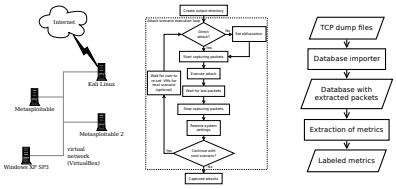
Excel@FIT 2015
http://excel.fit.vutbr.cz

Extension of Behavioral Analysis of Network Traffic Focusing on Attack Detection

Bc. Martin Teknős*



Abstract
This paper is focused on a network behavior analysis (NBA) designed to detect network attacks. It is expected that NBA trained without knowledge of obfuscated attacks will have some difficulties with their detection. Secondly, detection accuracy of NBA should improve by providing obfuscated attacks together with direct attacks to NBA learning process. The goal is to increase accuracy of detection of obfuscated network attacks with NBA and thus improve NBA. This work suggests some obfuscation methods which can be used to evade NBA. A semi-automatic attack tool was implemented. It was used for systematic exploitation and recording of this activity. This was essential for getting enough data before experiments. Acquired data was analyzed with chosen NBA to verify assumptions. Current results of experiments confirmed an assumption that NBA needs knowledge about obfuscated attacks in order to detect them with good accuracy. Detection accuracy of NBA trained without obfuscated attacks can vary a lot, depending on a degree of overfitting. NBA learned with knowledge of all attacks (direct and obfuscated) was able to achieve 99.99% classification accuracy. This work also showed that there is still a lot of further research to be done in this field. There is a need to examine all kinds of attacks and their obfuscations in order to provide various knowledge about attacks to NBA.

Keywords: network behavior analysis — NBA — detection of network attacks — Advanced Security Network Metrics — ASNM — network attack obfuscation — network traffic classification — data mining — machine learning — exploit

Supplementary Material: *N/A*

*xtekno00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

[Motivation] A field of network security is still important and quickly developing. New threats are showing up on daily basis, while older often remain relevant. Relatively new approach with network behavior analysis (NBA) was deployed into battlefield of cybersecurity. NBA system is one type of IDS/IPS[1] technology. It analyses network behavior by watching network traffic in order to recognize attacks with unexpected packet flows. It has some advantages compared to classic approaches. However, this method also has its limitations. One of them is, that attackers can mask their activity in various ways and NBA may have problems with their detection (if it is totally new for it).

[Problem definition] The main goal of this work is to help in improving detection accuracy of NBA

[1] Intrusion Detection/Prevention System

when it is dealing with obfuscated attacks on network services. Firstly, it is necessary to design some attack obfuscation techniques. Secondly, tool for semi-automatic execution of attacks and recording of their activity should be implemented in order to be able to acquire enough data with various obfuscated attacks for later experiments. Thirdly, all data that will be acquired must be processed into dataset. Fourthly, dataset will be used to verify assumptions. Further, gained dataset may be used to learn a classifier's model of NBA to handle obfuscated attacks as well as direct attacks.

[My solution] Achieved results of experiments confirmed assumptions about a need of providing obfuscated attacks together with direct attacks in order to get better classification accuracy with NBA. Without this information, resulting model of NBA may be too specific for direct attacks, thus achieving only low detection accuracy. As a bonus, the tool for semi-automatic execution of attack scenarios was implemented. It proved to be very useful considering, how much time it saved. Additionally, it brought some consistency and systematic approach into process of attacking machines and recording traffic flows. Otherwise, it would be extremely error prone process.

[Contributions] This work showed that there is still a lot of further research to be done in this field. There is a need to examine all kinds of attacks and their obfuscation in order to provide various knowledge about attacks to NBA. Created tool for semi-automatic attacks can be useful in further research.

2. State of the art

Research around NBA concentrated on behavioral signatures. Various sets of metrics which characterize network behavior of malicious traffic were developed. Methods using standard NetFlow proved to be insufficient. Researchers started creating their own metrics that allowed to gain more information and context about analyzed connections. [1]

There are two publicly known examples of metrics which can be used for NBA. Both concentrate on TCP connections that have well defined start and end. On the other hand, UDP protocol is problematic by its very nature. A brief introduction to mentioned metrics:

- Moore *et al.* [2] presented discriminators that provide many features for description of network packet flows. They have done some more research around discriminators and Internet traffic classification in their following papers [3, 4].
- Advanced Security Network Metrics (ASNM) were first published in thesis [5]. Later, Homoliak *et al.* [1] published updated version. They defined method for generating signatures of network behavior from set of ASNM. They consist of 167 metrics divided into 5 categories according to their nature. Those categories are: Statistical, Dynamic, Localization, Distributed and Behavioral.

Corona *et al.* [6] systematically categorized attacks against IDSs. They defined six main goals of attack on IDSs: evasion, overstimulation, poisoning, denial of service (DoS), response hijacking and reverse engineering. Those attacks can exploit various vulnerabilities of IDSs. Authors also suggest a categorization of vulnerabilities of IDSs.

Homoliak *et al.* [7, 8] experimented with a network attack obfuscation hidden into HTTP(S) traffic. They examine detection capabilities of obfuscated network buffer overflow attacks with NIDS[2] (Snort) and NBA (AIPS[3]). Obfuscation was done by tunneling malicious traffic in HTTP(S) protocols. They wanted to simulate typical properties of legitimate HTTP network traffic. They presented that Snort was able to detect direct attacks. However, it did not detect any obfuscated attack. With classifier trained without obfuscated attacks, they achieved only 0.00% accuracy. This showed that a behaviorally and statistically based NBA is not capable of detecting obfuscated attacks without their previous knowledge. Then they included obfuscated attacks into training data. With that, they achieved 97.64% $\pm$ 0,45% accuracy in case of binominal classification and 98,87% $\pm$ 0,99% in case of polynomial classification. Further, they emphasize the importance of training a behaviorally and statistically based NBA with different obfuscation techniques in order to be able to detect such attacks.

Fogla *et al.* [10] designed a new subclass of mimicry attacks. They call it polymorphic blending attacks (PBA). Those attacks can effectively avoid detection by a byte frequency-based network anomaly detection IDS (eg. PAYL). This kind of attack carefully modifies statistics of its packets in order to copy legitimate network traffic. Authors demonstrated effectiveness of this approach with PBA attacks on PAYL.

3. Approach

In order to achieve the set goal, it was necessary to do a lot of small steps. Core parts of this process are presented in the following subsections.

[2] Network IDS

[3] This system, called *Automated Intrusion Prevention System* (AIPS), was described in the paper [9]. It is using ASNM and as a source of expert knowledge, it is using honeypots.

Table 1. Used obfuscation techniques.

Group	Method
spreading packets over time	constant delay of 1s
	constant delay of 8s
	5s delay with ± 2.5s variation (normal distribution) and 25% correlation
packet loss	25% packet loss
packet corruption	25% packet corruption
	35% packet corruption
	35% packet corruption with 25% correlation
packet duplication	5% packet duplication
packet reordering	25% packets reordered (sent with 10ms delay) with 50% correlation
	50% packets reordered (sent with 10ms delay) with 50% correlation
MTU modification	MTU 1000
	MTU 750
	MTU 500
	MTU 250

3.1 The methods suggested for evading a behavioral analysis of network communications

Following methods were suggested based on a plan to use the ASNM, consultations with my supervisor and with consideration of what was really feasible in my conditions. Some techniques were not included, mainly due to type of planed deployment (see section 3.2). As an example, a DoS attack on NBA would require real physically deployed NBA. In addition, ASNM [1] (and discriminators [2]) examine only packet headers, thus attacks that would obfuscate payload (eg. PBA [10]) would be almost pointless.[4] Finally, Corona *et al.* [6] showed number of attack vectors on IDSs, but many of them were not relevant due to type of deployment.

A summary of used obfuscation techniques is in the table 1. Those values were obtained empirically, ie. by trying different values, until attacks were not successful due to network problems. A whole point of those obfuscation techniques is to try and modify packet flows in various ways. NBA (ASNM) is examining diverse statistics and this obfuscation could bring some confusion for its detection model. An Exploit packet with an exploit content is usually a big one and those techniques may change it.

3.2 Attack scenarios

Attacks scenarios were carried out only in a virtual network (see the figure 1) with VirtalBox virtual machines (VMs). It was mainly due to legal and practical

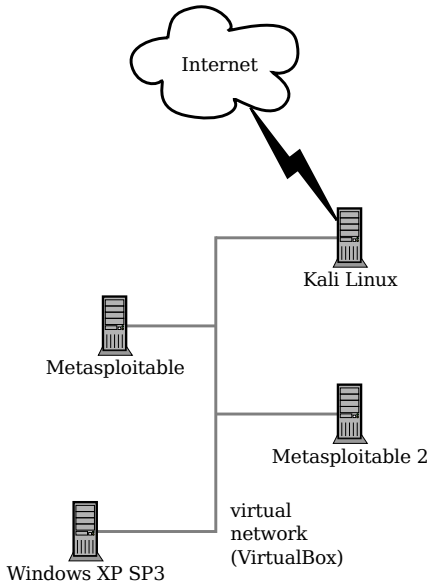


Figure 1. A Virtual network where attack scenarios were carried out.

issues of executing network attacks over the Internet. The VM with Kali Linux 1.1.0 was used as attacker's machine. Vulnerable machines that represented targets were running Metasploitable, Metasploitable 2[5] and Windows XP SP3 operating systems respectively.

Focus was on initial attacks. That is, attacker does not have any kind of access to a trusted system (eg. as it was done with a previously compromised machine used for HTTPS tunneling in [7, 8]).

Network attacks on various network services were performed. It wasn't particularly biased towards any kind of attack. It was more of a "anything that gets me into system" thing. An attacker usually thinks this way anyway. He doesn't care about complexity of attack, but rather if it enables him to meet his goal. A brief description of used services and vulnerabilities:

- Apache Tomcat 5.5 – Firstly, a simple dictionary attack. Secondly, a payload execution on Tomcat server (with acquired credentials) that have an exposed "manager" application.
- Microsoft SQL Server 2005 – Firstly, a plain dictionary attack. Secondly, execution of an arbitrary payload on a Microsoft SQL Server using a "xp_cmdshell" stored procedure.
- Samba 3.0.20-Debian – Exploiting a command execution vulnerability in Samba when using a non-default "username map script" configu-

[4] NBA would not see a difference between a direct and obfuscated attack.

[5] The Metasploitable virtual machine is an intentionally vulnerable version of Ubuntu Linux designed for testing security tools and demonstrating common vulnerabilities.

ration option. By specifying a username containing shell meta characters, attackers can execute arbitrary commands. No authentication is needed.

- Server service (Windows XP SP3) – This service allows remote attackers to execute arbitrary code via a crafted RPC request that triggers an overflow during path canonicalization.

3.3 The tool for semi-automatic attack execution

The tool called *exploitator* was implemented for semi-automatic attack execution and recording of this activity. It was written in Python. It is using metasploit framework and its `msfconsole` for automatic attacks. A user must provide (manually tested) resource file which contains msfconsole commands. Exploitator needs it for an automatic exploitation. A user can choose some other parameters, but it is out of scope of this paper. A simplified exploitator execution flow is shown in the figure 2. Packets are captured in a subprocess with `tcpdump`. It captures a TCP traffic between an attacker and attacked machine(s). An attack is launched in second subprocess. After an attack is executed, a program waits some time (up to several minutes), before stopping packet capture for current attack scenario. It needs to wait, because otherwise there would be incomplete TCP connections, that would not make it into a dataset.[6] The program loops and performs every chosen attack scenario, until there is none left. This loop is shown as dashed rectangle in the figure 2. If there is to be a next attack scenario, program waits for a user to confirm that she restored attacked machine(s) to a pre-exploitation state. This waiting is optional, but highly recommended, considering some attacks might have been (semi-)destructive. If a user ignores it, some attacks could fail and/or provide inconsistent results. Otherwise, the program exits and an output directory contains a subdirectory for each executed attack scenario. It is in a form that can be directly used with post-exploitation tools (see 3.4). Therefore, next process of extracting ASNM is as straightforward as possible.

3.4 Obtaining a dataset for experiments

A set of post-exploitation tools was provided by my supervisor. It was used on pcap files[7] from realized attacks (and legitimate traffic) to get dataset with ASNM for experiments. Thanks to some careful preparation

[6] Tools for extracting metrics do not work with unfinished (not closed) TCP connections. Thus, they do not make it into extracted metrics.

[7] packets captured with tcpdump

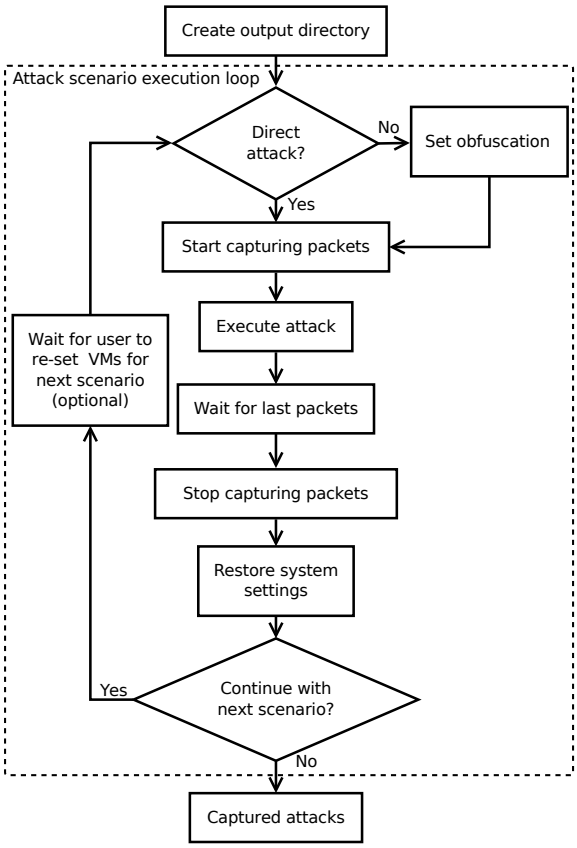


Figure 2. Simplified exploitator execution flow.

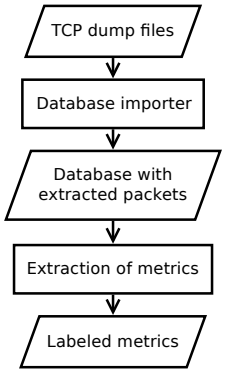


Figure 3. Process of extracting labeled metrics from acquired pcap files.

and a way that the exploitator works, it was a straightforward process (see the figure 3).

Obtained dataset consisted of 5663 malicious connections (see the table 2) and 2619 legitimate connections. Legitimate traffic was represented by a general usage of Internet (Web, email,...) and a legitimate communication between VMs (see the figure 1). A command line tool netem [11] was used to simulate delay, packet loss, packet corruption, packet duplication and packet reordering. However, it was far from extreme values that were used for obfuscation. Traffic class dis-

Table 2. Traffic class distribution for malicious TCP connections.

Service & attack	Direct attacks	Obfuscated attacks
Apache Tomcat mgr_deploy	12	158
Apache Tomcat mgr_login	256	3348
Microsoft SQL Server 2005 login	23	321
Microsoft SQL Server 2005 xp_cmdshell	22	690
Server service ms08_067_netapi	84	681
Samba usermap_script	4	64
Total	401	5262

tribution for legitimate TCP connections is as follows:

- a general usage of Internet – 2221,
- a general usage of Internet (netem) – 382 and
- a communication between VMs – 16.

4. Experiments

This work concentrated on detection of any attack, regardless of its nature (ie. direct or obfuscated). Thus, only binominal classification was practiced. This is more realistic for a real NBA deployed in the field. Companies want to detect attacks and let legitimate traffic through. They usually do not care about exact nature of an attack.[8]

Main ideas behind presented experiments are discussed in the following. It is assumed, that NBA trained without knowledge of obfuscated attacks will have trouble with their detection. On the other hand, if NBA was to get obfuscated attacks along with directs attacks for its training process, it should get much better at their detection. Some works already showed that this is true for attacks obfuscated by tunneling them withing HTTPS traffic [7, 8].

Data mining tool RapidMiner Studio was used for all experiments. Classification was done with a Naive Bayes (Kernel) operator. A legitimate and malicious communication was described by only 15 features (879 features was available). Those same features were used in [7, 8] as well.

In one set of experiments, only direct attacks and legitimate traffic were used to train a classifier and then the trained model was tested on whole dataset. In tables 3 and 4 you can see, how different parameters of classifier can lead to bad and good results respectively.

In another experiment, all data were given to a classifier that was tested with a 5-fold cross-validation (see the table 5). This experiment shows, how more data about attacks can improve accuracy of classification to almost 100.00%.

From those experiments, it is clear to see that there are some properties of obfuscated attacks which are different from direct attacks. Therefore, one can train a classification model for NBA without obfuscated attacks, but it may not generalize well (see tables 3). Without access to obfuscated attacks, one can not properly test, whether a created model can generalize well enough to detect those attacks. In summary, it is highly recommended to provide an information about obfuscated attacks to a process of learning a classification model for NBA.

[8] Even if they want to examine detected attack (eg. in order to fix exploited vulnerabilities) they do (not) do it regardless of, what kind of attack it is.

Table 3. Example of bad confusion matrices.

(a) Confusion matrix for a classifier trained without obfuscated attacks.

Accuracy: 98.48% ±0.60%	True legitimate	True attack	Class precision
Pred. legitimate	2512	44	98.28%
Pred. attack	0	338	100.00%
Class recall	100.00%	88.48%	

(b) Confusion matrix for a classifier tested on whole dataset.

Accuracy: 68.63%	True legitimate	True attack	Class precision
Pred. legitimate	2619	2459	51.58%
Pred. attack	0	2761	100.00%
Class recall	100.00%	52.89%	

Table 4. Example of good confusion matrices.

(a) Confusion matrix for a classifier trained without obfuscated attacks.

Accuracy: 99.97% ±0.07%	True legitimate	True attack	Class precision
Pred. legitimate	2619	1	99.96%
Pred. attack	0	400	100.00%
Class recall	100.00%	99.75%	

(b) Confusion matrix for a classifier tested on whole dataset.

Accuracy: 96.50%	True legitimate	True attack	Class precision
Pred. legitimate	2619	290	90.03%
Pred. attack	0	5373	100.00%
Class recall	100.00%	94.88%	

Table 5. Confusion matrix for a classifier tested on whole dataset with a 5-fold cross-validation.

Accuracy: 99.99% ±0.02%	**True legitimate**	**True attack**	**Class precision**
Pred. legitimate	2618	0	100.00%
Pred. attack	1	5663	99.98%
Class recall	99.96%	100.00%	

5. Conclusions

[Paper Summary] Methods designed to bypass NBA were presented. The tool (called exploitator) for semi-automatic network attacks obfuscation and recording of those attacks was created. Data for experiments was obtained with this tool. Experiments confirmed the assumption that it is necessary to provide obfuscated attacks to NBA in order to improve its detection accuracy.

[Highlights of Results] It was shown that NBA trained without knowledge of obfuscated attacks can be misguided under bad conditions. On the other hand, NBA tested with knowledge of obfuscated attacks achieved 99.99% classification accuracy. The exploitator proved to be very useful. It enabled to focus on a systematic approach of data collection while limiting errors. A manual data collection in this scale would be extremely difficult and error prone.

[Paper Contributions] This work built on works of other researchers. It showed that with different obfuscation methods, it is really useful to have a dataset with variety of attacks for NBA. Otherwise, NBA can have problems with some (eg. obfuscated) attacks. Created exploitator tool was really helpful and I hope, it can help other researchers as well[9].

[Future Work] It the future, I am going to experiment with various combinations of obfuscation techniques. Those experiments can bring some new interesting results. I plan to do some payload (exploit content) modifications too. I expect that this will not present problem for NBA, as it does not really examine payload content. However, it can be interesting to confirm this assumption. As a bonus, I want to find out which obfuscation techniques work best in evading NBA. This will be done on final dataset. This work is part of my thesis and more details will be published there, later this year.

Acknowledgments

I would like to thank to my supervisor Ivan Homoliak for his support, comments and suggestions which helped to improve my work and this paper.

[9] It will be released as part of my thesis in the next month.

References

[1] I Homoliak, M Barabas, P Chmelař, M Drozd, and P Hanáček. ASNM: Advanced Security Network Metrics for Attack Vector.

[2] Andrew Moore, Denis Zuev, and Michael Crogan. *Discriminators for use in flow-based classification*. Queen Mary and Westfield College, Department of Computer Science, 2005.

[3] Andrew W Moore and Konstantina Papagiannaki. Toward the accurate identification of network applications. In *Passive and Active Network Measurement*, pages 41–54. Springer, 2005.

[4] Tom Auld, Andrew W Moore, and Stephen F Gull. Bayesian neural networks for internet traffic classification. *Neural Networks, IEEE Transactions on*, 18(1):223–239, 2007.

[5] HOMOLIAK Ivan. Metriky pro detekci útoků v síťovém provozu. Master's thesis, 2012.

[6] Igino Corona, Giorgio Giacinto, and Fabio Roli. Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues . *Information Sciences*, 239:201–225, 2013.

[7] Ivan Homoliak, Daniel Ovšonka, Matěj Grégr, and Petr Hanáček. NBA of Obfuscated Network Vulnerabilities' Exploitation Hidden into HTTPS Traffic. In *Proceedings of International Conference for Internet Technology and Secured Transactions (ICITST-2014)*, pages 311–318. IEEE Computer Society, 2014.

[8] Ivan Homoliak, Daniel Ovšonka, Karel Koranda, and Petr Hanáček. Characteristics of Buffer Overflow Attacks Tunneled in HTTP Traffic. In *International Carnahan Conference on Security Technology*, 48th Annual International Carnahan Conference on Security Technology, pages 188–193. IEEE Computer Society, 2014.

[9] Maroš Barabas, Michal Drozd, and Petr Hanáček. Behavioral signature generation using shadow honeypot. *World Academy Science Engineering Technology*, 2012:829–833, 2012.

[10] Prahlad Fogla, Monirul Sharif, Roberto Perdisci, Oleg Kolesnikov, and Wenke Lee. Polymorphic blending attacks. In *Proceedings of the 15th conference on USENIX Security Symposium-Volume 15*, pages 241–256. USENIX Association, 2006.

[11] Stephen Hemminger et al. Network emulation with NetEm. In *Linux conf au*, pages 18–23. Citeseer, 2005.