Practical Network Security: Experiences with ntop

Luca Deri [1 2] and Stefano Suin[2]

[1]Finsiel S.p.A., Via Matteucci 34/b, 56124 Pisa. Email l.deri@finsiel.it
[2]Centro Serra, University of Pisa, Lungarno Pacinotti 43, Pisa, Italy. Email {deri, stefano}@unipi.it

As networks become large and heterogeneous, network administrators need efficient tools for monitoring network activities and enforcing global security. In open environments such as universities and research organizations it is rather difficult to prevent access to core network resources without restricting user's freedom.
ntop is an open-source web-based traffic measurement and monitoring application written by the authors and widely used over the Internet. This paper shows how ntop can also be effectively used for network security as it is able to identify potential intruders and security flaws, as well as discover misconfigured or faulty applications that generate suspicious traffic.

Keywords: traffic monitoring, network security, intrusion detection, TCP/IP.

1. Introduction

Early in 1997 the Centro Serra, responsible for providing network services at the whole University of Pisa, needed an application for monitoring relevant network activities flowing across the campus backbone. Traditional Unix tools for testing basic connectivity problems as well as network sniffers such as *tcpdump* [1] or *snoop* were considered not sufficient. These tools are very powerful for tracking network traffic but they need off-line tools for better analysing and correlating captured data as well as identifying security violations. Other tools for network monitoring such as *RMON* [30] probes and *NeTraMet* [2] offer advanced programming languages for analysing network flows and building statistical event records. Unfortunately, these tools have been designed for analysing well known network flows whereas it is not always easy to guess what network resources will be attacked. Beside some exceptions such as *NFR* (Network Flight Recorder) [3], many security tools available on the Internet are usually designed for detecting attacks against a single host (usually the one where the tool has been activated) and do not provide network/subnet detection/protection nor sport traffic monitoring and measurement facilities. Traffic measurement tools do not usually offer support for security nor allow active actions to be taken when an attack happens but they simply notify the administrators when an attack already took place. This is because measurement tools classify network traffic according to some specified static rules with defined thresholds. These thresholds are often not able to either express complex traffic patterns (e.g. security attack) nor flexible enough to cover a whole subnet without having to define the same rule for all the hosts of the subnet hence to significantly increment the processing time of each received packet. Nevertheless security monitoring is not really effective until actions for facing with the ongoing attack are performed. For instance, if the attack originates from a subnet different than the one of the victim, then the ACL (Access Control List) of the victim's router/firewall [4] could be temporarily instrumented to discard packets originating from the attacker.

After having done a survey of available tools, we decided to write a new application able to both monitor campus traffic and report information about the overall network security. This decision was motivated by the fact that none of the above tools offered all the features we needed while an integration of a few of them would have been too challenging considering that most of the tools were not designed to be plugged together.
ntop is a web-based [5] traffic measurement and monitoring application. It was initially written by the authors for tackling performance problems of the campus network backbone [6] given that available traffic monitoring tools were not satisfactory for the reasons listed before. Similar to the Unix *top* [7] tool that reports processes CPU usage, authors needed a simple tool able to measure network

traffic and report information about captured packets. ntop then evolved into a more flexible, extensible and powerful tool as people over the Internet downloaded it and reported problems and suggestions.
ntop is currently used for traffic measurement and monitoring. This paper describes some security extensions we recently designed for enhancing hence turning it into a sophisticated intrusion detection system [40]. The goal of this paper is to show how ntop can be effectively used both for traffic monitoring and intrusion detection. The following sections cover the ntop architecture, some implementation details and scenarios where ntop can be effectively used for detecting potential attacks and security violations.

2. ntop Architecture

ntop [35] is an open-source [8] application written in C, available free of charge under the GNU public licence, and widely adopted by many people and companies over the world for monitoring their networks. This statement does not just mean that ntop's source code is freely available on the Internet, but also that many requirements came directly from the ntop adopters. The authors designed the first version of ntop and then accommodated new requirements and extensions to the original architecture, strongly influenced by the *Webbin* [9] architecture. ntop's design shares the Unix philosophy: applications do not necessarily have to be large and monolithic [10] but they can profitably be divided into small independent pieces that cooperate for achieving a global goal. Each ntop component is implemented asynchronously by means of a thread and synchronised with other components by means of semaphores. This design solution allows components not to block each other when some potentially long standing actions (e.g. domain name resolution) take place.
The ntop kernel is responsible for handling efficiently basic tasks and providing facilities to plugin developers that can exploit kernel services.
The packet sniffer, capable of handling multiple network interface simultaneously, is based on the *libpcap* [11] library that provides a portable and unified packet capture interface, whereas other operating systems provide proprietary capture facility. Thanks to good design of libpcap, authors decided to port libpcap to non Unix platforms embedding native platform capture facilities (e.g. *NDIS* [12] on Win32) into it. This allowed the ntop source code to be unique across various platforms.
The packet analyser processes one packet at time. Packet headers are analysed according to the network interface being used. Hosts information is stored in a large hash table whose entries contain several counters that keep track of the data sent/received by the host, sorted according to the supported network protocols. For each packet, the hash entry corresponding to packet sender and destination is retrieved or created if not yet present. Since it is not possible to predict the number of different hosts whose packets will be handled by ntop, periodically or if there are no entries left ntop purges the host table in order to avoid exhausting all the available memory and creating huge tables that are then slow to walk.
In order to improve the overall performance and decrease the amount of data that has to be kept in memory, ntop implements a first level caching facility based on GNU *gdbm* [13] whereas second level caching is implemented using a SQL database. First level cache contains semi-persistent information such as IP address resolution (mapping numeric/symbolic IP address) and remote host operating system [14] (computed using the *neped* [15] tool). In order to reduce DNS queries, ntop processes DNS reply packets and caches mappings for future use. Network events (e.g. TCP sessions), performance data and other relevant information are stored permanently into the SQL database. Storage happens either periodically or whenever the garbage collector has to purge some data.
Network flows, specified at ntop start-up time on the command line, are implemented using the *BPF* (Berkeley Packet Filter) [16] facility part of libpcap. BPF allows filters to be specified using simple english-like expressions as those accepted by tcpdump.
Traffic reports are done in both text and HTML. An HTTP server embedded into ntop serves pages to users that can then access traffic reports without having to install and configure an external web server. HTML reports make extensive use of charts implemented using the *GDchart* (http://www.fred.net/brv/chart/) library.

3. Network Security

Requirements for ntop security facilities have been drawn from our experience administering the campus backbone. Beside some exceptions such as administration and accounting, most of the campus hosts are attached directly to the Internet with no firewall protection. The lack of security is necessary as users will not accept restrictions on the use of network services. This means that we have to protect core network resources while allowing external users to access them. In addition, because each department administers its own network independently, we must also make sure that traffic originated by such departments comply with specified security policy. In particular, we have to carefully monitor those departments that statistically originated most of the security attacks.
We decided to run ntop in the core servers network, at the university network border (where the university is attached to the Internet) and in few selected subnets. Because the campus network makes extensive use of switching technology [33], whenever necessary, we mirror VLANs (Virtual LAN) traffic to the switch port where ntop is attached. In general we have noticed that most of the attacks are originated from inside the university, because of lack of security or because some students install software applications that make intruders life much simpler.

We decided to first implement facilities that allow us to identify issues at TCP level (e.g. portscan) and then to add more sophisticated application facilities later on. This is motivated by the fact that in our experience most of the attacks always begin with a scan of the available network resources. Scans can be performed either discovering the network topology and then identifying the victim hosts, or attacking directly core servers such as DNS, mail, and HTTP server whose addresses can either be guessed (e.g. www.<domain name>) or read from the DNS.

ntop provides the users support for both tracking ongoing attacks and identifying potential security holes including:

- Portscan detection
  The classic (send a packet to every port) and slow (a kind of portscan where port scan happens very slowly in order to make its detection more difficult) portscan [17] can be easily detected because ntop reports the name of the last three hosts that sent a packet to each port less than 1024. We are currently adding support for detecting a new portscan method that makes use of packets with SYN/FIN flags for detecting ports that are not open and then, by difference, the list of open ports. In fact, when a host sends a packet containing the FIN flag to an open port no reply is sent, whereas if port has not been open, an ICMP [18] message is issued. Please note that portscan is detected not only for the host where ntop is running but for all the hosts for which ntop can capture packets (usually the whole subnet). This means that ntop provides (sub)network portascan detection whereas very few OSs sport portscan detection just for the local host.
- Spoofing detection
  Spoofing [19] [36] happens when a host claims to be another host for the purpose of intercepting packets. In general, for packets that do not originate on the subnet where ntop runs, it is not possible to detect spoofing. Instead, spoofing can be detected at least for hosts belonging to the same subnet of the host where ntop is running. This is because, hosts that spoof packets usually intercept ARP [29] requests directed to the victim and send ARP responses containing their physical address. The *arpWatch* plugin part of ntop, warns the user when two distinct IP addresses map to the same hardware address. Please note that spoofing detection should be used properly on networks where proxy ARP routers are installed or whenever a host has enabled multihoming support.
- Spy detection
  A spy [20] is an host whose network card is set in promiscuous mode for capturing packets independently whether they are directed to the host or not. Neped is a tool that sends to each host X of the local subnet an ARP request containing as target IP address the X IP address.

The peculiarity of neped is that the ARP packet is not broadcasted but sent to a dummy hardware address. If host X has its card set in promiscuous mode, it will process the ARP packet reply as if the packet was sent to it. ntop periodically runs neped and warns users about hosts whose cards are set in promiscuous mode. Unfortunately, the algorithm just described does not work for all operating systems hence neped is used to report information about hosts that certainly have their NIC set in promiscuous mode whereas nothing can be said about the remaining hosts.

- Trojan Horses detection
  Users do not usually detect the presence of trojan horses applications such as *BO2K* [21] until they experience severe problems. Because these kind of applications make use of well known ports (e.g. BO2K default port is 31337), ntop can detect their presence by periodically verifying whether there is some network traffic originated/designated to these ports. Even if those applications do not make use of the default port, ntop is able to count the traffic and the number of connections to any IP port. In general, trojan horses on a given host can be identified by first studying the host traffic patterns over the time, and periodically verifying if the actual traffic matches the pattern. In other words, if a host has never used the port X and suddenly send/receives a significant amount of traffic on the port X, we can assume that an unwanted application is using such port.
- Denial of service
  synflood [22] [23] is the ability of an host to send packets with the SYN flag set (the SYN flag is used for opening a TCP connection) on victim's open ports without further proceeding in the connection establishment. In this way the attacker fills all the victim's IP stack connection slots until the victim cannot accept new connections. Although some OSs are not affected by this problem because natively offer synflood protection, ntop can be used to detect attackers and report the problem to the network administrator. Please note that other kind of attacks including *smurf* [34] and network melt-down are also detected by analysing the traffic sent/received by each host.
- Network Discovery
  ntop comes with a couple of plugins that allow ARP and ICMP traffic to be monitored. We have noticed that comparing the number of ARP/ICMP Echo requests with the number of replies we can often identify hosts that run network discovery applications. Peak of unanswered requests in a given amount of time are usually the proof of existence of such applications running on the network. In particular, some applications (e.g. HP OpenView) discover hosts belonging to the same network using ARP, whereas ICMP is used for hosts outside of the network.
- Suspicious Packets
  Thesedays it is rather simple to find a packet generator using libraries freely available on the Internet. Using these tools, hackers exploit security flaws of the TCP/IP protocol suite [37] and weakness of some TCP/IP stack implementations [39] hence forge packets [32] for several purposes including, disconnection of active TCP sessions, OS guessing [14], and application/OS crash. In general it is difficult to identify when a packet has been forged. Nevertheless it is possible to identify some suspicious situations and report a warning to the network administrator. In particular, ntop is able to recognise:
  - Peak of packets having the RST (reset) flag set.
    The RST flag is used to close a connection when it is not possible to use the TCP three way handshake. A simple way of closing a TCP connection is by means of a packet containing the RST flag. A peak of packets with the RST flag could indicate that somebody is closing somebody else’s TCP connections. Using this technique, it is possible to achieve a denial of service if attackers close connections as they are open.
  - Packets with SYN/ACK flag that do not belong to an established connection.

Generally, these packets are used for probing a remote host (e.g. portscan) or for attacks such as IP spoofing via sequence guessing [24].

- Packets with SYN/FIN flag that do not belong to an established connection.
  The TCP specification does not clearly specify some state transitions [38] and hence allow some spurious state transitions [25]. Generally, these packets are used for denial of service attacks because on receiving a packet with SYN/FIN set, TCP makes a transition to CLOSE_WAIT state even if there was not an established connection.
- Overlapping offsets of fragmented packets.
  Some applications (neped for instance) are able to violate firewall/router access control lists by sending fragmented packets with overlapping/negative offsets. For instance, if the firewall allows external connections to the smtp port, an attacker could send the first packet fragment to the smtp port (hence the packet gets through) and a second packet with a negative offset that overrides the information on the first packet and that changes the destination port from smtp to another port violating firewall security. ntop checks that packet fragments do not overlap, and if so it warns the system administrator.

When a security violation or a network misconfiguration/problem is identified, ntop offers facilities for:

- reporting the problem to the network administrator;
- understanding where/how the attack originated by using the traffic information stored into the SQL database;
- performing specific actions (when applicable) in order to block the attack hence limit its extension to the whole network.

In case of a security violation, we would like to prevent the attackers from persisting with their bad behaviour until the administrator can fix the flaw. If the attacker does not belong to the local subnet, we could block packets sent/directed to it using the router ACL or using the appropriate firewall command if the attacker has to cross the firewall. If the attacker belongs to the local subnet, we could add a static entry into the host ARP cache of the host where ntop runs so that hosts that want to communicate with the attacker get confused because they receive multiple replies to ARP requests directed to the attacker.

4. Performance Issues

Users have tested ntop extensively on various network types running at different speeds. In general, ntop performance is greatly influenced by other running processes because some CPU-greedy applications may take up the whole CPU cycles for a few seconds causing packet loss. Supposing to run ntop on an average loaded host, tests shown that ntop can work with very low (if any) packet loss on a 100 Mbit ethernet.
Nevertheless, performance is strongly influenced by per-packet processing. In fact the more network flows are defined, the more processing time is required hence the higher is the probability of dropping some packets. In order to overcome the above mentioned problems, ntop implements internal timeouts and periodical garbage collection in order to purge old data and speculate about the state of active connections. For instance, if there is no data flowing on a connection for a very long period of time, then the connection might have been closed. In this case ntop assumes that the connection has been closed and then the connection entry is purged. This allows ntop to recover whenever some packets get lost and not to get stuck waiting for some lost packet to arrive.

5. Future Work

At the moment we are developing plugins for notifying network administrators either via email, SNMP [31] traps or GSM SMS (Short Messaging System) using some Perl scripts that make use of free SMS/WWW gateways available on the Internet. We are also running some experiments with

WAP (Wireless Application Protocol) appliances [26] using the Nokia WAP toolkit (it can be freely downloaded from http://www.forum.nokia.com). In fact, we believe that WAP-based phones can be a simple, yet effective way to both monitor relevant network activities from remote and notify network administrators when some network problems are identified. In particular WAP provides a push facility that fits very well in our context because it allows servers to push WML (Wireless Markup Language) pages to WAP devices when some conditions (e.g. security violation) happen. Unfortunately such a facility is missing in HTTP, hence administrators that use ntop have to keep their web browsers open in order to automatically pull a new HTML page from ntop.

Another work item is the development of a plugin able to identify potential security issues without human intervention. Although threshold-based solutions are widely adopted in the industry, we believe that in general it is quite difficult to express complex traffic patters using simple thresholds. A stated already, we are working towards a tool that studies a host traffic pattern and tries to identify potential security attacks by analysing how the actual traffic differs from the model. In the past we made some experiments embedding *SWI Prolog* [27] into a plugin. Then we decided to adopt a lighter solution as in some cases Prolog rules need far too much computing power and also because network administrators seem not to like much non imperative languages. Currently we are studying whether case-based reasoning [28] can fit our needs. In fact, we believe that it is much simpler to describe what are the conditions (case) that produce a certain event than to write an application coded using a procedural language.

6. Final Remarks

This paper attempted to show how ntop can be used not only for traffic measurement and monitoring, but also as an intrusion detection system. Although ntop sports some features present in tools such as NFR and NeTraMet, it has been designed as a web-based application able to present network traffic and security information in a simple fashion without the need to purchase expensive tools. Features such as embedded HTTP server, support of various network media types, lightweight cpu utilisation, portability across various platforms, storage of traffic information into an SQL database, extensibility via software components and integration with many network tools, make ntop suitable for all those people who want to analyse traffic and network security without having to afford expensive tools that often have a limited scope and lack many of the above features.

7. Availability

Both ntop and libpcap for Win32 are distributed under the GPL2 licence and can be downloaded free of charge from both the ntop home page (http://ntop.unipi.it/) and other mirrors on the Internet. Some Unix distributions including but not limited to FreeBSD and Linux, come with ntop preinstalled.

8. Acknowledgments

The authors would like to thank all the ntop users and early adopters who deeply influenced the design of the overall architecture with all their comments and suggestions. In addition, a special thank to Claudio Telmon <claudio@telmon.org> who has helped us better understanding network security.

9. References

[1] V. Jacobson C. Leres, and S. McCanne, *tcpdump*, Lawrence Berkeley National Labs, ftp://ftp.ee.lbl.gov/, 1989.

[2] N. Brownlee, *NeTraMet v.4.2 Users' Guide*, http://www.auckland.an.nz/net/Accounting/, 1998.

[3] M. Ranum, and others, *Implementing a Generalized Tool for Network Monitoring*, Proceedings. of LISA'97, USENIX 11th System Administration Conference, 1997.

[4] W. Cheswick, and S. Bellovin, *Firewalls and Internet Security: Repelling the Wiley Hacke*r, Addison-Wesley, 1994.

[5] M. Jander, *Web-based Management: Welcome to the Revolution,* Data Communications, 1996.

[6] M. Schultze, and others, *Homebrew Network Monitoring: a Prelude to Network Management*, Curtin University of Technology, 1993.

[7] W. LeFebvre, *top: a top-CPU Usage Display*, http://www.groupsys.com/topinfo/, 1993.

[8] E. Raymond, *The Cathedral and the Bazaar*, http://www.tuxedo.org/~esr/, 1999.

[9] L. Deri, *Surfin' Network Management Applications Across the Web,* Proceedings of 2nd Int. IEEE Workshop on System and Network Management, 1996.

[10] L. Deri, *Droplets: Breaking Monolithic Applications Apart*, IBM Research Report RZ 2799, 1995.

[11] S. McCanne, C. Leres, and V. Jacobson, *libpcap*, Lawrence Berkeley National Labs, ftp://ftp.ee.lbl.gov/, 1994.

[12] Microsoft Corporation, *NDIS Packet Driver 3.0,* 1996.

[13] Free Software Foundation, *GNU gdbm*, http://www.gnu.org/software/gdbm/, 1999.

[14] Fyodor, *Remote OS detection via TCP/IP stack fingerprinting,* http://www.insecure.org/nmap/nmap-fingerprinting-article.txt, 1998.

[15] E. Apostols, *Network Promiscuous Ethernet Detector (Neped),* http://apostols.org/projectz/neped/, 1998.

[16] S. McCanne, and V. Jacobson, *The BSD Packer Filter: A New Architecture for User-level Packet Capture*, Proceedings of 1993 Winter USENIX Conference, 1993.

[17] Fyodor, *The Art and Detection of Port Scanning*, Sys Admin Magazine, Nov. issue, 1998.

[18] J. Postel, *Internet Control Message Protocol (ICMP),* RFC 792, 1981.

[19] Computer Emergency Response Team, *TCP SYN Flooding and IP Spoofing Attacks*, CMU Report CA-96:21, 1996.

[20] B. Mukherjee, and others, *Network intrusion detection*, IEEE Network, 8(3), 1994.

[21] DidDog, *BO2K Tutorial,* http://www.bo2k.com/, L0pht Industries, 1998.

[22] C. Schuba, and others, *Analysis of a Denial of Service Attack on TCP*, COAST Laboratory, Purdue University, 1998.

[23] Phrack Magazine, *TCP/IP Security: TCP SYN Flooding,* 7(48), 1996.

[24] C. Chambers and others, *TCP/IP Security*, http://www.cis.ohio-state.edu/~dolske/gradwork/cis694q/, Dept. of Computer and Information Science, Ohio State University, 1997**.**

[25] B. Guha and M. Mukherjee, *Network security via reverse engineering of TCP code: vulnerability analysis and proposed solutions*, IEEE Network, 1(4), July/August 1997.

[26] WAP Forum, *WAP White Paper,* http://www.wapforum.com/what/whitepapers.htm, June 1999.

[27] J. Wielemaker, *SWI-Prolog 3.2.9 Reference Manual,* http://www.swi.psy.uva.nl/projects/SWI-Prolog/, University of Amsterdam, 1999.

[28] D. Leake, *Case-based Reasoning: Experiences, Lessons, and Future Directions*, AAAI Press/MIT Press, ISBN 0-262-62110-X, 1996.

[29] D. Plummer, *An Ethernet Address Resolution Protocol (ARP)*, RFC 826, 1982.

[30] S. Waldbusser, *Remote Monitoring management Information Base (RMON)*, RFC 1271, 1991.

[31] J. Case, and others, *Simple Network Management Protocol (SNMP)*, RFC 1157, 1990.

[32] S. Bellovin, *Packets Found on an Internet*, Computer Communications Review, 23(3), 1993.

[33] D. Comer, *Internetworking with TCP/IP*, Volume 1, 3rd Edition, ISBN 0-13-216978-8, Prentice Hall, 1995.

[34] C. Huegen, *The Latest in Denial of Service Attacks: Smurfing*, http://www.quadrunner.com/~chuegen/smurf.txt, December 1998.

[35] L. Deri, and S.Suin, *ntop: beyond Ping and Traceroute*, Proceedings of DSOM '99, Zurich, Switzerland, October 1999.

[36] P. Ferguson, and D. Senie, *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing, RFC 2267*, January 1998.

[37] S. Bellovin, *Security Problems in the TCP/IP Protocol Suite*, Computer Communications Review, 19(2), 1990.

[38] K. Ilgun, and others, *State Transition Analysis: a Rule-based Intrusion Detection System*, IEEE Transactions on Software Engineering, 21(3), March 1995.

[39] R. Morris, *A Weakness in the 4.2 BSD UNIX TCP/IP Software*, Technical report, AT&T Bell Labs, February 1985.

[40] B. Mukherjee, and others, *Network Intrusion Detection*, IEEE Network, May/June 1994.

10. Vitae

Luca Deri is currently sharing his time between Finsiel S.p.A. and the Centro Serra at the University of Pisa. He received his Ph.D. in Computer Science with a thesis on Software Components from the University of Berne in 1997. He previously worked as research scientist at the IBM Zurich Research Laboratory, and as research fellow at the University College of London. His professional interests include network management, software components and object-oriented technology. His web page address is http://www.tlcpi.finsiel.it/~deri/.

Stefano Suin got its degree in Computer Science from the University of Pisa in 1986. After a short experience running its own company, he is currently heading Serra, the networking centre of the University of Pisa. He co-designed the actual city backbone based on single mode optical fiber, wireless connections, ATM and Gigabit ethernet network transport. Additionally, he is member of several national research projects focusing on networking, and the creator and maintainer of the 'it.' Usenet hierarchy. His interests include network management, traffic measurement and network security.