

RAID2006

Poster submissions

September 13, 2006

From Exploits to Vulnerabilities: Challenge for IDS appliances

Proneet Biswas (pbiswas@ipolicynetworks.com), IPolicy Networks Inc.

In the year 2005, there were two distinct worm scenarios which shared the limelight — The MYTOB family which exploited the Microsoft LSASS vulnerability and had more than 40 offspring, each emerging within short time intervals. The second culprit was the Zotob worm which emerged within 5 days of the release of Microsoft PNP vulnerability highlighting the decreasing incubation period for worms. These cases reconfirmed the challenge which IPS vendors have been facing— move from network exploit detection to generic network vulnerability detection. Yet, network vulnerability detection is a daunting technology by itself.

Desktop versus Servers — The new vulnerabilities target the software hosted on laptops/desktop and not server machines. What this does is redefine the perimeter of protection from critical servers to include the whole network and thus IPS deployed at the perimeter will no longer suffice.

Client/Server Software — On the software side, this implies that the target is no longer web/ftp server but soft targets like P2P, IM, browsers, music players. These software packages incorporate complex application/file protocols requiring a different level of packet inspection. It also changes the performance tests as the average packet size is no longer restricted to IMIX.

Response Packet Inspection — Many of these exploits would require the client to initiate an innocuous request to a site which would respond with malicious code. Traditional, IPS devices that do request frame processing, in some cases not even the complete request packets are now faced with the perspective of analyzing response frames. These frames are larger in size and pattern matching algorithms running on them can in many cases lead to a major performance hit.

Buffer Overflows — Static pattern matching have been replaced with generic regular expression matching algorithms to measure pattern length. If the pattern length exceeds a predefined length, a buffer overflow case is detected.

A TASTE OF APHRODITE: AN ARCHITECTURE FOR FALSE POSITIVE REDUCTION

Damiano Bolzoni, Sandro Etalle
University of Twente,
Distributed and Embedded System Group,
P.O. Box 2100, 7500 AE Enschede, The Netherlands
{damiano.bolzoni, sandro.etalles}@utwente.nl

Network intrusion detection systems (NIDSes) are considered an effective second line of defense against network-based attacks directed at computer systems. The Achille’s heel of NIDSes lies in the large number of *false positives* (i.e., false attacks) that occur: practitioners as well as researchers observe that it is common for a NIDS to raise thousands of mostly false alerts per day.

A common way to reduce false positives in SBSes is by deactivating the signatures (*tuning*) relative to vulnerabilities that are not present in the given environment, but this could prevent the NIDS from finding a certain sort of attack. When using a signature-based system (SBS), most alerts are irrelevant or benign phenomena, i.e. related to ICMP protocol, and it is easy to tune the system to avoid this kind of alerts. The same kind of tuning is not applicable to an anomaly-based system (ABS), where commonly a *statistical model* describing the normal network traffic is built and then any behaviour that significantly deviates from the model is flagged as an attack, using a threshold value. The value of the threshold has a direct influence on both false negatives and false positives rates, since they are intrinsically related.

The main idea of our approach is the following: a successful attack (incident) to a system (e.g., a web service) usually produces an anomaly in the *output* of the system. On the other hand, if something in the input of the system causes the NIDS to raise an alarm but does not cause the system to produce an *unusual* output, then this alarm is likely to be a false positive.

APHRODITE works by detecting anomalies in the output of the system and by correlating them to the alerts raised by the NIDS monitoring the input of the system (see Figure 1). Since different systems presents very different outputs (this is also the main problem in applying SBS monitoring to output data), we employ an ABS to monitor the output traffic. In particular we used the payload-

based system POSEIDON, though at least in principle any ABS could be used instead. We want to stress that the NIDS monitoring the input is essential for the functioning of APHRODITE, but it is not part of APHRODITE’s architecture: as a matter of fact APHRODITE can work together with *any kind of* NIDS (be it anomaly-based or signature-based).

To validate our architecture, we have benchmarked APHRODITE in combination with the signature-based NIDS Snort as well as APHRODITE in combination with the anomaly-based NIDS POSEIDON. To do this, we have employed two different data sets: the well-known DARPA 1999 data set (focusing on FTP, Telnet, SMTP, HTTP protocols) and a private data set (in this data set we focus on HTTP traffic because nowadays Internet attacks are mainly directed to web-based services). In 8 out of 9 cases, our benchmarks show a reduction of false positives between 50% and 100%.

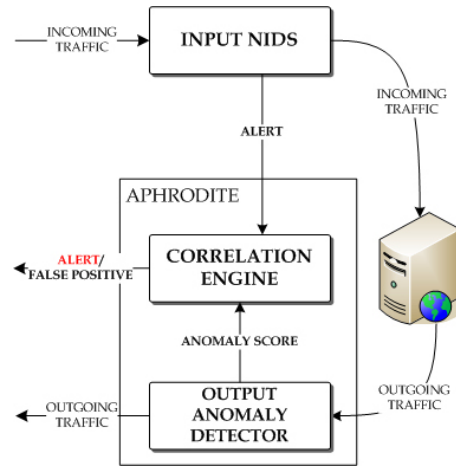


Figure 1: APHRODITE architecture

Automated Response Systems in Mobile Ad Hoc Networks

Shiau-Huey Wang, Karl Levitt

Abstract

This work addresses the problems of automated response systems in Wireless Mobile Ad Hoc Networks (MANETs), which is challenging because the topology of these networks are dynamic and the fully distributed cooperation among nodes makes not only the false positives of intrusion alarms high but also complicates the response analysis. We present a cooperative, distributed and automated response model that addresses these challenges by establishing a secure communication channel and coordinating intrusion response among response agents. The first step of our automated response system (ARS) is *alarm validation* which is crucial in response systems because it can compensate for the false positives of intrusion systems and catch forged alarms. We investigate alarm validation in two aspects, local and global alarm validation. For a single, non-correlated attacker, the response system can use the majority-voting strategy among response agents to locally validate alarms. Furthermore, alarms can be globally validated using *Public Key Infrastructure (PKI)* technique. In order to perform alarm validation, response agents who have related alarms against the same node need to exchange their alarms with each other. However, how to perform this exchange efficiently without hopping through the suspicious node is a critical issue. Currently, we focus on developing a temporary coordinator to aggregate alarms from all response agents monitoring the same node. Through this efficient alarm exchange mechanism, response agents can reduce false positive rate, broadcast global alarms with low message overhead, and prevent false accusation locally and globally. In our future work, response agents can utilize this exchange framework to correlate alarms and determine cooperatively the best intrusion response among a feasible set of responses.

Using the Dempster-Shafer theory for Traffic Classification

Giovanni B. Barone¹, Claudio Mazzariello², Carlo Sansone²

¹Centro di Ateneo per i Servizi Informativi - ²Dipartimento di Informatica e Sistemistica
Università degli Studi di Napoli Federico II
{gbbarone,cmazzari,carlosan}@unina.it

When addressing the problem of network intrusion detection, it is often necessary to have a database of packets captured from a real network, whose actual class (i.e., *normal* or *attack*) is known. As an example, the employment of supervised pattern recognition techniques might be useful for discovering novel attacks; such techniques, however, might need some labeled traffic, containing both normal and attack packets, in order to learn traffic models for correctly detecting anomalous behaviors. A labeled database can be also used for comparing the detection performance of different Intrusion detection Systems (IDS). To build suitable labeled datasets, it should be necessary to perfectly know the network environment and the traffic characteristics in advance. Furthermore, a manual labeling phase is typically required, which is very time-consuming and annoying.

In this work we propose an architecture for automatically building up a traffic database, made up of packets each labeled either as *normal* or as an *attack*. Moreover, we describe a general approach for giving rise to an IDS that makes use of such labeled database for training itself. As we propose to use multiple detection techniques, it is necessary to employ some fusion strategy in order to combine their outputs. As we have no prior knowledge of the traffic characteristics, we use the Dempster and Shafer [1] combination rule, which completely disregards any prior knowledge about the data to be classified. The Dempster-Shafer combination rule, in fact, was calculated regardless of the analyzed data and the classifiers employed. Though some works already propose to exploit the results of such a theory for intrusion detection (see for example [2]), we propose also an architecture able to classify unlabelled data sets. As said before, possible use cases are the automated construction of a labelled dataset for training learning algorithms and the employment in a real network scenario. The proposed architecture consists of two types of Intrusion Detection Modules, respectively Master IDS (M-IDS) and Slave IDS (S-IDS). M-IDS must not require to be trained on labeled data. Typical examples of M-IDS are signature-based IDS, such as *Snort*TM, or IDS based on unsupervised techniques. A bank of M-IDS starts analyzing offline the packets contained in a dumped traffic database. No prior knowledge is needed about such traffic. According to the theory of Dempster and Shafer (D-S theory) [1], a Basic Probability Assignment (*bpa*) can be associated to each M-IDS. Each *bpa* describes the subjective degree of confidence attributed to each different classification performed by the considered IDS, by means of prior hypotheses or observations. Starting from the category a M-IDS belongs to (i.e., signature-based, unsupervised, etc.), we defined some criteria for suitably assigning *bpa*'s to M-IDS. The D-S theory has been frequently applied to deal with uncertainty management and incomplete reasoning. It is worth noting that it is different from the classical Bayesian theory, since for Bayes the sum of $P(A)$ and $P(\neg(A))$ always equals one, while this is not necessarily true according to the D-S theory. Moreover, the D-S theory can explicitly model the absence of information, while in case of absence of information a Bayesian approach attributes the same probability to all the possible events. These characteristics can be very attractive for managing the uncertainties of our domain, due, for example, to the presence of the so-called *zero-day* attacks. Thus, each decision from each of the M-IDS will be supported by the *bpa* associated to it, which will express the degree of belief in the classifier decision. Such *bpa*'s will be combined by using the Dempster-Shafer combination rule [1], in order to obtain a *bpa* for each possible class the packet can belong to. We also defined a criterion for obtaining a crisp classification of each packet from the overall *bpa*, by means of a properly defined index. After a suitable number of packets has been classified with a satisfactory degree of confidence, such packets can be fed to each of the S-IDS. S-IDS are based on supervised learning techniques; they will use the portion of reliably labeled traffic as a training set. Once their training is over, they will contribute to a further classification of packets according to what they learned so far. Once again, the D-S rule can be used for suitably combining the decisions of the banks of M-IDS and S-IDS. It is also possible to employ the whole system as an online IDS. The well known ability of supervised systems in generalizing known attack patterns for detecting novel attacks might in fact help in detecting attack which were not known when the statically configured M-IDS were set up.

We evaluated the capability of the proposed architecture in correctly labeling a traffic database. In order to benchmark the system performance, we used the Lincoln Lab network traffic database. Though such a database has been heavily criticized in the past years, it is still referred to as a benchmark for data mining and pattern recognition techniques in intrusion detection. In particular, we evaluated the performance, in terms of false alarm rate and missed detections, obtained by combining two M-IDS (*Snort*TM and an unsupervised neural network) according to the proposed architecture and the D-S theory. We observed that the proposed system can improve the performance of each base classifier alone in correctly labelling traffic data. Furthermore, we are interested in demonstrating that an S-IDS can be reliably trained by using traffic data labeled by the bank of M-IDS's. In order to prove it, we compared the performance obtained by the chosen S-IDS (a rule-based one based on *Slipper* [4]) when trained with data labeled by our architecture with respect to those obtained when they are trained on data whose real labels are known in advance. The attained difference in terms of error rate turned out surely acceptable (only 0.07% with a recognition rate of 99.9%), so confirming our claims. Finally, we were also able to observe an interesting tradeoff between the attainable performance and the size of the training set used. Different training set sizes can be obtained by setting a suitable threshold on the index that furnishes an estimate of the accuracy of the packet labels provided by the bank of M-IDS. The above described tests have been carried out by using the hardware equipment of the University Federico II server farm. Further analysis on real data will be run on traffic sniffed at this server farm, in order to make them significant for the deployment in a real enterprise scenario.

References

- [1] J. Gordon, E.H. Shortliffe, "The Dempster-Shafer Theory of Evidence", in B.G. Buchanan and E.H. Shortliffe (Eds.), *Rule-Based Expert Systems*, Addison-Wesley, pp. 272-292, 1984.
- [2] T.M. Chen, V. Venkataramanan, "Dempster-Shafer Theory for Intrusion Detection in Ad Hoc Networks", *IEEE Internet Computing*, pp. 35-41, November-December 2005.
- [3] W.W. Cohen, Y. Singer. "Simple, Fast, and Effective Rule Learner" in Proc. of the 16th National Conference on Artif. Intell. and 11th Conference on Innovative Applications of Artif. Intell., July 18-22, Orlando, Florida, USA, pp. 335-342, 1999.

Simulation Environment for Investigation of Cooperative Distributed Attacks and Defense

Igor Kotenko, Alexander Ulanov

Computer Security Research Group, Saint-Petersburg Institute for Informatics and Automation (SPIIRAS)
{ivkote, ulanov}@iias.sbp.su

Nowadays we are witnesses of increasing number of distributed attacks on global computer networks. Much of them are aimed on the distributed denial of service (DDoS) of critical information resources. These attacks are realized due to joint efforts of many malicious software components that are deployed on compromised Internet hosts. The general approach to DDoS defense includes mechanisms of *attack prevention, detection, tracing the malicious traffic sources and attack counteraction*. Because of gravity and complexity of DDoS the design of effective defense is a complicated scientific and technical problem. It is sufficiently hard to examine and evaluate the effectiveness and efficiency of defense mechanisms in practice. However these mechanisms might be simulated with the necessary fidelity and thoroughly analyzed.

We propose *the approach and developed software environment intended for simulation and investigation of distributed cooperative attacks and defense systems*. The main attention is given to the *integrated agent-oriented and packet-level approach to the simulation of security processes in the Internet*. It can provide the acceptable fidelity and scalability of implementing computer attacks and defenses. The special attention is given to *cooperative distributed defense mechanisms that are based on the deployment of defense components in various Internet subnets*. That is intended for simulating the interactions of various ISPs security elements.

The cybernetic counteraction is supposed to be represented as the interaction of the teams of malefactors and the teams of security agents. The agent teams can be opposed to each other or cooperate. Attack agents are subdivided at least into two classes: “daemons” and “masters”. To simulate distributed cooperative defense, the security agents belong to the following classes: information processing (“samplers”); attack detection (“detectors”); filtering and balancing (“filters”); traceback and investigation (“investigators”).

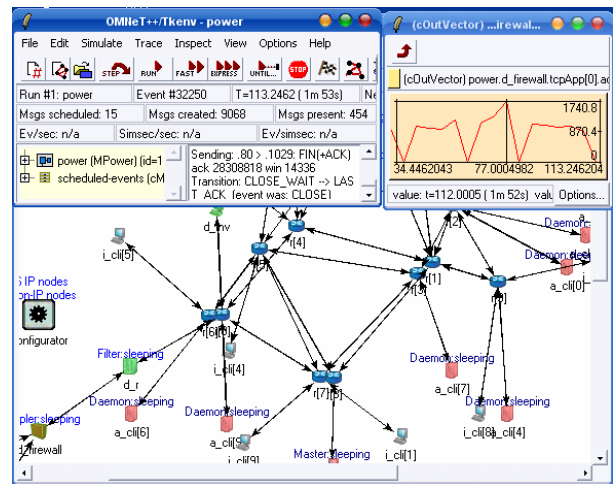
The proposed simulation approach presumes the following components of the simulation environment developed: (1) Discrete-event Simulation Framework (implemented on OMNeT++), (2) Internet Simulation Framework (using OMNeT++ INET Framework), (3) Attack and Defense Framework (Library of attacks and defenses), (4) Multi-Agent Simulation Framework. Attack and Defense Framework includes attack and defense modules and the modules that expand the hosts of INET Framework: filter table and packet analyzer.

The basic window *the simulation environment* developed (Fig.1) shows a simulated computer network (hosts and channels). Hosts can fulfill different functionality depending on chosen parameters and internal modules. Internal modules are responsible for functioning of protocols and applications at various levels of OSI model. Applications (including agents) are established on hosts. The window for simulation management allows looking through and changing simulation parameters. It is important that it is possible to see the events which are valuable for understanding attack and defense on time scale.

Corresponding windows show the current status of agent teams. It is possible to open windows which characterize functioning of particular hosts, protocols, agents, defense methods, see contents of the packets, etc. The following *parameters are used in the environment to define the attack*: victim type; type of attack; attack rate dynamics; impact on a victim; persistence of agent set; possibility of exposure; source address validity; degree of automation. *Defense mechanisms are determined in the environment* by the following parameters: deployment location; mechanism of component cooperation; covered defense stages; attack detection technique; attack source detection technique; attack prevention and counteraction technique; model data gathering technique; determination of deviation from model data technique.

The environment allows to analyze various classes of attack and defense mechanisms. The abstract and poster is devoted to the investigation of *the models of cooperation between distributed defense teams*: (1) *filter-level cooperation*: the team whose network is under attack can apply filtering rules on the filters of other teams; (2) *sampler-level cooperation*: the team whose network is under attack can get the traffic information from the samplers of other teams; (3) *“poor” cooperation*: the teams can get the traffic information from the samplers of some other teams and apply filtering rules on the filters of some other teams (each team knows a subset of other teams depending on the cooperation degree); (4) *“full” cooperation*: the team whose network is under attack can get the traffic information from all samplers of other teams and apply filtering rules on all filters of other teams. Such cooperation schemas are used in the cooperative DDoS defense methods: COSSACK, Perimeter-based DDoS defense, DefCOM, Gateway-based, ACC pushback, MbSQD, SOS, tIP router architecture, etc. The cooperation schemas can be investigated and compared using the analysis of various parameters: (1) incoming traffic before and after filters; (2) normal and attack traffic rate from the whole traffic coming into defended network; (3) false positives and false negatives rates, (4) detection and reaction times, etc.

In the *future research* we are planning to expand the attacks and defenses library, elaborate particular components functionalities. The important constituent of future research is numerous experiments to investigate various attacks, defense mechanisms (attack prevention, detection, tracing the attack sources and counteraction) and optimal defense combinations.



Belief representation of network behavior in user and application context to improve intrusion detection

Pedro Bados Aguilar
pedro.bados@nexthink.com
NEXThink S.A., Switzerland
www.nexthink.com

In this work we are firstly exploring an innovative framework to represent the classical network information based on binaries and user accounts contexts. At this point, we have developed a new generation of belief networks which model the usage of these applications by a user or a community of users within the corporate environment. Our research has been initially implemented as the core technology of REFLEX solutions and already validated in several customer deployments.

In our preliminary studies, we have identified that most commercial and research intrusion detection systems have a focus either on pure network traffic analysis or on a host behavior inspection. It's well-known that both approaches present particular advantages in accuracy and performance but at the same time they might have some drawbacks concerning operational aspects. We have also observed that since both alternatives are complementary and commonly used together, most of these shortcomings are aggregated often resulting in unmanageable systems.

In the research performed at NEXThink we have designed and developed a new hybrid context where every single network activity is transparently associated with a user account and application binary. This association is performed with certainty in 100% of cases. It's important to note that most of the research and market approaches to analyze network traffic are based on protocol flows to guess applications and on IP addresses to identify the users. In our framework, applications are the real binaries names running at end-points while the users are the real user accounts logged in the machine. Moreover, our technique of classification has been proved to silently perform on real-time for networks of thousands of computers.

At this point, our analysis engine stores and builds up a centralized model of network behavior for users and applications. This model corresponds to a belief network of the usage of this application by this concrete user account. Finally, an analyzer is able to detect and measure deviations in the models generating alarms of abnormal usage and providing explanations of such anomalies.

Belief networks [1] are representations of Bayesian relationships between evidences and facts. They are currently used in several domains to reason under uncertainty such as illness prediction in medicine or incident management. Two important properties make them especially suitable for our objective. Firstly, their expressiveness permits the security administrator to have a concrete explanation about the true or false positive when receiving an alarm. We have observed that meaningful explanations for abnormal events greatly reduce the reaction time in large organizations. Secondly, belief networks allow enriching the model with the insertion of new evidences and relations without changing or retraining the system.

Bayesian representations can be easily understood and extended to accommodate new types of detections. Since they actually represent the expert knowledge about a certain domain, they can be adapted and enhanced with user interaction in a very intuitive way. This simple link between the detection model and the end-user represents a very attractive field for risk analysis.

The advantages of this new framework for the security field are evident. The fact of having all network events associated to a certain executable name, version and their user account dramatically increases the detection accuracy of virus propagation, port scanning or abuse of privileges. At the same time we have observed that these parameters along with the simple explanations become capital for administrators to quickly discard irrelevant events from the list of abnormal incidences.

In addition, the fact of modeling the network behavior for a certain user account sheds some light on a challenging research field as identity theft detection. We have proved that a user network behavior model is unique in 96% of the cases and therefore sufficient to identify the person behind the computer while the discrimination of the rest of users is successfully achieved in 90% of the situations.

[1] Pearl, J. (1988) Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann.

A New Freely Available Data Set for the Evaluation of Signature-Based IDS

Frédéric Massicotte
Communications Research Centre
3701 Carling Avenue
P.O. Box 11490, Stn. H
Ottawa, ON, Canada K2H 8S2

François Gagnon
Carleton University
1125 Colonel By Drive
Ottawa, ON, Canada K1S 5B6

An Intrusion Detection System (IDS) is a critical aspect of a network security posture. Although today there are many IDS products available, the data used to evaluate these IDS is either proprietary or obsolete. For instance, even though no new data set has been released by DARPA since 2000, the DARPA traffic traces are still used today by the security community since they represent the only significant and freely available research data. In this poster, we present an automatically generated and publicly available data set that can be used for the evaluation of signature-based IDS.

The current version of our data set was developed to test and evaluate network and signature-based IDS for attack scenario recognition. It contains traffic traces of attack scenarios derived from the execution of well-known Vulnerability Exploitation Programs (VEP) taken from various sources such as SecurityFocus, Metasploit, Nessus and Operator. The data set is a collection of tcpdump traffic traces, each trace containing one attack scenario. To generate the data set, we used 124 VEP (covering a total of 92 vulnerabilities) and 108 different target systems. Each VEP was launched against vulnerable and non-vulnerable systems (among the 108 target systems) that offer a service on the port targeted by the VEP. Each combination (VEP + configuration + target) corresponds to an attack scenario and produces a traffic trace in the data set. The 124 VEP are distributed among 17 different ports and the data set contains more than 10000 traffic traces. The data set contains a vast diversity of attacks such as buffer overflow, information leak, privilege gain and denial of service. In particular, these attacks use different remote access techniques after exploiting the vulnerability such as direct and reverse shell. Moreover, some attacks in the data set are not detected by the current version of Snort and Bro. In addition to those traffic traces, our data set also contains traces resulting from the application to the VEP of different IDS evasion techniques, such as Fragroute and Whisker. This part of the data set is used to verify whether IDS are able to detect modified attacks.

One of the main features of our data set is the documentation of each traffic trace. Each traffic trace is documented through four characteristics: the target system configuration (operating system and installed software), the VEP configuration (options used), whether or not the target system has the vulnerability exploited by that VEP (based on the target configuration and the information available on SecurityFocus for the VEP) and the actual success or failure of the attack attempt (based on the VEP output and the effects on the target). Since the traffic traces contained in the data set are properly documented, the evaluation process can be automated.

To automatically generate such a large scale documented and maintainable data set, we developed a controlled virtual network using VMware Workstation 5.0. This controlled virtual network allows us to record network traffic produced during attacks, control the network (e.g., traffic noise), to control the attack propagation (confinement), to use various heterogeneous target system configurations, and to quickly recover from attacks. It is flexible (it can easily apply IDS evasion techniques on attacks), updatable (it can easily incorporate new target configurations and new attacks) and completely automated (from virtual network setup, attack execution to the documentation of traffic traces). Our controlled environment is not only a database of virtual machines, it is also a layer on top of VMWare allowing to set up experiments that can be described using a script language. For example, this virtual network infrastructure was also used to provide data to the LEURRE and SCRIPTGEN projects of Eurecom.

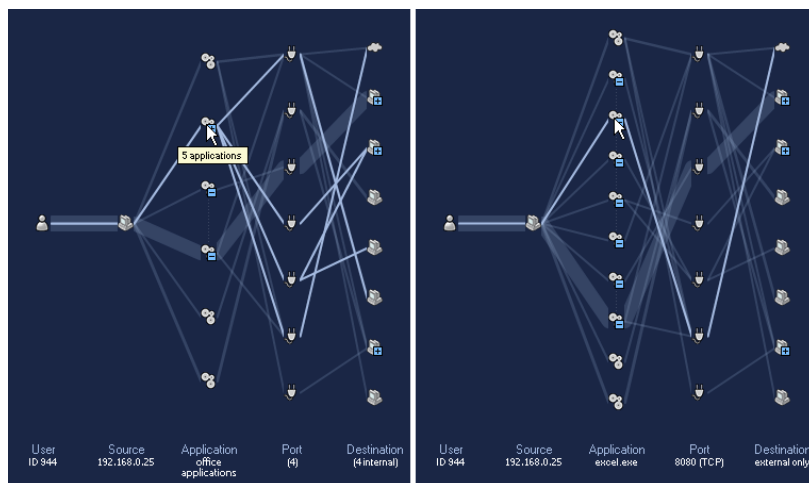
For now, the data set presented here can be used to evaluate signature-based IDS to see if they can detect known real attacks (the false negative rate) and if they can distinguish between successful and failed attempts of an attack (false positive on failed attempt). Our data set has proven to be useful in several projects. For example, we found out that Snort and Bro were not able to distinguish successful from failed attacks. By periodically updating and sharing the attack traces we generated with the research community in network security we could provide a recent common reference to evaluate IDS.

Enhanced parallel-coordinates visualization to help understand anomaly-based alarms

Patrick Hertzog, NEXThink S.A.¹, Scientific Park (PSE-B), 1015 Lausanne, Switzerland
patrick.hertzog@nexthink.com

Security systems can generate alarms and to be able to take the right decision, the security administrator has first to understand them, i.e., understand why the alarm has been generated and which items (i.e., users, applications, hosts and ports) are involved. In most systems, the generation of alarms can be either signature-based or anomaly-based. Signature-based alarms should be easy to understand as they are triggered when a connection (or a set of connections) respects a precise rule (e.g., the signature of a known attack). Understanding anomaly-based alarms is another story: security systems often use sophisticated algorithms taking into account a large number of parameters whose output cannot be directly interpreted by the human operator. In that situation, we claim that if one want to understand what is abnormal, one need first to understand what is normal. We need then to be able to visualize the normal behavior (i.e., normal connections) along with the alarms.

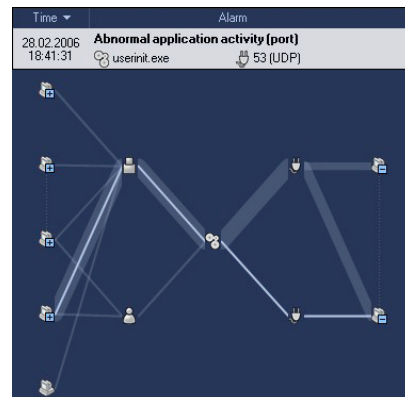
A good way to display connections is to use a parallel-coordinates visualization [1]. Each connection is represented by a polygonal line linking the involved network items (i.e., user, application, source, port, destination) on the different axes. The thickness of the line is proportional to the number of connections involving the same items. Although they seem to be an ideal solution, parallel coordinates visualizations become quickly unreadable and therefore unusable when displaying a lot of information. They are also an easy target for deception attacks based on occlusion and jamming as explained in [2]. Those issues are mainly due to the number of lines concurrently drawn on the graph and to reduce that number, we introduce a grouping method that enables to display individual network items or groups of network items as values on the axes.



Network items can be interactively grouped (and inversely) according to similar characteristics. For instance, all applications of the same category (e.g., browsers) can be grouped to form a meta-application (e.g., representing all browsers). Meta-applications can be grouped together to form a higher order group (e.g., browsers and e-mail clients grouped into internet applications) and so on. The process is the following: first, the different network items have to be organized in a tree structure. Each node of the tree corresponds to a similar characteristic of its children elements and the leaves contain the actual network items. Each node also

has an attribute *state* indicating if it is *collapsed* (i.e., its children will be displayed as a group) or *expanded* (i.e., its children will be displayed individually). One can choose any characteristics for the nodes as long as one network item is only represented by one leaf of the tree. However, the more balanced the tree is, the better the result will be. Then, an optimal tree state is computed. Once an optimal tree state has been computed, it can be mapped to the actual graphical representation. It means that each leaf whose state of its parent node is *expanded* will be displayed as an individual network item. On the other hand, each collapsed node whose state of its parent node is *expanded* will be displayed as a group of network items.

With this grouping method, we are able to display parallel-coordinates visualizations able to represent a big number of connections while remaining readable. We can therefore use such visualization to display the normal behavior of network items along with the incriminating connections. For instance, in the figure below, the administrator receives an alarm stating that a particular application has a strange behavior due to the port it used. With the help of the visualization representing all connections of the said application, she/he can quickly and easily understand why the alarm has been generated and see which resources have been involved.



- [1] A. Inselberg. The plane with parallel coordinates. In *The Visual Computer*, pages 69–91, 1985.
- [2] G. Conti, M. Ahamad, and J. Stasko. Attacking information visualization system usability: Overloading and deceiving the human. In *Proceedings of the Symposium On Usable Privacy and Security (SOUPS 2005)*, Pittsburgh, PA, USA, July 2005.

¹ <http://www.nexthink.com>

Mining a Worm Detection System Data

Urko Zurutuza*and Roberto Uribeetxeberria
Mondragon Unibersity
{uzurutuza,ruribeetxeberria}@eps.mondragon.edu

James Riordan and Yann Duponchel
IBM Zurich Research Laboratory
{rij,ydu}@zurich.ibm.com

July 21, 2006

Billy Goat is a reliable Worm Detection System (WDS). It is focused on detecting machines in the network infected with known worms, and in this respect it is free of false positives by construction. It also provides additional information that can be analyzed to detect new worms or other emerging threats in the network. Billy Goat is designed to take advantage of the propagation strategies of worms. To discover machines to infect, most worms try to connect to IP addresses selected at random or scan entire ranges of addresses. By doing so, they find most of the machines in a network, but they also try to connect to a large number of unused addresses. Billy Goat functions by responding to requests sent to unused addresses, feigning the existence of a large number of machines and services.

Billy Goat has been deployed both throughtout IBM corporate network and on the Internet. It uses a rule based engine to determine the presence of an infected machine, but there is still a big amount of data captured by Billy Goat which is not identified by the rule engine. Thus, this data is out of the scope of Billy Goats' objectives, but analyzing it can give very useful information of what is going on in the network. In addition, it has no sense trying to identify infected machines when deploying a Billy Goat sensor on Internet, but it also gives a very rich information of the automated attacks being carried out every day.

To address those issues we are applying data mining techniques to the data gathered by both the internal and external Billy Goats. We have used CLARATy to analyze the Billy Goat data. CLARATy is based on the AOI (Attribute Oriented Induction) algorithm, which had been modified to better support the root cause analysis, avoiding over-generalization of the clusters obtained.

Our experiments have shown that the patterns obtained can very helpfully complement the findings of the rule based engine, making Billy Goat a hybrid rule-based and anomaly detection system.

When mining internal Billy Goat data, we found that more than the 50% of the data collected corresponded to device misconfigurations. This gives a real perspective of what happens daily in any network, and turns Billy Goat a very helpful tool for system administrators to fix those misconfigurations, as it can provide precise information on 'what happens to who'.

Instead, when mining external Billy Goat data, results show that it provides much more information on the attacks that are being carried out. Thus, CLARATy is configured to focus on identifying this attack patterns instead on identifying the infected machines. This makes Billy Goat very suitable for acting as a worm early warning system.

This preliminary analysis of Billy Goat data has helped us in establishing an adecuate framework for data mining of worm related activity, and the next logical step will consist on developing automatic signature generation techniques for their use in traditional Intrusion Detection Systems (IDS).

*This autor is supported by the grant BFI05.454 of the Department of Research, Education and Universities of the Basque Government

A Collaborative Architecture for Intrusion Detection Using Information Fusion Techniques

Claudio Mazzariello², Francesco Oliviero², Lorenzo Peluso¹, Simon Pietro Romano², Carlo Sansone²

¹Consorzio Campano di Ricerca per l'Informatica e l'Automazione Industriale
{l.peluso}@criai.it

²Dipartimento di Informatica e Sistemistica (DIS), University of Napoli Federico II
{cmazzari, folivier, spromano, carlosan}@unina.it

DDoS (Distributed Denial of Service) attacks constitute one of the major threats for modern computer networks. By exploiting the constantly growing bandwidth available to end users, and the spread of cheap and powerful hardware, malicious users are able to control and coordinate many resources to attack the victim. Hence, a distributed and collaborative system, by observing several events occurring in different places of a network, might be able to detect anomalous activities in their early stages, eventually improving the detection timeliness and the effectiveness of the reaction. According to the autonomic communication paradigm, we defined a system made up of entities capable of self-organization. Such entities are able to exchange information according to a well defined communication protocol, which is aimed at optimizing the spread of security-related information. Each of the entities involved in security assurance is committed to take into account both local and global information for deciding whether, in its own *opinion*, an anomalous activity is in progress. Such an *opinion* is then notified to its neighbors. Decisions regarding local observation disregard the global status of the monitored scenario, and are taken by observing some metrics related to critical traffic flows. Each node will take into account the *opinion* of the neighbor nodes for taking its own decision about the activity in progress and build its own *opinion*. To avoid opinions biasing due to the corruption of one or more detection nodes, we use mechanisms based on reputation and reliability. In the following, we will refer to the entities constituting the system as *probes*. Each of the probes implements some classification techniques; a measure of reliability can be associated with each of them, according to several criteria. For example, systems which proved to perform better on a given training set might have a higher intrinsic value of reliability. Furthermore, each of the probes will keep a value expressing the reputation it assigns to each of the neighbors. A neighbor which has always been disagreeing in the past with my *opinion*, and which has a low reliability value, might be assigned a bad reputation, while quite a reliable node which agreed with me most of the time in the past might be assigned a good reputation. Hence, reliability is an objective evaluation parameter, while reputation is subjectively evaluated at each node. A powerful way to express the degree of confidence each probe has in its own opinion, and to combine such degrees of confidence in the reported event in order to evaluate the *opinion* at each probe, might be the Dempster-Shafer mathematical theory of evidence [1]. It helps in expressing the degree of confidence in each of the considered hypotheses, which in the case of intrusion detection might be the occurrence or non-occurrence of an attack, and explicitly models the uncertainty among such hypotheses when reporting an event. The *opinions* coming from the neighboring probes can be furthermore weighted according to the local reputation of each of them.

The idea behind the self-organization capability of the system is that the closer a probe is to the attack victim, the more the attack symptoms will be evident and straightforward to interpret when available. On the other hand, a rough analysis at the edges of the network might be useful for early detection of the activities that might be ascribed to a malicious behavior. The proposed architecture tries to combine both the requirements, in order to allow the deployment of the most suitable analysis tool in the place where it is most needed. The architecture is made up of a set of probes, which might be regarded as peer sensors, each one carrying out the prescribed type of analysis. Once an anomalous behavior is detected by any of them, it starts spreading the new of attack symptoms present in its neighborhood. Many strategies can be adopted to spread such information, by taking into account the trade-offs between network overhead due to signaling and alert communication, and the speed and effectiveness of the information dissemination. Among the proposed information spreading strategies, we are evaluating a modified version of gossiping [2]. When using gossiping, each node transmits information to each of its neighbors with non-zero probability p . When coping with attack detection, we might be able to identify nodes along the attack path. If a node is on the attack path, it transmits information to the next hop on the attack path with probability 1. By transmitting detection-related information outside the attack path, we increase the probability of propagating such information to the entities which are interested in it. As we are dealing with DDoS detection, some nodes outside the main detected attack path might be involved in the attack too. With randomized information spreading we explore a wider part of the network rather than the mainstream attack path. Both the nodes along and outside the main attack path analyze the traffic and express their own opinion, and in turn can reinforce or weaken the spreading new. Such an architecture might be able of understanding which is the main path of an attack, and by means of a suitable traceback protocol trigger a reaction which counters the effects of the malicious behavior. The self-organizing nature of the proposed architecture allows it to adapt to changing scenarios and to reconfigure, in order to cope with the everchanging nature of network attacks.

References

- [1] J. Gordon, E.H. Shortliffe, "The Dempster-Shafer Theory of Evidence", in B.G. Buchanan and E.H. Shortliffe (Eds.), *Rule-Based Expert Systems*, Addison-Wesley, pp. 272-292, 1984.
- [2] D. Kempe, A. Dobra, J. Gehrke, "Computing aggregate information using gossip", in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, Cambridge, MA, 2003.

Semantical File Integrity Checking Using Reconfigurable Hardware

Jakub Botwicz

Warsaw University of Technology

We have identified a class of problems involving immense effort during data classification and therefore it can benefit from hardware acceleration. These problems are:

- detection of unsolicited electronic mail (spam),
- identification of document's language,
- detection of malicious code (malware),
- identification of defaced web pages,
- content-based files classification.

The common feature of systems solving these problems is that all of them are based on n-class categorization with supervision. The next common factor of this systems is that the feature extraction process is usually based on n-gram or byte histogram analysis. This task can be made efficiently in reconfigurable hardware. Our goal is to provide hardware acceleration module that can be used in all of listed problems and also we show in the example system.

To combine malware detection and filetypes identification we have created new definition of the problem called *semantical integrity checking*. File is correct if:

- there is a consistency of file: extension, header, footer and its content,
- it is compliant with defined file exchange policy (e.g. attachments),
- there no embedded executable code inside non-executable files,
- there are no predefined signatures of malicious code in executable files.

The main features of our work:

- data is being scanned using so called *sliding window of analysis*,
- it is possible to detect embedded malicious code inside non-executable files,
- we combine signature matching techniques with data classification: for different types of file there are different signatures to detect – e.g. there are different viruses and worms for Windows and for Linux platforms.

Preliminary results show that the FPGA data classification module will use less then 10% of resources of the biggest chip currently available at the market (in our case it is from Stratix II family designed by Altera) and it is possible to process data with 800 Mb/s throughput. Currently we work on a corpus of 6000 files from 12 different types and the precission of the classification is bigger the 95%.

There are many possible application fields of the system presented in this abstract. The most obvious are content-inspecting network gateways, which are more and more overwhelmed by data throughput. Also intrusion detection and prevention systems, antisпам and antivirus e-mail gateways can benefit from this type of hardware acceleration.

YAMA: Simplifying Computer Network Intrusion Detection Experiment Analysis

Nestor Hernandez, Robert K. Cunningham
MIT Lincoln Laboratory,
244 Wood Street, Lexington MA, USA
rkc@ll.mit.edu

Abstract. Computer network intrusion detection performance analysis requires precise labeling of all traffic into two and perhaps three categories, so that the false alarm and detection rates can be correctly measured. In the first category is “background traffic” and is related to the mission of those using the network. In the second category is “attack traffic.” When an experiment is performed on-line with an intrusion detection system, it may also be important to identify traffic that originates from that system. Today doing this is tedious and difficult, requiring analysis by personnel with a deep understanding of multiple protocols.

In a now-famous April Fool’s day RFC, *The Security Flag in the IPv4 Header*, Bellovin jokingly proposed solving this by requiring attackers set an evil bit. Attackers demurred. As a result, controlled evaluations cannot use such an in-packet flag either, as it introduces an unwanted artifact into the testing procedure. Much better is to use a tool that can mark packets after the fact, but doing this is extremely difficult. As Bellovin noted: “Firewalls, packet filters, intrusion detection systems, and the like often have difficulty distinguishing between packets that have malicious intent and those that are merely unusual.”

All of those devices have the difficult problem of marking packets given only the contents of the packet and perhaps the configuration of the network. Controlled experiments have a significant advantage—the intent of each user action is known, so a tool can be built to associate user actions and mark produced packets. Doing this is difficult—a single user action produces multiple sessions using different protocols, and the total number of packets produced can easily range into the thousands.

YAMA (Your Able Marking Aide) is a tool that can correctly label sessions and packets associated with that set of actions, given a network configuration (domain names, IP addresses, and a web page corpora) and a set of user actions. YAMA’s implementation currently includes processing modules for web traffic. Packets associated with a user visiting a web page can be correctly labeled, including the domain name service lookup of the IP address, loading of pictures, multiple frames, and advertisements. An evaluation of the tool using data from Alexa’s “Top 100 Sites” show that nearly all sessions and packets can be correctly associated with the action of visiting the web site.

Keywords: Intrusion Detection Evaluation, Computer Networks, Evil Bit