AUTOMATED FIREWALL RULE SET GENERATION THROUGH PASSIVE TRAFFIC INSPECTION

Georg-Christian Pranschke[1], Barry Irwin[2] and Richard Barnett[3]

Security and Networks Research Group
Computer Science Department
Rhodes University
Grahamstown, South Africa

[1]g05p3292@campus.ru.ac.za, [2]b.irwin@ru.ac.za,
[3]barnettrj@acm.org

ABSTRACT

Introducing firewalls and other choke point controls in existing networks is often problematic, because in the majority of cases there is already production traffic in place that cannot be interrupted. This often necessitates the time consuming manual analysis of network traffic in order to ensure that when a new system is installed, there is no disruption to legitimate flows.

To improve upon this situation it is proposed that a system facilitating network traffic analysis and firewall rule set generation is developed. A high level overview of the implementation of the components of such a system is presented. The system makes use of a third party package, named *Firewall Builder* which provides firewall rule sets for a wide variety of firewalling solutions. Additions to the system are scoring metrics which may assist the administrator to optimise the rule sets for the most efficient matching of flows, based on traffic volume, frequency or packet count.

KEY WORDS

firewall, choke point control, automatic configuration, network traffic analyser, pcap, netflow

AUTOMATED FIREWALL RULE SET GENERATION THROUGH PASSIVE TRAFFIC INSPECTION

1 INTRODUCTION

In order for firewalls to serve their intended purpose, it is imperitive that they are correctly configured. This is because each individual network setup is different and if the firewall is to become an integral part of a network's infrastructure it has to cater for the individual properties of the network it is deployed in. A misconfigured firewall will, almost certainly, only provide the illusion of network security [15]. While configuring host based firewalls and firewalling solutions protecting small networks and correctly documented networks may be a relatively straight forward task for an experienced network administrator, it does become a very much harder task when dealing with poorly documented legacy and organically developed networks.

This research is focused on the feasability of automatically generating the configuration for a rule set generator called *Firewall Builder* [5], to further automate the process of configuring and deploying firewalling solutions.

The remainder of this paper is structured as follows. After a brief problem statement in section 2, in which we describe in what situations and setups the system is to be employed, we turn to a high level desgin overview of the proposed system in section 3. Each component of the components that make up the system is individually highlighted in sections 3.1, 3.2, 3.3 respectively. Section 3.4 deals with *Firewall Builder*, a third party product upon which the system relies. Section 4 describes possible future extensions to the system and section 5 concludes the paper.

2 PROBLEM STATEMENT

A firewall is rarely a single piece of hardware or software [12] and, therefore, combining the various technologies involved into a well configured firewalling solution is often a non trivial task in itself. The process of configuring and deploying a firewalling solution is further complicated when a firewall is to be introduced into a network segment that previously did not have any choke

point controls. This is often the case in growing network infrastructures where it might be desirable to introduce choke point controls at a node that previously had no such system. Because the flows passing through the node might not be known to their full extent and because an unintended interruption of production traffic is surely undesirable, it is often necessary to inspect the traffic flows at the node manually and then to derive firewall policies for the particular firewalling solution proposed at the node. While there are many traffic analysers available, ranging from propriety commercial products, that cost up to 25,000 USD, to free and open source solutions [9]; that perform traffic flow analysis, there is no product that specifically caters for the configuration of firewalls.

3 PROPOSED SOLUTION

To alleviate this situation the authors are currently developing a system, which specifically aims to automate as much of the firewall configuration process as possible. This shall be achieved by firstly building upon automation solutions already available, specifically *Firewall Builder* [1], and secondly by taking a flow based traffic analysis approach utilising portions of Cisco's *NetFlow* format [9]. The system shall consist of two distinct modules, one for analysing the traffic at the proposed choke point and one for generating a rule set to match the observed traffic. The traffic analyser should be capable of analysing either live traffic at the node or trace files recorded at the node, in *pcap [4]* or *NetFlow* format.

The output of the traffic analysis shall be in a format similar to that of *NetFlow*, as this has a high information density, which is desirable for all consecutive steps in the configuration process. The resulting flows shall be stored in a database upon which the GUI-based rule generator can act. The rule generator shall then, in turn, propose a set of rules based on the observed flows and allow the administrator to review and refine these rules from within a GUI. The refined rules shall then be exported to a format understood by *Firewall Builder*, which is capable of deploying these rules to a wide variety of firewalling solutions. Thus *Firewall Builder* shall act as the backend of the configuration process. A graphical representation of how the components interact is given in figure 1. Because it is unusual for a dedicated firewall machine to have windowing capabilities, this modular approach enables the traffic analyser to run on these machines on the command line, whereas the rule generator can be run on a remote desktop machine.

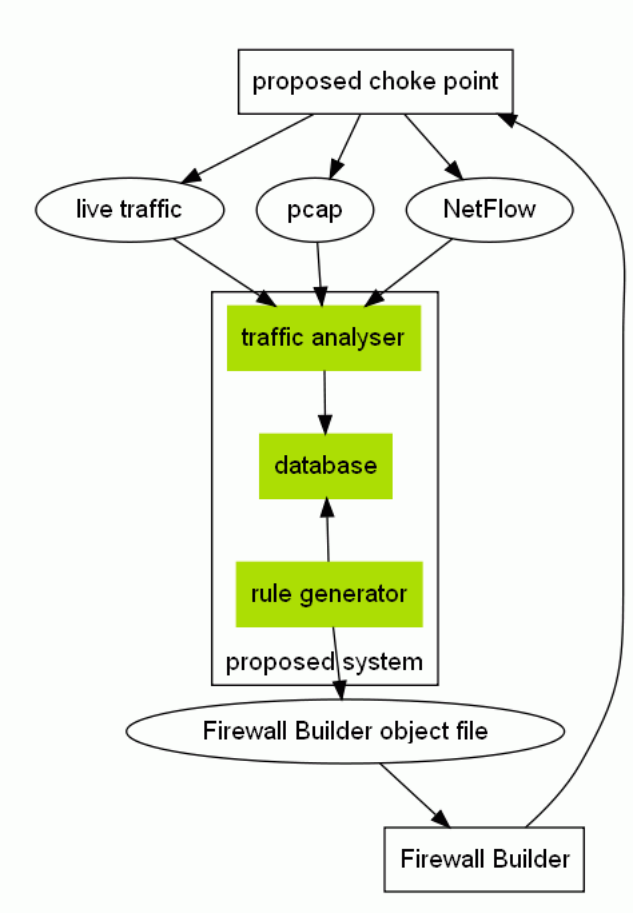

Figure 1: Overview of the high level design of the proposed system and its position in the configuration process.

It is expected that this solution will speed up the process of configuring and deploying firewalls considerably because the administrator does not need to concern himself with a tedious manual traffic analysis, or the intrinsic details of writing firewall policies for a particular firewalling solution.

As most network infrastructures are inherently heterogeneous, the project puts a strong emphasis on cross platform portability and the use of free and open source tools and libraries.

3.1 Traffic Analyser

The traffic analyser's primary task is to create or extract traffic flows from its input data and to consequently store these flows in a database. Working with

flows is advantageous because of their high information density, and because they contain stateful information about the prevalent network connections. The three supported types of input data are *NetFlow* flow records, live traffic and *pcap* dump files.

3.1.1 NetFlow

NetFlow is both a format and a technology. Initially developed in-house at Cisco, it has quickly become the *de facto* standard for network analysis and is used for a variety of purposes including, but not limited to, billing, network planning, traffic engineering and the detection and analysis of security incidents[8, 9]. *NetFlow* enabled devices can export flow data via a UDP based protocol to a *NetFlow collector*, which then files, filters and stores the flow data. Ideally the traffic analyser should be capable of processing both the UDP exported flows and *NetFlow collector* files.

Flows are created by continuously analysing IP packets and categorizing them into IP flows. A packet is either categorized into an existing flow or creates a new flow. Finished or expired flows are then exported to the *NetFlow collector* via UDP. A flow is defined by seven key fields, namely, source IP, destination IP, source port, destination port, protocol, type of service and input interface. Any two packets sharing the same entries for all seven fields belong to the same flow [7]. There are several versions of *NetFlow*, some of which are more commonly used than others, namely versions 1, 5, 7, 8 and 9 that incrementally improve upon another and provide a richer feature set with more detailed flow records [8].

The traffic analyser only requires a subset of the information provided by *NetFlow* and shall extract the relevant bits for its operation into a custom flow format and discard the rest.

3.1.2 Live Traffic and *pcap* Trace Files

The traffic analyser uses *libpcap* [4, 11](*WinPcap [6]* on Windows) to handle both, live traffic and *pcap* dump files. The processing of these two types of input is nearly identical. The strategy to obtain the same custom flows that are extracted from *NetFlow*, is to screen the packet data for three way handshakes and TCP FIN and RST packets.

The ACK packets involved in the three way handshake can be determined through the packets' sequence numbers [14]. This establishes the sources and destinations and hence the direction of the traffic flows. The difference in the timestamps between these packets allows for an estimation of the duration of any given connection. The packets that are neither SYN nor FIN are matched to one of the existing flows and their payloads added to the total volume of traffic in the flow. Their occurrence is also recorded in the packet count of the flow. Because packets might arrive out of order, care must be taken when reconstructing the flows to not disregard valuable information, meaning that non SYN or FIN packets without a corresponding flow do create their own flow so that it is possible to reconstruct them later or at least take them into consideration when generating the rules at a later stage.

The flow information is then stored in a database for later analysis by the rule generator. The database table that records the traffic flows should feature fields for a flow identifier number, the flows type of service, the timestamp of the SYN - ACK packet, the timestamp of the FIN - ACK packet, the total packet count in the flow, the total volume of traffic transferred in the flow so far, the flows source address and port and the flows destination address and port.

3.2 Database

Currently the project uses the embedded SQL database engine *SQLite* [3, 13] to record the flows, which has various advantages over other more sophisticated database solutions. From a performance perspective, this in-process library is simply much faster than any networked database solutions could ever be. Because *SQLite* is serverless, self-contained and requires no configuration it also increases the ease of use of the system. *SQLite* stores its databases in a file, in a format that is consistent across all platforms, thus its database files lend themselves as the perfect format for information exchange between the traffic analyser and the rule generator, especially across different platforms [13, 3]. An added advantage is that the database files can be efficiently compressed and are therefore ideal to be sent across the network.

3.3 Rule Generator

The rule generator then uses the database file to propose matching firewall rule sets. The user interface of the rule generator shall provide an integrated terminal so that the user does not have to leave the GUI to start and control

the traffic analyser on the remote site. A facility to perform custom SQL queries against the database shall also be provided. The flows recorded in the database shall be visualized in a table like structure for close inspection by the user. The security policies are visible in another tab and all changes made from anywhere within the system should be immediately reflected here. At the end of the review and refinement process the user should be able to export the policies into a *Firewall Builder* network object file or alternatively invoke one of *Firewall Builder's* policy compilers directly.

The basic strategy to automatically generate a rule set is to divide the network into an 'inside the wall' and an 'outside the wall' part. Initially both sides start off with the least possible privileges (deny all). Then all incoming flows targeted at commonly known services are permitted. Flows targeting high port numbers are only allowed as a response to outgoing flows. The presence of services that are commonly considered outdated such as Telnet is pointed out to the user and a suggestion for their replacement made. This quite lax basic configuration can then be refined by the administrator by either individually allowing or denying flows or by specifying wildcards on IP, protocol or port level.

By default ICMP rules will be generated from a template. The user can then activate or deactivate the desired subtypes. A facility for reverse DNS lookup shall be provided, so that unwanted sites can be blocked at the discretion of the user.

3.4 Firewall Builder

Firewall Builder is a GUI-based firewall configuration and management tool that supports *iptables* (netfilter), *ipfilter*, *pf*, *ipfw*, *Cisco PIX* (FWSM, ASA) and *Cisco routers extended access lists* [1]. It features a set of policy compilers that compile the rule sets created from within its GUI, from xml based object files, into, firewalling solution specific, firewall rule sets. The policy compilers do also create automatic deployment scripts, that allow the firewall to be brought up remotely and to roll back the installation if necessary. *Firewall Builder* also ensures that the SSH connection between the configuring host and the firewall will never accidentally be interrupted. Because Firewall Builder's GUI is built upon *Qt* [2, 10] it is capable of running on a wide variety of target platforms, such as Linux, FreeBSD, OpenBSD, Mac OS X and Windows [5]. All of the above mentioned features make *Firewall Builder* the ideal backend for the project.

4 POSSIBLE FUTURE EXTENSIONS

Since the proposed system is considered a proof of concept, most future extensions considered at this time are related to adding features that will make it a stable production release. A very obvious one is adding IPv6 support, which should be relatively straight forward. Furthermore the traffic analyser could be extended to generate additional scoring metrics that can help to further optimize the generated rule sets. The inclusion of an intrusion detection and prevention system such as *snort* and in turn its automatic configuration and deployment would certainly result in a more powerful and complete firewalling solution. At a later stage, the option of customizing and integrating *Firewall Builder's* policy compilers into the rule generator might prove desirable, to increase ease of use and lessen the dependency on this third party package.

5 CONCLUSION

Although this research is still at a very early stage, it is anticipated that the approach of automatic firewall rule set generation by means of passive traffic inspection will prove feasible and that a working prototype can be developed within the given timeframe. It is hoped that the proposed system will not only be quicker and more convenient than manual configuration, but possibly prove to be more accurate and allow for faster turnaround in the deployment of new firewalling solutions. This should result in decreased risk and cost for organisations deploying such solutions.

ACKNOWLEDGEMENT

The authors would like to acknowledge the support by Telkom SA, Comverse, Tellabs, Stortech, Mars Technologies, Amatole Telecommunication Services, Bright Ideas Project 39, THRIP and the NRF through the Telkom Centre of Excellence in the Department of Computer Science at Rhodes University.

References

[1] Firewall builder cookbook. Online: http://www.fwbuilder.org/guides/.

[2] Qt - a cross-platform application and ui framework. Online: http://www.qtsoftware.com.

[3] Sqlite. Online: http://www.sqlite.org.

[4] Tcpdump/libpcap public repository. Online: http://www.tcpdump.org.

[5] What is firewall builder. Online: http://fwbuilder.org/about.html.

[6] Winpcap: The windows packet capture library. Online: http://winpcap.org.

[7] Cisco ios ipsec accounting with cisco ios netflow. Tech. rep., Cisco Systems, 2004.

[8] Cisco cns netflow collection engine user guide, 5.0.3. Tech. rep., Cisco Systems, 2005.

[9] Introduction to cisco ios netflow - a technical overview. Tech. rep., Cisco Systems, 2007.

[10] Blanchette, J., and Summerfield, M. *C++ GUI Programming with Qt 4*. Prentice Hall, 2006.

[11] Garcia, L. M. Programming with libpcap - sniffing the network from our own application. *hackin9 3* (2008), 39.

[12] Ogletree, T. *practical firewalls*. Que, 2000.

[13] Owens, M. *The Definitive Guide to SQLite*. Apress, 2006.

[14] Siyan, K. S., and Parker, T. *TCP Unleashed*. SAMS Publishing, 2002.

[15] Zwicky, E. D., Cooper, S., and Chapman, D. B. *Building Internet Firewalls*. O'Reilly, 2000.