Behavioral signature generation using shadow honeypot

Maros Barabas, Michal Drozd, Petr Hanacek

***Abstract*—** A novel behavioral detection framework is proposed to detect zero day buffer overflow vulnerabilities (based on network behavioral signatures) using zero-day exploits, instead of the signature-based or anomaly-based detection solutions currently available for IDPS techniques. At first we present the detection model that uses shadow honeypot. Our system is used for the online processing of network attacks and generating a behavior detection profile. The detection profile represents the dataset of 112 types of metrics describing the exact behavior of malware in the network. In this paper we present the examples of generating behavioral signatures for two attacks – a buffer overflow exploit on FTP server and well known Conficker worm. We demonstrated the visualization of important aspects by showing the differences between valid behavior and the attacks. Based on these metrics we can detect attacks with a very high probability of success, the process of detection is however, very expensive.

***Keywords*—**behavioral signatures, metrics, network, security design.

I. Introduction

MALWARE detection based on behavioral analysis is a method that can be used to effectively defend systems against growing trend of highly sophisticated and specialized malware against which standard NIDS and ADS techniques are little or completely ineffective [1]. Behavioral analysis is already used for malware detection on the operating system level for different platforms. Behavioral analysis of network flow is more demanding on computing resources as well as false-positive elimination. Our approach focuses on the possibility of using behavioral signatures based on detection metrics that could be effectively distributed and mutually optimized.

This short paper introduces the novel Automated Intrusion Prevention System (AIPS) which uses honeypot systems for the detection of new attacks and the automatic generation of behavioral signatures based on network flow metrics. While the long-term objective of AIPS is to address all types of attacks and aspects of intrusion detection, in this paper we present only the detection technique and the process of generation of the behavioral signature upon buffer overflow attacks [2].

The paper is organized as follows. Section 2 discusses related work in a network intrusion detection and signature generation. In Section 3 we describe the detection model and signature generation technique. Section 4 presents the metrics definition used for training detectors. Section 5 presents the results and evaluations of two attack examples in comparison with valid behavior and section 6 contains the conclusion of this paper.

M. Barabas is a PhD student at Faculty of Information Technology, Brno University of Technology, Czech republic (e-mail: ibarabas@fit.vutbr.cz).

M. Drozd is a PhD student at Faculty of Information Technology, Brno University of Technology, Czech republic (e-mail: idrozd@fit.vutbr.cz).

P. Hanacek is an associate professor at Faculty of Information Technology, Department of Intelligent Systems, Brno University of Technology, Czech republic (e-mail: hanacek@fit.vutbr.cz).

II. State of the Art

There are many signature-based IDS and statistic ADS systems that fail on detecting unknown or zero-day attacks and new variants of old exploits. Thus a new generation techniques and systems based on anomaly detection appeared. The Anomaly Detection systems model the normal or expected behavior in a system, and detect interest deviations and differences that may indicate a security breach or an attempted attack [3]. There are two types of systems based on anomaly detection: those that work with a predefined specification (or set of rules) of what is regarded as normal behavior and others that learn the behavior under normal operation.

In [4] authors introduced the MINDS detection system which uses data mining techniques to automate the detection process. This system works with netflow [5] data with 10-minute data windows and time and connection features that represent complex metrics upon netflow data. In the case study section (section 5) we show that a full network dump is needed to model the connection to represent some attacks for further detection. That extensive netflow cannot be described. The second problem of this approach is a need of a human expert that has to look at the output of the system to determine if the detected connection is actually an attack. But this approach can have suitable results in detection of unknown threats and some malware morphisms.

Our approach is similar to those systems that reconstruct the network packets and extract features that describe the higher level interactions between the end hosts like MADAMID [6], Bro [7], EMERALD [8], STAT [9], ALAD [10] etc. The extracted features – for example session duration time, service type, bytes transferred and so forth – are regarded as higher level, temporally ordered features not discernible by inspecting only the packet content.

Approach presented in this paper uses much lower level of abstraction and focuses on the processing of generated metrics for the attack description.

III. AIPS Detection model

We developed a new system for automate intrusion prevention (AIPS) that focuses on different subset of behaviors of anomaly detection techniques that is common for most existing detectors. Our system does not specify what a normal

behavior is, but what seems to be abnormal or is very likely an attack. In this approach we need an expert knowledge that defines what the abnormal behavior is. For this purpose we use Shadow Honeypot systems for the detection of new threads. We primary focused on Buffer Overflow attacks. These high-interaction honeypots simulate various operation systems with many vulnerable services that attract attackers. There is a tcpdump listening on the network interface and sniffing the communication on the honeypot. The next parts of the AIPS system are communication and metric extractor that work upon tcpdump data. These two parts are used to extract metrics from the communication for further analysis. The last but not least parts of the system are IDPS with metric dataset used by IDS learning algorithms.

The schema of AIPS system is shown in the fig. 1: Detection model. We can see three honeypot systems connected to the network. These systems are Argos honeypots [11] emulating different operating systems with various vulnerable services. In the real deployment we assume that similar honeypot functionality will be implemented directly to the virtualized system with capability of detection unknown or zero-day buffer overflow attacks [12].

A. Principle of detection

In case that an attacker attacks this vulnerable system and causes buffer overflow incident, his attempt is detected and recorded by honeypot in real time. The dump of the communication from tcpdump with the timestamp of the attack and the actual packet that caused the buffer overflow (from Argos) are sent to the communication extractor where all data are parsed. From this set of data the extractor parse only relevant packets that are further sent to the metric extractor system. Metric extractor creates dataset of metrics for this specific attack and sends all relevant information with dataset to the database. The metric dataset is then further distributed from database to the IDPS systems for learning process (artificial intelligence, data mining algorithms, etc.). We assume that the whole process could run in real time (the performance testing is planned in the near future).

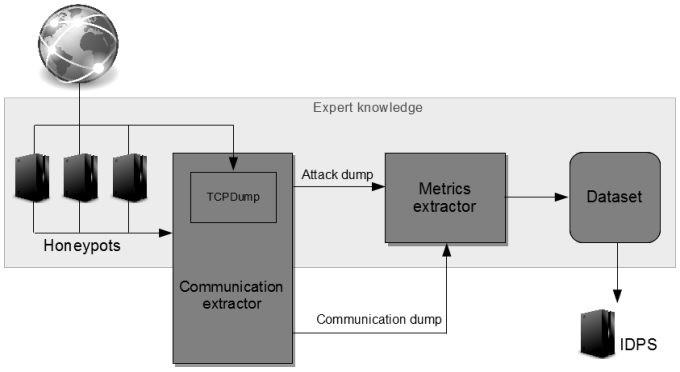

Fig. 1 Detection model

The part of the system with honeypots creates a set of expert knowledge. This set is expanded only when the honeypot detects a new attack so this new entry is apriority set as true positive and is added to the knowledge set. In the next section we will describe in more detail how is honeypot connected with the tcpdump and other detection systems thru a database.

B. Database scheme

In the fig. 2 is shown an important part of database schema used for storing the incident data from various subsystems. This part consists of four primary classes. First class “aips” represents the bridge between subsystems. It connects the Argos system, tcpdump and other detection systems like snort IDS. Argos is represented by classes “incidents” and “exploit packets”.

In case that a new attack is detected, then this attack is recorded as incident with unique ID, timestamp and other properties. The honeypot also saves the packet that causes the buffer overflow and adds it to the incident data. The system that manipulates with tcpdump data will record whole TCP traffic associated with the incident. AIPS system actually works only with the TCP communication, other protocols of third and fourth layer will be implemented in the future.

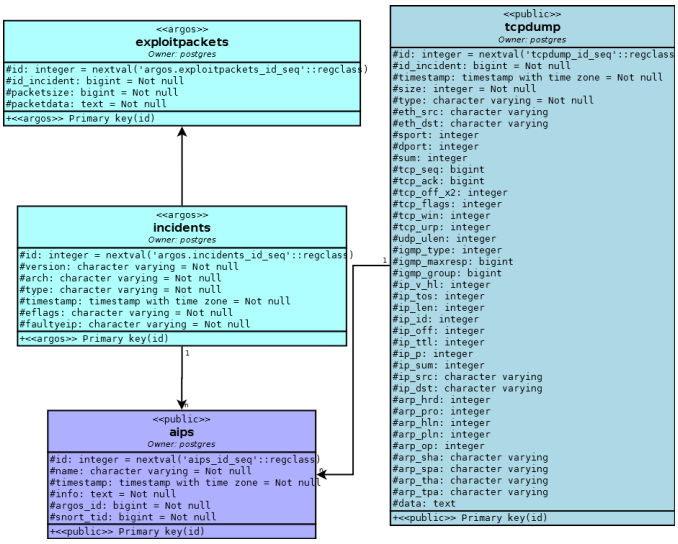

Fig. 2 Database model

IV. METRIC DEFINITION

The detection model described in previous section records detailed network flow dumps which can be used for automated generation of metrics that describe properties, process and behavior of the attack. By using these metrics we are able to unambiguously identify the attack.

For this purpose the number of measurable metrics is defined to be able to describe properties of detected attack not upon the fingerprint of common signature, but based on its behavior – behavioral signature. Behavioral metrics are in a limited extent used in commercial ADS (A-NIDS) or NBA systems for intrusion detection. However they are not used for creation of portable detection profiles. Detection behavioral metrics were described here [1], nevertheless they were not suitable for describing malware but for the detection of network attacks such as port scan, different types of DoS attacks or as existing variant of ping tools. To a certain extent a similar principle is used in, nowadays obsolete, KDDCUP 99 [13] which was created with a much higher

abstraction level. This model already worked with compromised system and information from the honeypot such as an attacker's access to shell, the escalation of privileges from local to root etc.

Our goal was to define such metrics that can be used for detailed description of malware behavior and its behavioral characteristics and features during the attack in network and transport layer.

112 unique metrics were defined on the whole. About a one quarter of them is represented by vector set describing the attack in time and data axis with various dependencies. The individual metrics that make up the behavioral signature are divided according to their nature into five categories (Fig. 3).

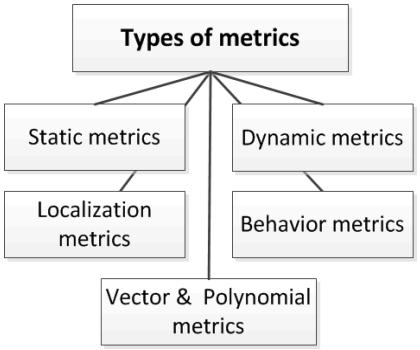

Fig. 3 Types of detection metrics

A. Static metrics

Static metrics define the attack properties from the static events point of view, such as amounts of data, the number of flows, the number of ports, the number of resources in defined flow/event. It was defined 49 unique static metrics.

B. Dynamic metrics

Dynamic metrics represent dynamic network behavior such as speed, number of bytes/packets per second in the outbound and inbound traffic etc. changing in the timeline.
It was defined 30 unique dynamic metrics.

C. Localization metrics

Localization metrics are used to specify the position of sources and trace of the attack. Their aim is to provide the arguments in decision-making process of the data mining engine. 9 localization metrics were defined.

D. Behavior metrics

Behavior metrics is a set of metrics based on the description of the properties directly associated with the attack behavior. Examples include legal or illegal connection closing, number of flows at defined time intervals, polynomial approximation of the length of packets, polynomial approximation of the sum of packets and similar information that are directly related to the exploitation of vulnerable service. This includes also the parallel creation of new services, periodic communication or the change of the profile in terms of ADS. Behavior metrics were defined 9.

E. Vector and Polynomial metrics

Vector metric is defined as an ordered n-tuple. Each value represents the current state of monitored function (the amount of outgoing and incoming data, the size of outbound and inbound packets) per unit time (sampling frequency is 1ms, 5ms, 10ms, 30ms, 50ms and 1s) in the measured network flow. So the number of individual members in n-tuples is not the same and is dependent on sampling frequency and the duration of the measured flow.

Polynomial metrics are defined as polynomial approximation of the length of packets and polynomial approximation of the sum of packets.

There were defined 22 of these metrics.

The final dataset consist of all 112 metrics mentioned in previous subsections. Each metric represents a value in a form of number, polynomial and vector (time-dependent values). In all metrics the statistical functions such as mean, median, mode, the sum etc. are used.

V. Case Study

In this section we show the use of defined metrics in the behavioral analysis of the network flow on two reference examples of buffer overflow attack. First example is an attack that exploits stack buffer overflow in MKD command of FTP server [14]. The second one is well known Conficker worm that exploits parsing flaw in the path canonicalization code of NetAPI32.dll through the Server Service [15].

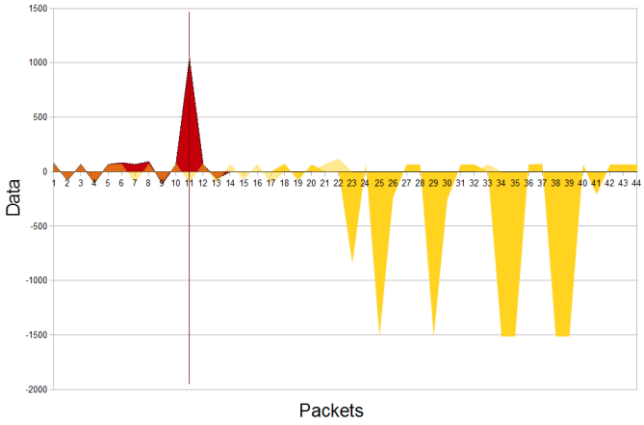

Fig. 4 FTP attack and valid connection

In the fig. 4 are presented inbound and outbound packets of the valid connection and connection that represents an attack on the server. The bordered line filled with red color is the attack communication, with yellow color is filled the flow of the valid communication. The orange parts of the graph are the parts of communication identical for both connections. Yellow flows are the incoming and outgoing data parts, in this case it represent the downloading of files. Under the x axis there is outbound part and above the x axis there is inbound part of communication. By the red vertical line is marked the packet that caused the buffer overflow. This example shows a human-friendly way how to detect buffer overflow attack by occurrence of specific packet in the communication (in this

case specific by size and location). In case that buffer overflow occurred in the first part of communication – in the authentication part – the detection is instantaneous. In case that the buffer overflow packet is injected beyond the legitimate initialization part (for example by anonymous account) the detection is more complicated and other metrics have to be used.

In the next three graphs is presented the time analysis of the same attack as was mentioned before and is shown that common IDS systems using time analysis based on higher abstraction of the communication (several seconds) are not able to detect an incident that is caused in very low time interval (milliseconds), for example by specialized malware.

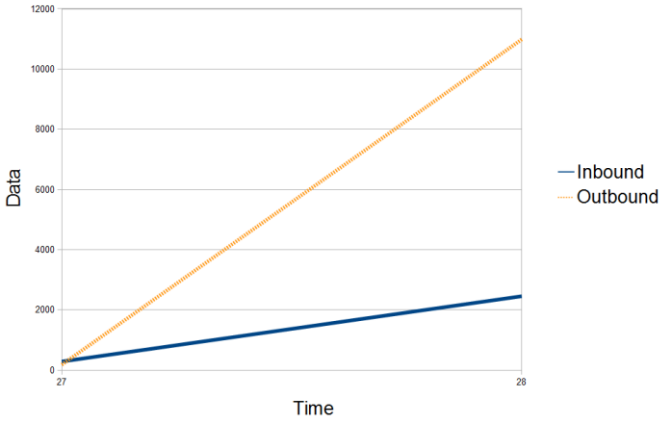

Fig. 5 Inbound and outbound packets in 1s interval metrics

On the fig. 5 is shown the time analysis of communication in second level granularity. This granularity is not able to show the attack peak because the injection of exploit packet occurred within the time interval that includes the data segment of the communication. On the fig. 6 we can see the same attack with the granularity level of tenths of seconds where the exploit incoming packet is marked by vertical red line. The fig. 7 illustrates the communication with the granularity level of milliseconds where the exploit packet (marked with red vertical line as well) can be better differentiated from other communication.

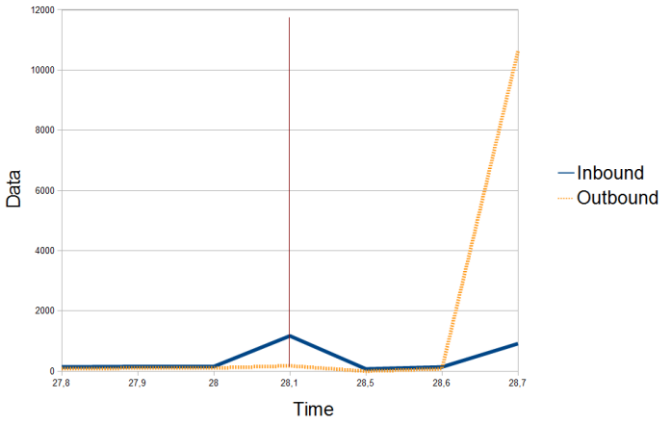

Fig. 6 Inbound and outbound packets in 10-1s interval metrics

As we can see in this example, a very high abstraction or too high granularity of communication analysis can lead to higher false-negative rate.

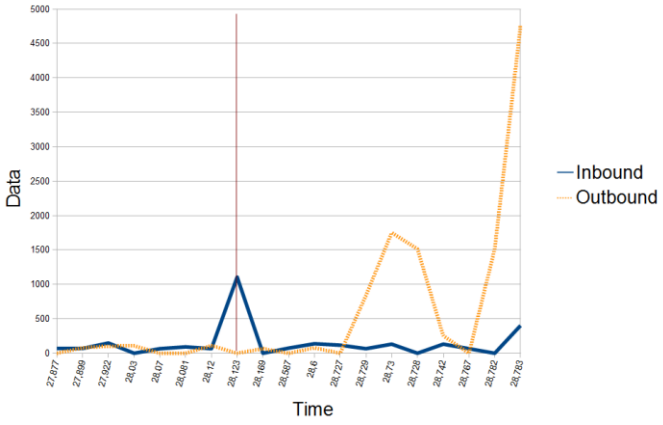

Fig. 7 Inbound and outbound packets in 10-3s interval metrics

In the second example we show the valid communication and the attack on IIS server on port 445 by Conficker worm, which exploits MS08-067 vulnerability in Server service. In the fig. 8 is shown the communication that is divided into two parts. There is a valid communication with the check of the operating system version and service implementation in the left side and the exploitation by the worm on the right side. The picture illustrates the way of how to use the "right" metrics for detection of possible exploitation. The packet which carries the exploit data is marked with the red circle. From the graph it is evident that the last two peaks on the server (last two local maxims of data-inbound communication) can be replaced by other type of valid communication with even higher data size and still it can be a valid behavior and with common metrics it could be detected as malicious packets. We can't say for sure that if any packet exceeds the data threshold it is malicious. The question is how we can detect this attack.

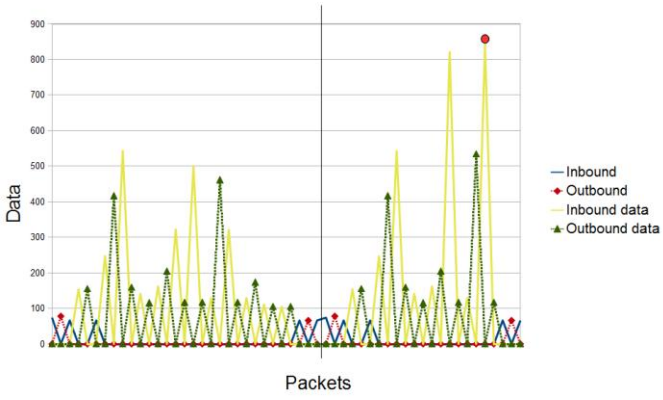

Fig. 8 Conficker check and attack behavior

We can see that in a certain cases it is impossible to determine whether this anomaly is the malicious one or not without increasing false-positive rate. In these situations it is possible to apply more different metrics that could characterize the communication in a more complex way and then we can

determine if the communication is valid or if it is an attack. The example of solution in the situation from fig. 8 could be a detection if a new dynamic port has been opened during or after the suspicious packets (the connection was closed) or the case in which attacked process has been replaced by a new one (for example shell) and parameters of the communication have changed or if exploitation caused the process to crash or the communication is not ended properly (missing FIN packets) and then the attack can be detected.

During the experiments with various attacks on honeypot systems with implementation of AIPS we used other available systems for detection of malicious behavior like Snort IDS. In some tested cases of attacks on honeypots these systems couldn't detect the attack as was described in this section.

VI. Conclusion

This short paper shows the first observation and results of the project focused on the behavioral description of network communication of malware abusing the buffer overflow vulnerability.

We have provided a way of detecting zero-day attacks that combines traditional methods based on extensive knowledge of attack signatures and the generation of signatures based on characterization of network flow and honeypot systems representing the expert knowledge systems.

The model for generating the behavioral structures and description of metrics characterizing the malware behavior were presented. On two case study examples were shown the principles of describing the buffer overflow attacks and possible ways of their detection. The first experiment results show that the method is effective with minimal impact on false positives. The model assumes the expertise knowledge provided by the honeypot systems. For online system deployment a relatively large computational resources are needed because of the complexity of the proposed metrics and low abstraction. These findings are still subject of further study and will be presented in a short time.

In the future we plan to check the effectiveness of each metric using genetic algorithms, optimization of detection sets and processing of individual metric by the agent system capable to mutually communicate the results to increase the efficiency of signature generated.

One of the interesting issues that were found during the tests is the detection of unknown attacks misusing the old vulnerability (MS08-067) which were not recognized by IDS and which performed the effective exploitation. During the three days, when this system was exposed to the Internet, 68 various undetected attempts to abuse the Microsft-ds/tcp 445 service were detected.

Acknowledgment

This project has been realized with a financial support from the Czech Republic state budget through the Ministry of Industry and Trade by the research plan FR-TI1/037. This work was partially supported by the research plan MSM0021630528.

References

[1] Garcia-Teodoro, P., Díaz-Verdejo, J. E., Maciá Fernández, G., Vázquez, E., Anomaly-based network intrusion detection: Techniques, systems and challenges", p. 18-28, 2009.

[2] C. Cowan, P. Wagle, C. Pu, S. Beattie, J. Walpole, BufferOverflows: Attacks and Defenses for the Vulnerability of the Decade, Oasis, p.227, Foundations of Intrusion Tolerant Systems (OASIS'03) 2003.

[3] Ke Wang, Salvatore J. Stolfo, Anomalous Payload-Based Network Intrusion Detection", 2004.

[4] L. Ertoz, E. Eilertson, A. Lazarevic, P.-Ning Tan, P. Dokas, V. Kumar, J. Srivastava, Detection and Summarization of Novel Network Attacks Using Data Mining", 2004.

[5] "NetFlow", Cisco Systems, Inc, 2011, URL: www.cisco.com/go/netflow.

[6] W. Lee and S. Stolfo, "A Framework for Constructing Features and Models for Intrusion Detection Systems", ACM Transactions on Information and System Security, 3(4), November 2000.

[7] M. Mahoney, P. K. Chan, "An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection", RAID 2003, 220-237.

[8] P. Porras and P. Neumann, "EMERALD: Event Monitoring Enabled Responses to Anomalous Live Disturbances", National Information Systems Security Conference, 1997.

[9] G. Vigna and R. Kemmerer, "NetSTAT: A Network-based intrusion detection approach", Computer Security Application Conference, 1998.

[10] M. Mahoney, P. K. Chan, "Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks", Proc. SIGKDD 2002, 376-385.

[11] G. Portokalidis, A. Slowinska, H. Bos, "Argos: an Emulator for Fingerprinting Zero-Day Attacks", in Proc. ACM SIGOPSEUROSYS'2006, 2006.

[12] J. Berg, E. Teran, S. Stover, "Investigating Argos", an Article in USENIX Magazine: ;login, 2008.

[13] KDD Cup 1999, October 2007, URL: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[14] Stack-based buffer overflow in CesarFTP 0.99g, , URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=2006-2961.

[15] Server Service Vulnerability, URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=2008-4250.