DETECTION OF ILLICIT TRAFFIC USING NEURAL NETWORKS

Paulo Salvador, António Nogueira, Ulisses França and Rui Valadas
University of Aveiro, Instituto de Telecomunicações, Aveiro, Portugal

Keywords: Intrusion Detection System, Firewalls, Port Matching, Protocol Analysis, Syntatic and Semantic Analysis, Traffic Signature, Traffic Pattern, Neural Networks.

Abstract: The detection of compromised hosts is currently performed at the network and host levels but any one of these options presents important security flaws: at the host level, antivirus, anti-spyware and personal firewalls are ineffective in the detection of hosts that are compromised via new or target-specific malicious software while at the network level network firewalls and Intrusion Detection Systems were developed to protect the network from external attacks but they were not designed to detect and protect against vulnerabilities that are already present inside the local area network. This paper presents a new approach for the identification of illicit traffic that tries to overcome some of the limitations of existing approaches, while being computationally efficient and easy to deploy. The approach is based on neural networks and is able to detect illicit traffic based on the historical traffic profiles presented by ”licit” and ”illicit” network applications. The evaluation of the proposed methodology relies on traffic traces obtained in a controlled environment and composed by licit traffic measured from normal activity of network applications and malicious traffic synthetically generated using the SubSeven backdoor. The results obtained show that the proposed methodology is able to achieve good identification results, being at the same time computationally efficient and easy to implement in real network scenarios.

1 INTRODUCTION

The detection of compromised machines is currently performed at two levels: network and PC. At the PC level, we have antivirus, anti-spyware and personal firewalls. All these systems are ineffective in the detection of hosts that are compromised via new or target-specific malicious software. Network firewalls and Intrusion Detection Systems (IDS)(Denning, 1987; Mukherjee et al., 1994; Ilgun et al., 1995) systems were developed to protect the network from external attacks, but they are not designed to detect and protect against vulnerabilities that are already present inside the local area network. The PC security is commonly compromised by careless behaviors of its legit users: a Trojan horse can get inside a PC via an e-mail message rashly open or by browsing less safe web sites. Therefore, the firewall is not able to protect the network and their users because the attack is made using information that is embedded in licit traffic. IDSs analyze network traffic by looking for characteristic traffic profiles of compromised machines or their attacks; these profiles are called signatures. These signature-based systems are effective only if the compromised host do not simulate licit traffic. Nowadays, the main goal for taking control of network machines is the search and retrieval of classified information, maintaining themselves discreetly active and making their detection almost impossible by traditional IDSs. Antivirus and anti-spyware can only detect and remove already known viruses or Trojan horses. Illicit applications that were recently created are discrete, undetectable and are not listed in any threats database, making its detection and removal virtually impossible. Personal firewalls could be an effective tool to avoid PC infection but they require an average know-how level of the tool functionalities and special attention by the user, which rarely happens. A PC user may unconsciously open doors to a mechanism that turns its PC into a compromised machine. As soon as the system becomes compromised, the personal firewall becomes ineffective because the PC's total or partial control now belongs to other users.

This paper will present a new approach for the identification of illicit traffic/applications that tries to overcome some of the limitations of existing approaches, while being computationally efficient and easy to deploy. The approach is based on neural networks and detects illicit traffic based on the historical traffic profile presented by each network application. In a first phase, a neural network model is conveniently trained using data traces corresponding to completely/undoubtedly identified traffic profiles. These traces can result from traffic measurements collected on a real network operating scenario or from synthetically generated traffic obtained in totally controlled conditions. In a second phase, the trained model is used to identify new traffic profiles that are presented as inputs. The results obtained show that the proposed model is able to achieve good identification results. Besides, the computational burden is relatively small and the proposed methodology can be easily implemented in real network scenarios. Neural networks have been proposed to detect intrusions at host level (Debar et al., 1992; Ryan et al., 1997) and we extended this approach to detect illicit traffic at network level.

The proposed framework is intended to be incorporated in a future platform that will allow on-line detection of illicit traffic: the platform should interact with traffic analysis systems, should have a database of utilization profiles from different network users, an easy-to-update list of illicit traffic signatures and a system of alert and counter measures.

It is known that several traditional techniques that have been used to identify IP applications have important drawbacks that limit or dissuade their application: (i) port based analysis presents some obvious limitations since most applications allow users to change the default port numbers by manually selecting whatever port(s) they like, many newer applications are more inclined to use random ports, thus making ports unpredictable and there is also a trend for applications to begin masquerade their function ports within well-known application ports; (ii) protocol analysis (Jiang et al., 2005; Chen and Laih, 2008) is ineffective since IP applications are continuously evolving and therefore their signatures can change, application developers can encrypt traffic making protocol analysis more difficult, signature-based identification can affect network stability because it has to read and process all network traffic, protocol analysis is not able to deal with confidentiality requirements; (iii) syntactic and semantic analysis of the data flows (Wang and Stolfo, 2004; Jiang et al., 2005) can be a burden to network stability due to its high processing requirements and is not appropriate when dealing with confidentiality requirements because, in these situations, it is not possible to have access to the packet contents.

The paper is organized as follows: section 2 will present the state of the art on intrusion detection and prevention systems; section 3 describes the main ideas behind the proposed detection approach and the details of the developed framework; section 4 presents and discusses the most important results and, finally, section 5 presents the main conclusions of this study.

2 STATE OF THE ART ON INTRUSION DETECTION AND PREVENTION

Intrusion detection system (IDS) are software platforms aimed to detect the undesirable remote manipulation of computers from the Internet. IDSs are used to detect several types of malicious traffic that can not be detected by a conventional firewall, like network attacks to vulnerable services, attacks directed to applications, unauthorized logins, access to files and malware distribution. An IDS includes several components: (i) sensors, that generate security events; (ii) a console, to monitor events and alerts and control the sensors; (iii) a central engine that saves the events generated by sensors on a database and uses the system rules to generate security alerts of the received events. There are several types of IDSs: Network Intrusion Detection System (NIDS), an independent network platform that identifies intrusions by examining network traffic; Protocol-based Intrusion Detection System (PIDS), system or agent usually located in front of a server that monitors and analyzes communication protocols between a device and a server; Application Protocol-based Intrusion Detection System (APIDS), system or agent usually located in front of a group of servers that monitors and analyzes communication protocols that are specific to a certain application; Host-based Intrusion Detection System (HIDS), a PC agent that detects intrusions by analyzing system calls, application logs, changes on the file system, among other activities; Hybrid Intrusion Detection System (HyIDS), that combines one or several of the above approaches. For example, Snort or Prelude can operate as NIDS or HIDS.

An IDS is said to be passive if it simply detects and alerts. When suspicious or malicious traffic is detected an alert is generated and sent to the administrator or user and it is up to them to take action to block the activity or respond in some way. A reactive IDS will not only detect suspicious or malicious traffic and alert the administrator, but will take pre-defined proactive actions to respond to the threat. Typically, this means blocking any further network traffic from the source IP address or user.

IDSs can be signature-based or anomaly detection systems. A signature based IDS will monitor packets on the network and compare them against a database of signatures or attributes from known malicious threats. This is similar to the way most antivirus software detects malware. The issue is that there will be a lag between a new threat being discovered in the wild and the signature for detecting that threat being applied to the IDS. During that lag time, the IDS would be unable to detect the new threat. An IDS which is anomaly based will monitor network traffic and compare it against an established baseline. The baseline will identify what is normal for that network - what sort of bandwidth is generally used, what protocols are used, what ports and devices generally connect to each other - and alert the administrator or user when traffic is detected which is anomalous, or significantly different, than the baseline.

An intrusion prevention system (IPS) is a computer security device that exercises access control to protect computers from exploitation. Intrusion prevention technology is considered by some to be an extension of IDS technology but it is actually another form of access control, like an application layer firewall. IPS have many advantages over IDS: (i) they are designed to sit inline with traffic flows and prevent attacks in real-time; (ii) most IPS solutions have the ability to look at (decode) layer 7 protocols like HTTP, FTP, and SMTP, which provides greater awareness. There are several types of IPS: Host based (HIPS), one where the intrusion-prevention application is resident on a specific IP address, usually on a computer; Network based IPS (NIPS) is one where the IPS application/hardware and any actions taken to prevent an intrusion on a specific network host is done from a host with another IP address on the network; Content based IPS (CIPS) inspects the content of network packets for unique sequences, called signatures, to detect and hopefully prevent known types of attacks such as worm infections and hacks; Rate based IPS (RIPS) are primarily intended to prevent denial of service (DoS) and distributed DoS attacks and work by monitoring and learning normal network behaviors - through real-time traffic monitoring and comparison with stored statistics, RIPS can identify abnormal rates for certain types of traffic (for example TCP, UDP or ARP packets, connections per second, packets per connection, packets to specific ports); Protocol analyzer IPS (PAIPS) is an IPS that uses a protocol analyzer to fully decode protocols - once decoded, the IPS analysis engine can evaluate different parts of the protocol for anomalous behavior or exploits and the IPS engine can drop the offending packets.

Snort (www.snort.org) is an open source network intrusion prevention and detection system utilizing a rule-driven language, which combines the benefits of signature, protocol and anomaly based inspection methods. Snort is the most widely deployed intrusion detection and prevention technology worldwide and has become the *de facto* standard for the industry. Snort is capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, Common Gateway Interfaces (CGI) attacks, Server Message Blocks (SMB) probes, OS fingerprinting attempts, amongst other features. The system can also be used for intrusion prevention purposes, by dropping attacks as they are taking place.

OSSEC (www.ossec.net) is an open source HIDS that performs log analysis, integrity checking, Windows registry monitoring, rootkit detection, real-time alerting and active response. It runs on most operating systems, including Linux, OpenBSD, FreeBSD, MacOS, Solaris and Windows.

SamHain (www.la-samhna.de/samhain) is a multiplatform, open source solution for centralized file integrity checking and/or host-based intrusion detection on POSIX systems (Unix, Linux, Cygwin/Windows). It has been designed to monitor multiple hosts with potentially different operating systems from a central location, although it can also be used as standalone application on a single host.

Osiris (osiris.shmoo.com) is a HIDS that periodically monitors one or more hosts for change. It maintains detailed logs of changes to the file system, user and group lists, resident kernel modules, and more. Osiris can be configured to email these logs to the administrator. Hosts are periodically scanned and, if desired, the records can be maintained for forensic purposes. Osiris keeps an administrator apprised of possible attacks and/or nasty little trojans. The purpose here is to isolate changes that indicate a break-in or a compromised system. Osiris makes use of OpenSSL for encryption and authentication in all components.

Cfengine (www.cfengine.org) is one of the most powerful system administration tools available today. In a useful deviation from most scripting tools, cfengine allows describing the desired state of a system rather than what should be done to a system. Cfengine itself takes care of testing compliance with that state and will do its best to correct any misconfigurations. It also includes powerful classing capabilities that allows grouping hosts into classes and create different states on each class of host. Like all tools, it has its drawbacks, but overall it should be considered the most important and most capable tool in the sysadmin toolbox today.

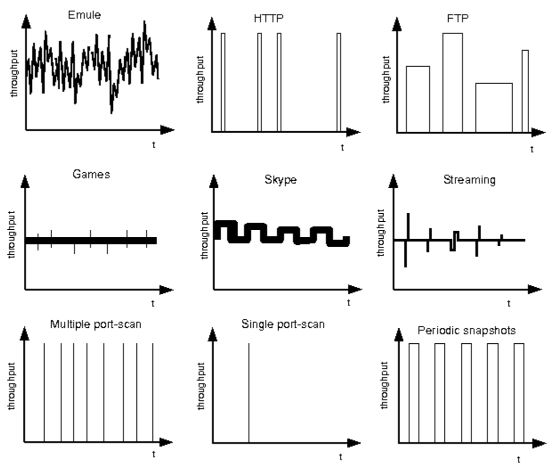

Figure 1: Traffic profiles for different Internet applications.

Nessus (www.nessus.org) is a comprehensive vulnerability scanning program. Its goal is to detect potential or confirmed weaknesses on the tested machines, like for example: (i) vulnerabilities that allow a remote cracker to control the machine or access sensitive data; (ii) misconfiguration (e.g. open mail relay); (iii) un-applied security patches, even if the fixed flaws are not exploitable in the tested configuration; (iv) default, common and blank/absent passwords; (v) denials of service against the TCP/IP stack.

3 DETECTION FRAMEWORK

It is known that Internet applications usually exhibit characteristic traffic profiles, as illustrated in Figure 1:

- File sharing applications use high bandwidths and are characterized by a high variability that can be attributed to the enormous number of TCP sessions that are opened/closed;
- HTTP traffic is characterized by short periods of high bandwidth utilization with non-periodic durations, intercalated by large periods of inactivity;
- FTP traffic also exhibits a high bandwidth utilization, since established connections tend to occupy all available resources. However, high activity periods are intercalated with inactivity periods;
- Games traffic is characterized by a medium level continuous bandwidth utilization pattern, with periodic peaks;
- Skype traffic has an almost periodic pattern - since it uses VoIP technology the bandwidth utilization pattern should be periodic, but the adjustment of the transmission parameters to the delay constraints imposed by the underlying network induces some variability on the bandwidth utilization profile;
- Streaming traffic is characterized by a low/medium level and almost constant utilization profile with some negative and positive short non-periodic variations from the average bandwidth. These variations can be explained by bandwidth starvation and the consequent bandwidth increment imposed by the streaming protocol with the purpose of achieving an average transfer rate.

The proposed detection framework will use this knowledge about different application profiles to build a ”memory” that will allow further identifications for new traffic traces that are presented to the framework. Besides ”regular” or licit Internet applications, the detection framework must also have a complete knowledge of the traffic profiles associated to tasks or actions that are commonly performed by controlled or compromised machines, so it can search for and identify those profiles in a real operation scenario. In order to obtain the typical traffic profiles associated to compromised machines, we had to mimic the behavior of a compromised PC and the SubSeven rootkit was chosen to generate the intended “illicit” traffic traces.

A rootkit is a set of software tools intended to conceal running processes, files or system data from the operating system. They have their origin in benign applications, but recently have been used by malware to help intruders maintain access to systems while avoiding detection. Rootkits often modify parts of the operating system or install themselves as drivers or kernel modules and can take full control of a system. A rootkit's only purpose is to hide files, network connections, memory addresses, or registry entries from other programs used by system administrators to detect intended or unintended special privilege accesses to the computer resources. However, a rootkit may be incorporated with other files which have other purposes. Rootkits are often used to abuse a compromised system and often include so-called ”backdoors” to help the attacker subsequently access the system more easily. A backdoor may allow processes started by a non-privileged user to execute functions normally reserved for the superuser. All sorts of other tools useful for abuse can be hidden using rootkits: this includes tools for further attacks against computer systems which the compromised system communicates with, such as sniffers and keyloggers. Rootkits are also used to allow its programmer to see and access user names and log-in information for sites that require them.

The SubSeven backdoor is a Trojan Horse that enables unauthorized people to access a computer over the Internet without the knowledge of its owner. When the server portion of the program runs on a computer, the individual who is remotely accessing the computer may be able to perform the following tasks: (i) set it up as an FTP server; (ii) browse files on that system; (iii) take screen shots; (iv) capture real-time screen information; (v) open and close programs; (vi) edit information in currently running programs; (vii) show pop-up messages and dialog boxes; (viii) hang up a dial-up connection; (ix) remotely restart a computer; (x) open the CD-ROM; (xi) edit the registry information.

In our simulation experiments, we have programmed three tasks on the SubSeven software: file transfer, multiple port scan and periodic snapshots. The file transfer bandwidth utilization profile is similar to the FTP profile previously discussed and the profiles associated to multiple port scan and periodic snapshots are also illustrated in Figure 1: the first one is characterized by several very short duration high activity peaks and the second is characterized by periodic periods of high activity, each one demanding constant bandwidth utilization.

The proposed framework for detection of illicit traffic is based on neural networks, specifically a feed-forward network model using the back propagation learning algorithm. Back propagation is a general purpose learning algorithm for training multilayer feed-forward networks that is powerful but expensive in terms of computational requirements for training. A back propagation NN model uses a feed-forward topology, supervised learning, and the back propagation learning algorithm. A back propagation network with a single hidden layer of processing elements can model any continuous function to any degree of accuracy (given enough processing elements in the hidden layer) (Demuth and Beale, 1998).

For the dimension of our problem a feed-forward back propagation network with three layers is appropriate. The correlation that exists between the temporal sequence of traffic values and the current distribution of traffic per application is taken into account by presenting the current and the last h (where h represents a configurable parameter) traffic values as inputs to the NN model. In this way, the input layer will have $h+1$ neurons, corresponding to the dimensionality of the input vectors. We have tried different values of h, concluding that the best performances were obtained for NN models having between 4 and 16 neurons in the input layer. The number of nodes of both the input and hidden layers are empirically selected such that the performance function (the mean square error, in this case) is minimized. The output layer has 1 neuron, since each output vector represents the existence of licit or illicit traffic.

The application of a NN model to solve a particular problem involves two phases: a training phase and a test phase. In the training phase, a training set is presented as input to the NN which iteratively adjusts network weights and biases in order to produce an output that matches, within a certain degree of accuracy, a previously known result (named target set). In the test phase, a new input is presented to the network and a new result is obtained based on the network parameters that were calculated during the training phase. There are two learning paradigms (supervised or non-supervised learning) and several learning algorithms that can be applied, depending essentially on the type of problem to be solved. For our problem the network was trained incrementally, that is, network weights and biases were updated each time an input was presented to the network. This option was mostly determined by the size of the training set: loading the complete training set at once and presenting it as input to the NN was very consuming in terms of computational memory. The training method used was the Levenberg-Marquardt (Madsen et al., 2004) algorithm combined with automated Bayesian regularization, which basically constitutes a modification of the Levenberg-Marquardt training algorithm to produce networks which generalize well, thus reducing the difficulty of determining the optimum network architecture.

The general approach used for the identification of illicit traffic is depicted in Figure 2: in the training phase, the first half of the different application traces, that are designated as application profiles, are presented to the neural network model, together with a previous classification of the different profiles. This phase leads to a trained neural network model whose parameters were conveniently adjusted according to the identification requirements. In the test phase, the second half of the application profiles are inputted to the trained neural network model, producing the corresponding identification results. By comparing these results with the pre-identification values, the model can be conveniently validated.

4 RESULTS

As previously explained, the neural network model is trained with the first half of each traffic trace and the second half will be used to test the accuracy of the trained model. The following applications were used

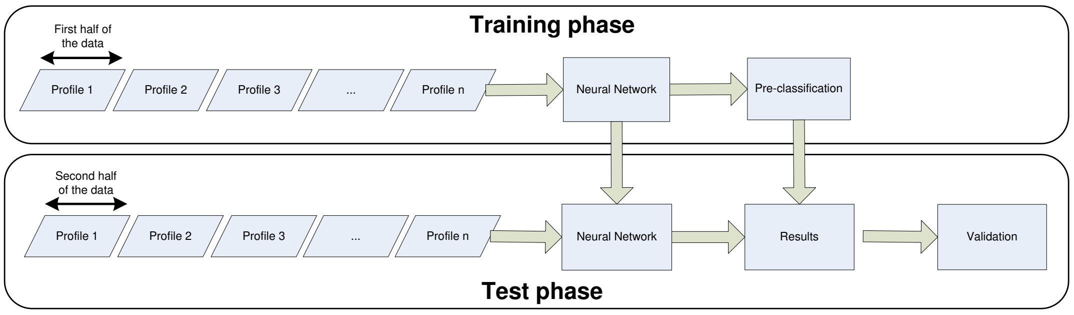

Figure 2: General approach for the detection of illicit applications.

Table 1: Correct identification percentages for the different traffic profiles - upload traffic.

Traffic type	*h*	# Neurons in hidden layer	Correct identification
HTTP + Port scan	10	16	98.26%
HTTP + Snapshots	16	20	100.00%
HTTP + File Transfer	8	16	94.49%
Streaming + Port scan	8	16	93.91%
Streaming + Snapshots	16	14	97.98%
Streaming + File Transfer	10	16	98.41%
HTTP + File Sharing + Skype + (Snapshots or File Transfer)	16	12	88.80%

to evaluate the efficiency of the proposed methodology: HTTP, Games, Peer-to-Peer File Sharing, Video Streaming and Skype. Besides these licit applications, illicit traffic was synthetically generated by programming SubSeven to perform file transfer, multiple port scan and periodic snapshots (with a 2 seconds periodicity), as described in the previous section. By combining these applications, the following traffic profiles were created: (i) HTTP Browsing + Port Scan; (ii) HTTP Browsing + Periodic Snapshots; (iii) HTTP Browsing + File Transfer; (iv) Streaming + Port Scan; (v) Streaming + Periodic Snapshots; (vi) Streaming + File Transfer; (vii) HTTP Browsing + P2P File Sharing + Skype + (Snapshots or File Transfer). Only the upload traffic will be considered and each trace has 1-hour duration and represents the number of upload bytes per sampling interval (the sampling interval is equal to 1 second).

Figure 3 depicts 5 minute samples of the HTTP browsing traces. Figure 3(a) shows a licit HTTP browsing typical profile with short non-periodical peaks generated by HTTP requests. The HTTP browsing and a non-aggressive port scan traffic joint profile is depicted in 3(b). It is possible to observe small periodic peaks characteristic of the port-scan embedded within the normal licit traffic. Figure 3(c) shows the traffic profile of periodic snapshots transferred to a remote machine characterized by periodic bursts of traffic in upstream direction. Figure 3(d) depicts a transference of a large file in the upstream direction. This type of traffic may not be illicit because it can be the result of a normal HTTP file upload. Nevertheless, a HTTP browsing trace should not include file uploads and we chose to classify these events as an anomaly that should be detected.

Figure 4 depicts 5 minute samples of the streaming traces and the same illicit traffic profiles, described above for the HTTP browsing traces, can be observed embedded now within a streaming trace. The streaming trace is characterized by a constant (in average) bandwidth occupation. The profile has some negative and positive short non-periodic variations from the average bandwidth. These variations can be explained by bandwidth starvation and the consequent bandwidth increment generated by the streaming protocol to counterbalance it and achieve an average constant rate.

Figure 5 depicts 5 minute samples of the joint HTTP Browsing + P2P File Sharing + Skype traces. The licit traffic is depicted in Figure 4(a) and Figures 5(b) and 5(c) show the licit traffic in conjunction with snapshot and file transfer traffic, respectively. The licit traffic is characterized by a high bandwidth consumption with a relative high variation around an average bandwidth value. We chose not to include a non-aggressive port scan because such traf-

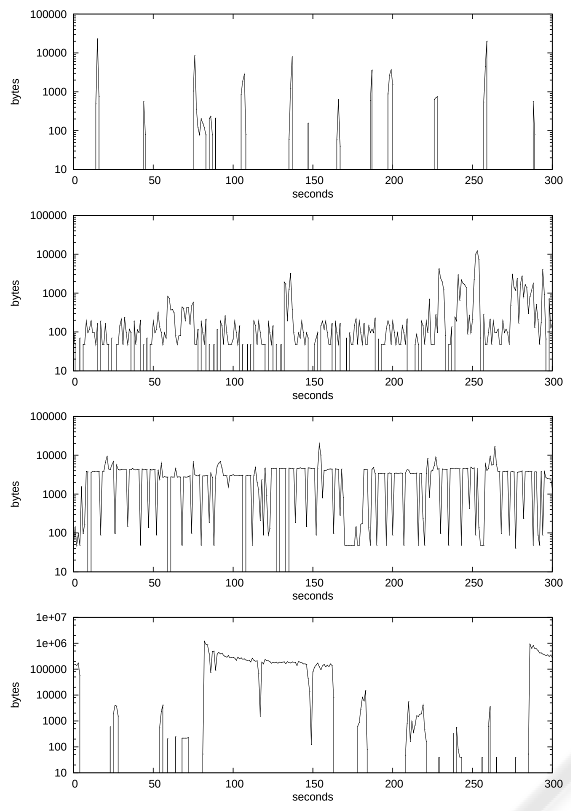

Figure 3: HTTP Browsing (from top to bottom): (a) Licit, (b) Licit + Port Scan, (c) Licit + Snapshot and (d) Licit + File transfer.

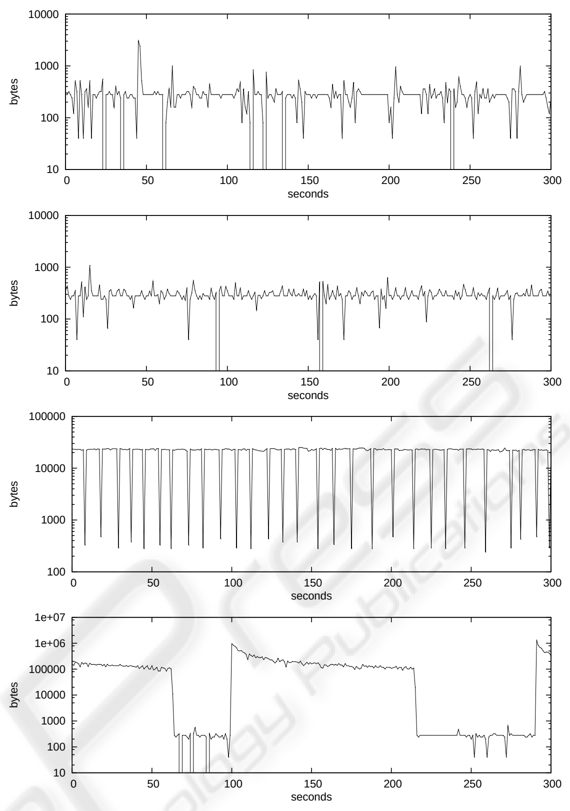

Figure 4: Streaming (from top to bottom): (a) Licit, (b) Licit + Port Scan, (c) Licit + Snapshot and (d) Licit + File transfer.

fic is imperceptible within such high licit bandwidth consumptions. The tested trace with illicit components was composed by an alternate usage of snapshot and file transfer (10 minutes each time) embedded in the licit traffic.

For each class of licit traffic, one neural network was trained to detect the presence of illicit traffic profile signatures and identify the specific class of illicit traffic. Table 1 presents the best correct identification percentages for the different traffic profiles, as well as the history length that was considered at the input of the neural network (that is, the amount of temporal correlation that was taken into account in the input traffic profiles) and the number of neurons in the hidden layer of each neural network model. As can be seen from these results, the identification percentages are quite high, illustrating the accuracy of the proposed methodology for all considered traffic scenarios. The worst result was obtained for the last traffic profile, since it is the most heterogeneous profile that includes a quite complicated mixture of applications. Nevertheless, the trained model was able to correctly identify more than 88% of the traffic values.

5 CONCLUSIONS

This paper presented a new approach for the identification of illicit traffic that tries to overcome some of the limitations of existing approaches, while being computationally efficient and easy to deploy. The approach is based on neural networks and is able to detect illicit based on the historical traffic profile presented by each licit and illicit network application. The test of the proposed methodology was based on traffic generated on a controlled environment, where malicious traffic was generated using the SubSeven backdoor. The results obtained show that the proposed methodology is able to achieve good identification results, being at the same time computationally efficient and easy to implement in real network scenarios.

ACKNOWLEDGEMENTS

This work was done under the scope of the Euro-NGI and Euro-FGI Networks of Excellence - De-

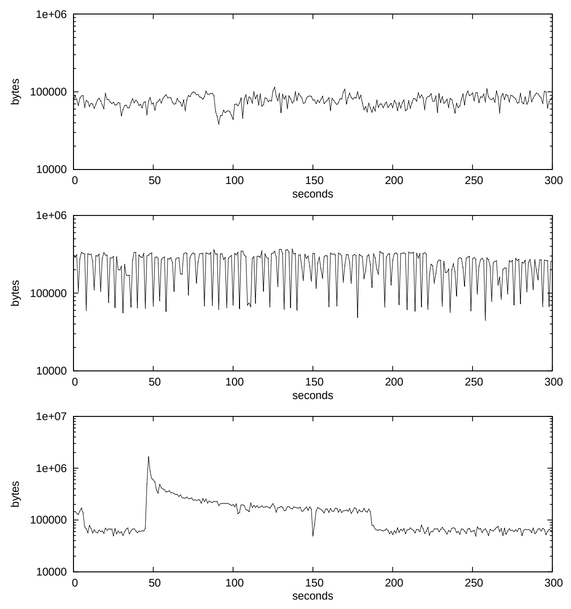

Figure 5: HTTP Browsing + P2P File Sharing + Skype (from top to bottom): (a) Licit, (b) Licit + Snapshots and (c) Licit + File Transfer.

sign and Engineering of the Next Generation Internet, and Euro-NF Network of Excellence - Anticipating the Network of the Future (from Theory to Design), funded by the European Union.

REFERENCES

Chen, P.-T. and Laih, C.-S. (2008). IDSIC: an intrusion detection system with identification capability. *International Journal of Information Security*, 7(3):185–197.

Debar, H., Becker, M., and Siboni, D. (4-6 May 1992). A neural network component for an intrusion detection system. *Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on*, pages 240–250.

Demuth, H. and Beale, M. (1998). *Neural Network Toolbox Users Guide*. The MathWorks, Inc.

Denning, D. (1987). An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, 13(2):222–232.

Ilgun, K., Kemmerer, R., and Porras, P. (1995). State Transition Analysis - A Rule-Based Intrusion Detection Approach. *IEEE Transactions on Software Engineering*, 21(3):181–199.

Jiang, W., Song, H., and Dai, Y. (2005). Real-time intrusion detection for high-speed networks. *Computers & Security*, 24(4):287–294.

Madsen, K., Nielsen, H., and Tingleff, O. (2004). *Methods for Non-Linear Least Squares Problems*. Technical University of Denmark, 2nd edition.

Mukherjee, B., Heberlein, L., and Levitt, K. (1994). Network Intrusion Detection. *IEEE Network*, 8(3):26–41.

Ryan, J., Lin, M.-J., and Miikkulainen, R. (1997). Intrusion detection with neural networks. In *NIPS '97: Proceedings of the 1997 conference on Advances in neural information processing systems 10*, pages 943–949, Cambridge, MA, USA. MIT Press.

Wang, K. and Stolfo, S. (2004). Anomalous Payload-based Network Intrusion Detection. *Recent Advances in Intrusion Detection, Proceedings*, 3224:203–222.