Emulation Environment for Ground Truth Establishment

Carlos Miranda, Paulo Salvador, António Nogueira, Eduardo Rocha
University of Aveiro / Instituto de Telecomunicações, Campus de Santiago, 3810-193 Aveiro, Portugal
E-mail:{cmiranda, salvador, nogueira, erocha}@av.it.pt
Rui Valadas
Instituto Superior Técnico - UTL / Instituto de Telecomunicações, Av. Rovisco Pais, 1049-001 Lisboa, Portugal
E-mail:rui.valadas@ist.utl.pt

***Abstract*—Network security is a hot topic to network users and managers, whether they are institutional, enterprise or domestic. New threats or mutations of existing ones appear at a very fast rate and the solutions that are nowadays used to fight them frequently require a real-time analysis of the network traffic or a previous training based on real data. Most of the times, this training must be supervised by humans that, depending on their experience, can create security breaches in the system without knowing it. Since existing anomaly detection methodologies have to be trained and tested in order to validate their efficiency, there is an increasing need for trustworthy network traffic data that can be used without compromising users confidentiality, obeys to some pre-established criteria and is completely known in terms of its underlying protocols. In fact, the effectiveness of network anomaly detectors cannot be fully evaluated without having a complete control of the entire evaluation experiment, which requires that it should be possible to change the location, magnitude and type of individual anomalies and background traffic. In this work, we propose an emulation environment that can be used to obtain trustworthy network data both in the presence of licit and illicit applications. We also present some topological and traffic scenarios that were already defined to start gathering network data and make it immediately available to the scientific community. The emulation environment was built in an evolutionary way, enabling the easy introduction of new network scenarios and services and/or the refinement of the existing ones.**

***Keywords*-Ground truth, emulation, licit and illicit applications.**

I. Introduction

Most of the research activities on network traffic modeling, application identification and anomaly detection methodologies require the association of application and protocol ground truth information with traffic traces [1]. There are several approaches that can be used to link ground truth data to Internet traffic traces. The first approach is to create a trace manually by instantiating a realistic pool of applications on many machines, but in this case the captured traffic typically lacks characteristics that only human intervention can induce. The second methodology is to record traffic on a live network and perform deep packet inspection to each packet in order to identify its underlying generating protocol/application [2], [3], [4], [5]. However, deep packet inspection is ineffective when traffic is encrypted and ambiguous when different protocols exhibit similar signatures. Finally, the last approach works by probing a set of monitored host kernels in order to obtain information on active Internet sessions, thus gathering ground truth at the application level [6]. However, this methodology is intrusive and requires the agreement of the different monitored users.

In order to circumvent some of these limitations, we propose in this paper an emulation approach that can be used to generate trusty network data. Basically, we have created a flexible emulation environment that can be used to obtain trustworthy data for traffic scenarios that include both licit and illicit applications. The details of the emulation platform will be presented in this paper, as well as the main development options that were adopted. Besides, the paper will also present the topological and traffic scenarios that were already defined to start the process of gathering trusty network data in order to make it immediately available to the scientific community.

By defining the topologies of the scenarios, the protocols to be used and the anomalies that will be ”injected”, a vast amount of trustworthy data can be obtained. We can see the ”before and after” the injection of a specific anomaly in order to see the behavior over the network, while having a high accuracy in the obtained data without compromising privacy. Simulators and emulators are a very effective and cheap way to create environments that otherwise could be very expensive to deploy with real equipment. They are also very useful to test new networking protocols or to change existing ones in a controlled and reproducible manner, saving a lot of time and money. Currently, there is a huge variety of simulators and emulators, ranging from payware to freeware solutions, destined to educational or to research purposes. Network emulators work in a similar way as simulators. However, emulators are able to run in real time while network simulators are not. In network emulators, the network behaves like a real network and it is possible to monitor, measure and test its performance. Real equipment can be connected to the emulation (computers, routers and other network entities), behaving exactly as if they were in a real network. Thus, no approximations are necessary to describe real processes, like routing and queuing mechanisms for example.

The emulation methodology that is proposed in this paper is completely generic, enabling the integration of automatic traffic generation tools with the normal human usage of different network services. In this way, we are able to capture

 ISBN: 978-1-61208-103-8

the different components and nuances of the network traffic characteristics.

The rest of the paper is organized as follows. Section II will briefly present the most important approaches to ground truth establishment that have been published so far; Section III will describe the emulation framework that was created and the network and service scenarios that were already defined, including their main configuration steps; Section IV describes the data that can be collected from the emulated scenarios and, finally, Section V presents the main conclusions of this work and points out some topics for future research.

II. Related work

In the context of network anomaly detection, ground truth requires a complete list of all anomalies existing in the data traces. Some previous works [4], [7] have injected anomalies in real traces taken from operational networks but this technique cannot provide truthful data because it relies on existing traces and there are no guarantees that they are free of anomalies and, if they are not, on the number and type of anomalies they include.

Several other techniques have been used to obtain trusty data. Payload inspection and port-based mechanisms [8], [9], [10], [11], [12] have been recurrently used to establish a form of protocol ground truth but, due to their uncertainty, they can only provide an estimate of the protocol that is being used. Besides, port-based analysis became quite an obsolete technique for protocol identification.

Manual generation of network traffic can provide ground truth at the application level [13] but the presence of background applications can lead to the generation of additional traffic that can not be accurately tagged. However, this approach is able to deal with encrypted traffic, which is one of its main advantages.

The approach presented by Szabo et al. [14] offers an advanced approach in application detection and tagging by embedding ground truth information directly into IP packets. However, this approach creates some methodological problems (like lowering the precision of the recorded traces or being unable to mark packets with sizes closer to the MTU), their implementation is Windows XP-specific and the created tool provides application but no protocol information for a given flow.

Trestian et al. [15] describe an heuristic that combines information freely available on the web (through Google) to retrieve the class of applications that generated a specific flow. This approach is interesting but it relies on external resources.

In [16], the authors present a platform, named Ground Truth Verification System (GTVS), that uses a combination of heuristics at different levels (host, flow, packet) to improve the quality of ground truth associated with packet traces, but does not include application labels with guaranteed accuracy.

Finally, in reference [6] the authors present a new mechanism to provide ground truth at the application level. The architecture of the proposed tool is based on a client tool that, by monitoring the kernel of a host, associates each packet flow with the name of its controlling application and transmits the collected information to a back-end. The post-processing toolset analyzes the traffic captured at the network border by an independent probe and associates each flow with its application label, being able to establish ground truth for that flow. This tool works on many operating systems and it is freely available under an Open Source (BSD) license [17]. However, this approach relies on monitoring the kernels of the different network hosts, which can be an intrusive task and requires the agreement of the corresponding network users.

III. Emulation environment

The main objective of this work is to create a real-time emulator environment that can be used to obtain trustworthy traffic traces and network data, without having to concern on all issues related to the usage of public traces. In this type of environment, the entire emulation process can be controlled, thus achieving a higher confidence degree on the collected data. The Graphical Network Simulator, version 3 (GNS3), was the chosen network emulator.

GNS3 [18] is a multi-platform, open-source Graphical Network Emulator primarily developed by Jeremy Grossman. GNS3 allows the emulation of complex network topologies by emulating many Cisco IOS router platforms, IPS, PIX and ASA firewalls, and JunOS with the help of Dynamips and Dynagen. Dynamips is the core program behind the emulation process and the Dynagen tool runs on top of it to create a user-friendly, text-based environment. GNS3 provides the graphical front-end for Dynagen, so that users can create the topologies in a graphical and user-friendly environment. GNS3 also allows the emulation of ATM and Frame Relay switches, enables packet capture using Wireshark and, most important, allows the connection of the emulated network to the real world. So, using this flexible tool, it is possible to create network scenarios with the desired degree of complexity.

A. Topological scenarios

Two network scenarios were already defined and emulated. The first scenario, illustrated in Figure 1, represents the interaction between two main routers of two departments of a small enterprise network and a backbone router that is connected to a server. The departments can also interact between them and if one of the links between the three routers goes down, there is a backup route passing through the link of the other department, so that access to the server is guaranteed. The clouds represent the LANs located inside each department. For this scenario, the LANs have a total of five emulated end-hosts performing random requests to the server and an also emulated end-host that can be connected in any part of the network. This movable end-host is used only for triggering security attacks to the network. By using this end-host, which is fully dedicated to network attacks, it is possible to easily identify the packets generated by the attacks and analyze them separately from the rest of the network traffic that is generated by the emulation scenario.

In this scenario, the end-hosts can send DHCP requests to obtain their IP address and can also send random HTTP and FTP requests to the server. The emulated routers belong to the

 ISBN: 978-1-61208-103-8

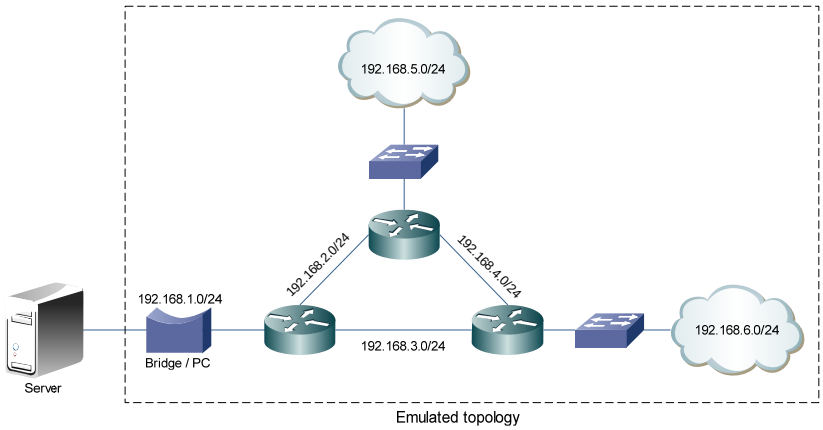

Fig. 1. Network Topology for Scenario A.

TABLE I
SERVICES CONFIGURED ON SCENARIO A

Routers	Server
DHCP Server	Apache
OSPF	BIND9
SNMP Agent	ProFTPD
	NMS

3745 series. The services that were configured at the routers and the server are listed in Table I.

This simple scenario was mainly used to test the Personal Computers (PCs) that host the emulated devices in order to look for problems in the emulation and to check if these hosts were able to handle all the processing that is needed for the emulation to run. By capturing only the first 100 bytes of each packet, this scenario generates, approximately, 5GB of network traffic per day.

The topology of scenario B, represented in Figure 2, represents the network infrastructure of a medium/large enterprise network. The network backbone part of this topology is composed by four real Catalyst 3750 Metro series layer 3 switches (IOS version 12.1(14r)) and three emulated 3745 Cisco routers. One of the layer 3 switches is directly connected to the server in order to allow communication between each point of the network and the server. In this topology, each part of the network is composed by three departments that are connected to the server via different points of the backbone network. Each router, representing the connection to a department network, holds a LAN composed by 6 end-hosts where 5 of them generate normal network traffic (DHCP, HTTP, FTP, SMTP and POP3 requests) and the sixth one is used to trigger network security attacks. The services that were configured at the routers and the server are the same of the previous scenario, except that now the server additionally includes the POSTFIX service. This scenario can generate 40GB of network traffic per day, capturing only the first 100 bytes of each packet.

B. *Traffic scenarios*

The above network topologies were already tested using different network traffic scenarios, each one having a 24 hours duration. These scenarios can be divided into three main groups:

- Clean scenario: this scenario generates network traffic without anomalies (that is, using only licit applications);
- Port-Scan Attack scenario: the network traffic generated in this scenario is a mixture of normal network traffic with port-scan security attacks made from a single or multiple end-hosts to the server;
- Snapshot Attack scenario: the network traffic generated in this scenario is a mixture of normal network traffic with snapshot security attacks made from a single end-host to the server.

The Port Scan attack is made using Nmap (Network Mapper) [19], an open source tool used for network exploration and security auditing. Nmap uses raw IP packets to obtain several characteristics of the scanned host, like the services that are offered by the host, the OS that is running on it or the type of firewall that is used. Nmap also retrieves a list with the used port numbers, the service that is running on those ports and their states. Port scans are normally used by hackers to discover open ports on the target end-host in order to gain access to it or to trigger Distributed Denial of Service (DDoS) attacks.

The Port-Scan attack traffic scenario is further sub-divided into three types of port scans, depending on the timing profile that is used:

- Sneaky: used for Intrusion Detection System (IDS) evasion. A complete port scan with this timing template can take up to five hours to complete because scans are not made in parallel and the waiting time between each probe that is sent to the server is 15 seconds.
- Normal: this is the default timing template used by Nmap. It allows the port scan parallelization process.
- Aggressive: in this case, the scan delay of the probes does not exceed 10 milliseconds (for TCP ports).

For each type of Port Scan attack that is used in this work there is also a division of the different types of attacks based

 ISBN: 978-1-61208-103-8

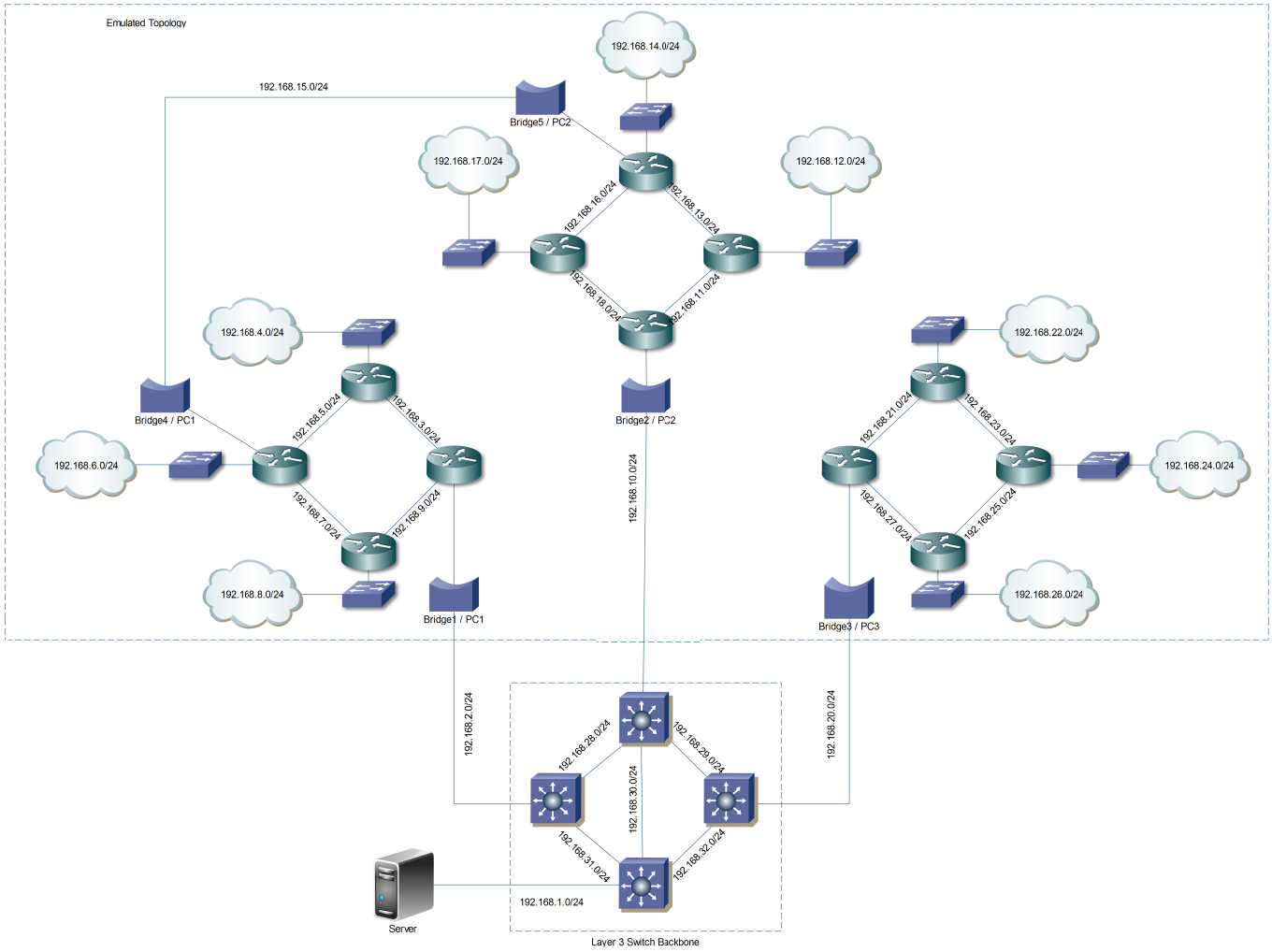

Fig. 2. Network Topology for Scenario B.

TABLE II
DIFFERENT TYPES OF PORT SCAN ATTACKS BASED ON THE NUMBER OF ATTACKERS AND TARGETS.

Port Scan attack	Description
1-to-1	Single attacker against a single target.
1-to-N	Single attacker against multiple targets.
N-to-1	Multiple attackers against a single target.
N-to-N	Multiple attackers against multiple targets.

on the number of attackers and targets. This division and a brief description of each sub-type are given in Table II.

The Snapshot Attack scenario, as the name suggests, consists of performing snapshots of the target monitor content. These images are then uploaded to a destination of the attacker preference and can be used to discover confidential information about the target. Some ”worms” that can be found on the Internet act like this: while consuming the target resources, they can ”catch up” snapshots of the target. In this scenario, the snapshot attack is divided into two categories:

- Exponential: in this attack, each time a users ”clicks” in some part of the monitor, a snapshot of a small area around the pointer is taken. The name of this category derives from the fact that intervals between user clicks are exponentially distributed, in order to model human behavior in the best possible way.
- Periodical: in this kind of attack, a snapshot of the entire screen is taken in periodical time intervals.

Because these attacks are uploads of relatively small images, in this work this attack is emulated by using a virtual end-host inside the emulation that uploads image files to the server using an exponentially distributed time interval, in the first case, and a periodic time interval in the second case. The tool that was used to emulate this behavior (and also for the FTP download and upload traffic that will be explained in the following sub-section) is called cURL, a command line tool for transferring files with a URL syntax. It currently supports FTP, HTTP, FTPS, HTTPS, TELNET and many others protocols that use the URL syntax.

 ISBN: 978-1-61208-103-8

C. Server configuration

Three services are implemented at the server: HTTP, FTP and Email (POP3 to retrieve mail messages and SMTP to send them). In order to have these services, the Network Interface Card (NIC) of the server is configured with one real IP address and several Virtual IP Addresses (VIPA), in such a way that the DNS server will be associated to the real IP address and the first VIPA will be associated with the domain configured for the APACHE, ProFTPD and POSTFIX servers. The other VIPAs are used for testing part of the attacks against multiple IP addresses that are inside the server IP range in the emulated scenario. All server IP addresses are statically configured.

D. Network device configuration

The routers' interfaces have to be configured as DHCP servers for their LANs, so that the end-hosts can obtain their IP addresses via DHCP. All routers involved in the emulation (real and emulated) were configured with the OSPF protocol, so that they can construct their routing tables and the topology map of the network. Finally, in order to gather the MIB information about all the interfaces of the different network devices that are present in the emulation, SNMP Agents were enabled at each device. For example, the number of incoming, outgoing and dropped packets per router interface is some of the information that can be gathered using SNMP agents.

E. PC configuration

In order to emulate a multiple end-host environment, several TAP interfaces were configured in each one of the host PCs. A TAP interface is a virtual Ethernet interface created in the system kernel that can behave as a real Ethernet interface and can be connected to the emulated topology in order to generate all the traffic requests for the emulated scenarios.

In the first tests that were made, we noticed that the only responses received from the server were those corresponding to the requests that were made by the first end-host. This is due to the fact that the Linux routing table establishes priorities for the configured routes. In order to solve this problem, the routes for each end-host need a set of rules for incoming and outgoing traffic, thus forming a routing table for each one of them. When packets arrive, the system will look for matching the set of rules in order to properly deliver the packets to the end-hosts. Besides, the *arp_filter* file corresponding to each interface (the interfaces of the end-hosts and the bridge), found in the "/proc/sys/net/ipv4/conf/interface_name/" directory, needs to be enabled so that all the interfaces' ARP requests can be correctly answered.

F. Operational procedure

After creating all the necessary virtual end-hosts and bridges, configuring the default route at the server and connecting all emulated and real network devices, the following procedure must be followed to start the emulation:

- Start the emulation inside GNS3 and configure all the network devices (emulated and real).

```
192.168.4.2 - - [21/Sep/2009:14:33:45 +0100] "GET /CNN.htm HTTP/1.0" 200 103483 "-" "Wget/1.11.4"
192.168.4.2 - - [21/Sep/2009:14:33:48 +0100] "GET /CNN.htm HTTP/1.0" 200 103483 "-" "Wget/1.11.4"
192.168.4.2 - - [21/Sep/2009:14:33:48 +0100] "GET /CNN.htm HTTP/1.0" 200 103483 "-" "Wget/1.11.4"
192.168.4.2 - - [21/Sep/2009:14:33:56 +0100] "GET /CNN.htm HTTP/1.0" 200 103483 "-" "Wget/1.11.4"
192.168.4.5 - - [21/Sep/2009:14:34:04 +0100] "GET /cyberhome.htm HTTP/1.0" 200 25247 "-" "Wget/1.11.4"
```

Fig. 3. APACHE log file.

- Run the DHCP script file so that all end-hosts can obtain their IP addresses.
- Configure the different routing tables and rules for each virtual end-host.
- Start the *tshark* application at the server in order to capture all the packets and execute the SNMP script file for gathering the MIB data.
- At the host PC, start the *tshark* application to capture all packets in all virtual interfaces.
- Execute the HTTP, FTP, SMTP and POP3 traffic generators.
- For the scenarios that include security attacks, execute the script file corresponding to the attack that is going to be launched.

It was assumed that a typical user makes a new HTTP or FTP request in time intervals of 120 seconds, in average. This value will be used by the emulation tool as the mean of the exponential distribution that will rule the timing profiles of the HTTP and FTP requests. It was also assumed that a typical user sends an E-mail every 10 minutes [20], [21], in average. This value is used to create the exponential distribution that rules the generation process of the SMTP requests. The average time interval used in this work to check for new E-mail messages is 10 minutes. This is the default time value that is used by almost all E-mails clients, like Thunderbird or Microsoft Outlook.

IV. COLLECTED DATA

A lot of data can be gathered from the virtual interfaces (in order to characterize the behavior of single users) or from the server (enabling the characterization of traffic aggregates). As previously explained, virtual interfaces are used in this work with two purposes: the interconnection between real equipment and the emulated topologies and the emulation of end-hosts connected to the network. In real networks, a user can randomly ask for any type of request to the server: for example, a user can see a web page while waiting for the e-mail client to verify his e-mail accounts. The end-hosts used for this work can emulate this behavior with the help of the traffic generator program that was previously explained. So, in the first scenario the end-hosts can obtain their addresses via DHCP, visualize websites and can upload and download files to/from a FTP server; in the second scenario, the end-hosts can also send and receive e mails.

Log files can play a vital role in a server and provide crucial information regarding the services that are running on it. The log files that can be gathered in this framework correspond to the logs of the HTTP (Apache), FTP (ProFTPD) and e-mail (Postfix) servers. Figure 3 shows an example of the APACHE server log file.

 ISBN: 978-1-61208-103-8

```
IF-MIB::ifInOctets.1 = Counter32: 50118
IF-MIB::ifInOctets.2 = Counter32: 54919
IF-MIB::ifInOctets.3 = Counter32: 117948
IF-MIB::ifInOctets.4 = Counter32: 0
IF-MIB::ifInOctets.5 = Counter32: 0
IF-MIB::ifInUcastPkts.1 = Counter32: 407
IF-MIB::ifInUcastPkts.2 = Counter32: 451
IF-MIB::ifInUcastPkts.3 = Counter32: 1328
IF-MIB::ifInUcastPkts.4 = Counter32: 0
IF-MIB::ifInUcastPkts.5 = Counter32: 0
IF-MIB::ifInNUcastPkts.1 = Counter32: 104
IF-MIB::ifInNUcastPkts.2 = Counter32: 111
IF-MIB::ifInNUcastPkts.3 = Counter32: 8
IF-MIB::ifInNUcastPkts.4 = Counter32: 0
IF-MIB::ifInNUcastPkts.5 = Counter32: 0
```

Fig. 4. Example of data collected using SNMP.

To collect SNMP data and store it at the server, the "Net-SNMP" utility (available in the Linux repositories) was installed. This utility allows the communication between SNMP clients and agents. The requests made to the agents are triggered from a terminal, so a bash file was constructed in order to perform this task automatically while emulation was running. Figure 4 shows an example of the data that can be collected using SNMP.

The different traffic profiles that were already collected from the emulation environment and processed *a posterior* are very similar to service profiles obtained from real traffic measurements. Due to lack of space, the comparison plots are not presented in this paper but all collected data will become available on a website that will be used to share all the emulation environment developments and results with the scientific community (the URL of the website is *groundtruth.av.it.pt*).

Regarding the computational requirements of the emulation environment, we can say that for the medium complexity scenarios that were considered in this paper the (quadcore) CPU usage of each one of the PCs hosting part of the topology went up to 40%; in terms of RAM, GNS3 has some memory optimitazion features (the "ghost IOS" and the "sparse memory" featureS) that allowed memory requirements to be kept to a minimum.

V. Conclusions and Future Work

Since most of the research activities on network traffic modeling, application identification and anomaly detection methodologies require the association of application and protocol ground truth information with traffic traces, this paper proposed an emulation environment that can be used to obtain trustworthy data, both in the presence of licit and illicit applications. Besides presenting the details of the created emulation environment, some topological and traffic scenarios that were already used to start gathering network data and make it immediately available to the scientific community were also presented. In order to complement this work, there are some interesting issues that will be addressed in a near future: define and include more complex topologies, emulate more services per interface (P2P file sharing, for example), implement other types of security attacks (DDoS, worm propagation and botnets), study the effects of Spam e-mail, correlate all the gathered data and upgrade the traffic generation tools in order to generate traffic according to different probability distributions and daily patterns.

VI. Acknowledgments

This work was part of the Euro-NF project, funded by the European Union.

References

[1] H. Ringberg, M. Roughan, and J. Rexford, "The need for simulation in evaluating anomaly detectors," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 1, pp. 55–59, 2008.

[2] Marco Canini, Wei Li, Martin Zadnik, and Andrew W. Moore, "Experience with high-speed automated application-identification for network-management," in *Proceedings of the 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'09)*, Oct 2009.

[3] Martin Zadnik, Marco Canini, Andrew W. Moore, David J. Miller, and Wei Li, "Tracking elephant flows in internet backbone traffic with an FPGA-based cache," in *Proceedings of the 19th International Conference on Field Programmable Logic and Applications (FPL'09)*, Aug 2009.

[4] A. Lakhina, M. Crovella, and M. Diot, "Diagnosing network-wide traffic anomalies," in *ACM SIGCOMM*, 2004, pp. 219–230.

[5] H. Ringberg, A. Soule, J. Rexford, and C. Diot, "Sensitivity of PCA for traffic anomaly detection," in *ACM SIGMETRICS*, 2007.

[6] F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso, and K. C. Claffy, "GT: picking up the truth from the ground for internet traffic," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 5, pp. 13–18, 2009.

[7] A. Soule, K. Salamatian, and N. Taft, "Combining filtering and statistical methods for anomaly detection," in *ACM Internet Measurement Conference*, 2005.

[8] Jeffrey Erman, Martin Arlitt, and Anirban Mahanti, "Traffic classification using clustering algorithms," in *MineNet '06: Proceedings of the 2006 SIGCOMM workshop on Mining network data*, New York, NY, USA, 2006, pp. 281–286, ACM.

[9] H. Kim, K. Claffy, M. Fomenkova, D. Barman, and M. Faloutsos, "Internet traffic classification demystified: The myths, caveats and best practices," in *ACM CoNEXT*, 2008.

[10] T. Karagiannis, A. Broido, N. Brownlee, K.C. Claffy, and M. Faloutsos, "Is P2P dying or just hiding?," *IEEE Global Telecommunications Conference (GLOBECOM'04)*, vol. 3, pp. 1532–1538 Vol.3, Nov.-3 Dec. 2004.

[11] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, "Acas: Automated construction of application signatures," in *MineNet '05: Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, Philadelphia, Pennsylvania, USA, August 2005, pp. 197–202, ACM Press.

[12] Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang, "Accurate, scalable in-network identification of P2P traffic using application signatures," in *Proceedings of the 13th international conference on World Wide Web (WWW'04)*, New York, NY, USA, 2004, pp. 512–521, ACM.

[13] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli, "Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting," *Elsevier Computer Networks*, vol. 53, no. 1, pp. 81–97, 2009.

[14] G. Szabo, D. Orincsay, S. Malomsoky, and I. Szabo., "On the validation of traffic classification algorithms," in *Proceedings of the Passive and Active Measurement Conference (PAM'08)*, 2008.

[15] I. Trestian, S. Ranjan, A. Kuzmanovi, and A. Nucci, "Unconstrained endpoint profiling (googling the internet)," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 279–290, 2008.

[16] Marco Canini, Wei Li, Andrew W. Moore, and Raffaele Bolla, "GTVS: Boosting the collection of application traffic ground truth," in *Proceedings of the First International Workshop on Traffic Monitoring and Analysis (TMA'09)*, May 2009.

[17] "The ground truth software tools, available at http://www.ing.unibs.it/ntw/tools/gt," 2009.

[18] "Graphical network emulator - GNS3, available at http://www.gns3.net/," 2009.

[19] "NMAP - network mapper, available at http://nmap.org," 2009.

[20] K. Mandia, C. Prosise, and M. Pepe, *Incident Response & Computer Forensics*, Mcgraw Hill, 2003.

[21] Luiz Henrique Gomes, Cristiano Cazita, Jussara M. Almeida, Virgílio Almeida, and Wagner Meira, Jr., "Characterizing a spam traffic," in *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, New York, NY, USA, 2004, pp. 356–369, ACM.

 ISBN: 978-1-61208-103-8