBFUSCATED/DIRECT ATTACKS' CLASS NEUTRALIZATION

(a) OBFUSCATED ATTACKS CONSIDERED AS LEGITIMATE

Accuracy: 97.72% ± 0.87%	True True	True False	Class precision
Predicted True	156	5	96.89%
Predicted False	13	616	97.93%
Class recall	92.31%	99.19%	

(b) DIRECT ATTACKS CONSIDERED AS LEGITIMATE

Accuracy: 97.71% ± 1.59%	True True	True False	Class precision
Predicted True	156	2	98.73%
Predicted False	17	643	97.42%
Class recall	90.17%	99.69%	

TABLE V: CONFUSION MATRIX OF POLY-NOMINAL CLASSIFICATION EXPERIMENTS

Accuracy: 98.87 +/- 0.99			True occurence													Class precision [%]
			Obfus. attacks				Direct attacks				Legit. traffic					
			Apache	BadBlue	DCOM RPC	Samba	Apache	BadBlue	DCOM RPC	Samba	Apache	BadBlue	DCOM RPC	Samba	Other traffic	
Predicted classification	Obfus. attacks	Apache	72	0	0	0	0	0	0	0	0	0	0	0	0	**100.00**
		BadBlue	0	7	2	0	0	0	0	0	0	0	0	0	0	**77.78**
		DCOM RPC	0	2	6	0	0	0	0	0	0	0	0	0	0	**75.00**
		Samba	1	0	0	8	0	0	0	0	0	0	0	0	0	**88.89**
	Direct attacks	Apache	0	0	0	0	101	0	0	0	0	0	0	0	0	**100.00**
		BadBlue	0	1	0	0	0	4	0	0	0	0	0	0	0	**80.00**
		DCOM RPC	0	0	0	0	0	0	4	0	0	0	0	0	**1**	**80.00**
		Samba	0	0	0	0	0	0	0	20	0	0	0	0	0	**100.00**
	Legit. traffic	Apache	0	0	0	0	0	0	0	0	267	0	0	0	0	**100.00**
		BadBlue	0	0	0	0	0	0	0	0	0	170	1	0	0	**99.42**
		DCOM RPC	0	0	0	0	0	0	0	0	0	0	221	0	0	**100.00**
		Samba	0	0	0	0	0	0	0	0	0	0	0	19	0	**100.00**
		Other traffic	**1**	0	0	0	0	0	0	0	1	0	0	1	66	**95.65**
Class recall [%]			**97.30**	**70.00**	**75.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**99.63**	**100.00**	**99.55**	**95.00**	**98.51**	

classification accuracy with respect to precision is equal to 98.87% (± 0.99%). The structured confusion matrix of this experiment is shown in Table V.

We reached the high inter-class precision rates and recall too. If we would consider obfuscated and direct attacks as one class, there would only be one attack instance classified as legitimate traffic – a true occurence of obfuscated attack on Apache service classified as another traffic class (the field with a black background in the last row of the table). The second situation of an incorrect classification decision[1] respecting the previous consideration is in the case when one legitimate communication was classified as an attack – true occurence of other traffic classified as a DCOM RPC direct attack (the field with a black background in the 7th row of the table). In this situation it is possible that this communication is a sub-part of DCOM RPC attack running on other ports and, therefore, automated expert knowledge processing phase marks this one communication as other traffic. But, on the other hand, the ASNM features extraction process of the AIPS NBA takes into account this fact in the context analysis of a potential DCOM RPC attack and empowers the class recall of the original attack's sub-part.

VII. CONCLUSION

In this paper we presented malicious network traffic obfuscation by tunneling in HTTP and HTTPS protocols. The main reason of supposed malicious traffic obfuscation was the intention of imitating characteristics of legitimate traffic by malicious traffic and to impose low detection capabilities of IDS and NBA. There were used advanced network features in contrast with tunneling obfuscation stated in paper [17]. We analyzed only network services vulnerable to buffer overflow attacks. We managed to find only four publicly available vulnerable network services in contrast to a huge amount of publicly available exploits. Therefore, the results of our experiments are limited to four examined network services. On the other hand, buffer overflow attacks have common characteristics which are very similiar, therefore, the generalization of our results is very possible and it could be part of our next research.

We performed exploitation of chosen services with a direct approach and by employing our proposed obfuscation. We performed legitimate traffic simulations in order to balance traffic class representatives too. All simulations were performed in four scenarios simulating traffic shaping, traffic policing and transmission on an unreliable network channel simulating the real network traffics' conditions.

We demonstrated the successful detection of direct attacks and the obfuscated attacks detection incapability of the SNORT. From the perspective of NBA, we used a statistically and behaviorally based AIPS system and demonstrated the low malicious traffic detection rate of the AIPS trained by direct attacks and legitimate traffic only. Next, we demonstrated the necessity of providing information about obfuscated attacks in the training phase in order to achieve higher classification accuracy of the employed Naive Bayes model. We achieved an accuracy of 97.64% (±0.45%) in bi-nominal classification and an accuracy of 98.87% ±0.99% in poly-nominal classification. The results were gained by 5-cross fold validation and the discretization of attributes into five bins. Another result is indication of different statistical and behavioral characteristics of obfuscated malicious traffic in contrast with direct malicious traffic. The next outcome of our work is to emphasize the necessity of training the NBAs model with divergent obfuscation techniques and their modifications in order to strengthen its

[1]There are another incorrectly classified entries but they keeps the consideration.

detection capabilities.

In future work, we plan to compare the detection properties of ASNM features with other network feature sets (eg. discriminators of A. Moore [9]) using presented tunneling obfuscation with various testing scenarios and a larger dataset of vulnerable network services. We plan to employ new kinds of network attacks' obfuscation techniques, including the PBA [12] to make general NBAs capable of detecting various obfuscated attacks.

Acknowledgment

The work was supported by the EU/Czech IT4Innovations Centre of Excellence project CZ.1.05/1.1.00/02.0070 and the internal BUT projects FIT-S-12-1 and FIT-S-14-2486.

References

[1] Insecure.Org, "How to write Buffer Overflows," Access Date: 1 Dec, 2014. [Online]. Available: http://insecure.org/stf/mudge_buffer_overflow_tutorial.html

[2] A. One, "Smashing the stack for fun and profit," *Phrack magazine*, vol. 7, no. 49, pp. 14–16, 1996.

[3] "Writing buffer overflow exploits – a tutorial for beginners," ECE/CIS labs, University of Delaware, Access Date: 1 Dec, 2014. [Online]. Available: http://www.eecis.udel.edu/bmiller/cis459/2007s/readings/buff-overflow.html

[4] C. Cowan, P. Wagle, C. Pu, S. Beattie, and J. Walpole, "Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade," in *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, vol. 2. IEEE, 2000, pp. 119–129.

[5] M. Barabas, M. Drozd, and P. Hanáček, "Behavioral Signature Generation Using Shadow Honeypot," in *World Academy of Science, Engineering and Technology*, ser. Issue 65, May 2012, Tokyo, Japan, no. 65. World Academy Science Engineering Technology, 2012, pp. 829–833.

[6] I. Homoliak, "Metrics for Intrusion Detection in Network Traffic," Master's thesis, University of Technology Brno, Faculty of Information Technology, Department of Intelligent Systems, 2011, In Slovak.

[7] M. Barabas, I. Homoliak, M. Drozd, and P. Hanacek, "Automated Malware Detection Based on Novel Network Behavioral Signatures," *International Journal of Engineering and Technology*, vol. 5, no. 2, pp. 249–253, 2013.

[8] I. Homoliak, M. Barabas, P. Chmelar, M. Drozd, and P. Hanacek, "ASNM: Advanced Security Network Metrics for Attack Vector Description," in *Proceedings of the 2013 International Conference on Security & Management*. Computer Science Research, Education, and Applications Press, 2013, pp. 350–358.

[9] A. W. Moore, D. Zuev, and M. Crogan, "Discriminators for Use in Flow-based Classification," Technical report, Intel Research, Cambridge, Tech. Rep., 2005.

[10] B. Sangster, T. O'Connor, T. Cook, R. Fanelli, E. Dean, W. J. Adams, C. Morrell, and G. Conti, "Toward Instrumenting Network Warfare Competitions to Generate Labeled Datasets," in *Proc. of the 2nd Workshop on Cyber Security Experimentation and Test (CSET09)*, 2009.

[11] D. Ovsonka, "Network Traffic Obfuscation for IDS Detection Avoidance," Master's thesis, University of Technology Brno, Faculty of Information Technology, Department of Intelligent Systems, 2013, In Slovak.

[12] P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, and W. Lee, "Polymorphic Blending Attacks," in *Proceedings of the 15th USENIX Security Symposium*, 2006, pp. 241–256.

[13] P. Fogla and W. Lee, "Evading Network Anomaly Detection Systems: Formal Reasoning and Practical Techniques," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 59–68.

[14] E. Hjelmvik and W. John, "Breaking and Improving Protocol Obfuscation," *Chalmers University of Technology, Tech. Rep*, vol. 123751, 2010.

[15] B. Cohen, "The BitTorrent protocol specification," 2008.

[16] H. Mohajeri Moghaddam, B. Li, M. Derakhshani, and I. Goldberg, "SkypeMorph: Protocol Obfuscation for Tor Bridges," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 97–108.

[17] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli, "Tunnel Hunter: Detecting Aapplication-layer Tunnels with Statistical Fingerprinting," *Computer Networks*, vol. 53, no. 1, pp. 81–97, 2009.

[18] R. Bar-Yanai, M. Langberg, D. Peleg, and L. Roditty, "Realtime Classification for Encrypted Traffic," in *Experimental Algorithms*. Springer, 2010, pp. 373–385.

[19] Q. Zhang, D. S. Reeves, P. Ning, and S. P. Iyer, "Analyzing network traffic to detect self-decrypting exploit code," in *Proceedings of the 2nd ACM symposium on Information, computer and communications security*. ACM, 2007, pp. 4–12.

[20] "The CLET polymorphism engine," Access Date: 1 Dec, 2014. [Online]. Available: http://www.phrack.org/show.php?p=61&a=9

[21] "The ADMmutate polymorphic engine," Access Date: 1 Dec, 2014. [Online]. Available: http://www.ktwo.ca/ADMmutate-0.8.4.tar.gz

[22] "Metasploit project," Rapid7 Community, Access Date: 1 Dec, 2014. [Online]. Available: http://www.metasploit.org

[23] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine learning for network intrusion detection," in *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 2010, pp. 305–316.

[24] G. Combs *et al.*, "Wireshark-network protocol analyzer," Access Date: 1 Dec, 2014. [Online]. Available: www.wireshark.org

[25] "CVE-2002-0082: Buffer overflow vulnerability of mod_ssl and Apache-SSL," NIST, Access Date: 1 Dec, 2014. [Online]. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2002-0082

[26] "CVE-2007-6377: Stack-based buffer overflow vulnerability in BadBlue." NIST, Access Date: 1 Dec, 2014. [Online]. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2007-6377

[27] "CVE-2003-0352: Buffer overflow in DCOM interface for RPC in MS Windows NT 4.0, 2000, XP and Server 2003," NIST, Access Date: 1 Dec, 2014. [Online]. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2003-0352

[28] "CVE-2003-0201: Buffer overflow in the Samba service." NIST, Access Date: 1 Dec, 2014. [Online]. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2003-0201

[29] "CVE identifiers," MITRE corporation, Access Date: 1 Dec, 2014. [Online]. Available: http://cve.mitre.org/cve/identifiers/index.html

[30] "RapidMiner Studio," Date: 1 Dec, 2014. [Online]. Available: http://rapidminer.com