Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium, Volume 21, No. 1, ISSN 1726-9679
ISBN 978-3-901509-73-5, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2010
Make Harmony Between Technology and Nature, and Your Mind will Fly Free as a Bird
DAAAM Symposium

SECURITY MEASURES IN VIRTUAL LABORATORY OF MICROPROCESSOR TECHNOLOGY

DULIK, T[omas] & BLIZNAK, M[ichal]

Abstract: *This paper presents the security framework we have developed for virtual laboratory of microprocessor technology deployed at TU Brno. The central part of the framework is network probe implemented with ulogd2 - an open source connection logging daemon, which was extended by several new plugins allowing analysing the network traffic with negligible CPU load. This solution provides performance similar to expensive dedicated network probes but is much more flexible as it can be implmented directly on the existing Linux servers or Linux routers without cannibalising their throughput.*
Key words: *network security analyzer Linux ulogd2*

1. INTRODUCTION

In the "Virtual laboratory for microprocessor technologies" (VLAM) project, we have built a laboratory virtualization infrastructure which is completely based on Linux. Therefore, when implementing the network security measures for VLAM project, we were looking for free Linux-based network probes first, leaving aside all the expensive standalone network probes as a fallback solution.

The network traffic probe is a basic building block of any network security tool. The probe captures packets from the network, pre-processes and/or analyzes them and sends the results to a collector service. The probe must be able to handle traffic even during 100% network load and this is quite hard to achieve with 1Gbit or 10Gbit Ethernet, because the packet analysis is much more CPU intensive than routing/firewalling tasks carried by ordinary network devices.

Therefore, although some network probes are implemented in most brand-name IP routers, the burden on their CPU often calls for deploying a standalone network probe devices with lots of CPU power and/or HW-accelerated packet analysis.

In VLAM project, we needed a system which would be capable of analyzing and protecting against security threats but we also wanted to have the accounting of both legitimate and unwanted network traffic. For these purposes, there are several solutions available in Linux, for example libpcap with PF_RING, nProbe, fprobe, softflowd, pmacct, libnetfilter_log, libnetfilter_conntrack, libe1000, ipt_netflow and others. The nProbe, fprobe, softflowd and ipt_netflow were analyzed and benchmarked by (Kuna, 2009) who found the ipt_netflow kernel module as the best performing tool out of those tested. The nProbe and fprobe were lacking far behind ipt_netflow even when the PF_RING sockets were used because these tools do the packet analysis in userspace which requires time-consuming context switches.

Based on our own analysis of the above mentioned tools, we decided to build our system on the libnetfilter_conntrack and ulogd2 tools. Unfortunately, to our knowledge, an analysis or performance benchmark of these tools has not been published yet. The only paper describing ulogd2 (Welte, 2005) was written by the author of ulogd2 but its content is now quite outdated. Due to the obligatory space limitation we are not able to publish the analysis and benchmark of ulogd2 with libnetfilter_conntrack in this paper. However, we have made these benchmarks and found that libnetfilter_conntrack induces the same or smaller CPU load as ipt_netflow. Moreover, we argue that libnetfilter_conntrack is much easier to deploy since it has already been included in all Linux distributions. Despite of this, it remains widely ignored by Linux users.

This paper is structured in following way: first, we introduce the VLAM project architecture. Then we describe the security framework and its components. Finally, we compare our approach with the available alternatives and show the possible future development paths in this area.

2. VLAM PROJECT ARCHITECTURE

VLAM security framework is implied by the VLAM network architecture depicted on Fig. 1.

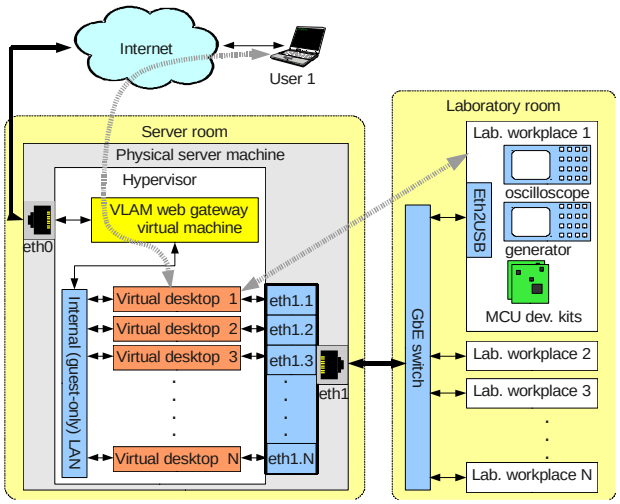

Fig. 1. VLAM project architecture

The project allows students to work remotely with the equipment located at the microprocessor laboratory similar way as they work in normal full-presence class setting, where each student sits at one laboratory workplace equipped by MCU development kit, an oscilloscope and a signal generator and has full control over this equipment. In the remote work scenario, the VLAM project provides access to the laboratory workplaces by means of virtual desktop machines which are connected to the laboratory equipment through ethernet or ethernet-to-USB bridges.

The VLAM web gateway system (Bliznak, 2010) assigns one virtual desktop machine to each remote student so that remote students are isolated from each other.

The other security element we have used are the 802.1Q VLAN interfaces (eth1.1- eth1.N on Fig. 1) separating the networks of each laboratory workplace. The separation is done by the GbE switch whose configuration is shown on Fig. 2.: the switch port connected to the physical server machine is configured as 802.1Q VLAN trunk and the ports connected to

the individual workplace networks are configured to pass only single VLAN. The packets coming from the workplaces are 802.1Q-tagged after entering their switch ports and packets coming from server are untagged before leaving the switch. This configuration creates the same level of isolation as if each workplace was connected to a dedicated NIC in the server by a dedicated cable – such a connection would be possible but highly impractical.

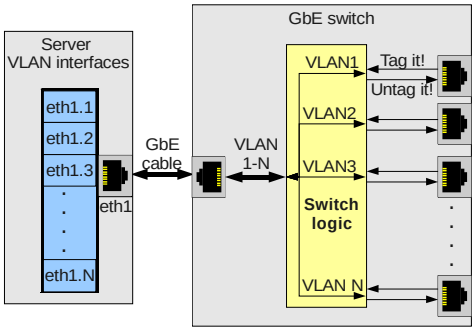

Fig. 2. 802.1Q VLANs in VLAM project

In fact, the VLAM project architecture was designed with respect to existing network infrastructure and space/power/air-conditioning constrains implied by the VLAM server running in 24/7/365 mode. We are aware that the server should be placed in server room which can by far away from the laboratory and that it is not feasible to install new cables between these places. Therefore VLAM needs only one ethernet cable between the server room and we implement the laboratory inner/inter-network isolation by 802.1Q VLANs.

3. VLAM SECURITY FRAMEWORK

The two thick grey dotted arrows on Fig. 1. are showing the user interaction with VLAM components. The only components visible to the user are the "VLAM web gateway" and the laboratory workplace that the web gateway assigns to the user after successful login. The implementation of the VLAM web gateway and its security measures are described in (Bliznak, 2010). In this paper, we only address the top-level security framework we have developed for VLAM.

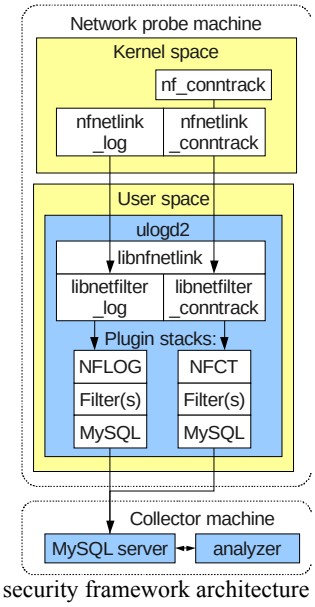

Fig. 3. VLAM security framework architecture

The framework structure is shown on Figure 3. It consists of the network probe deployed at the physical virtualization server and the data collector/analyzer component, which can be installed on any physical or virtual machine even outside of the VLAM project infrastructure.

The network probe is implemented by the ulogd2 daemon. The configuration of the daemon lies in defining arbitrary number of "Plugin stacks", each stack consisting of one input plugin, none, one or more filter plugins and one output plugin.

In VLAM, we currently use two stacks: "security violations" stack with the NFLOG input plugin and "traffic/connection accounting" stack with the NFCT input plugin.

The NFLOG input plugin collects messages from the iptables/netfilter firewall rules which are crafted to detect attacks against the VLAM infrastructure – e.g. DoS/DDoS, port scans, vulnerability exploits etc. We currently do not use any ulogd2 filter plugins in the NFLOG stack; we rather rely on processing implemented directly in the firewall/iptables rules.

The NFCT input plugin collects traffic accounting data. We have implemented a byte-counter filter which limits the amount of logged items to only successful connections – that is those with non-zero amount of data transferred in the reply. Thanks to the byte-counter filter the overall CPU load induced by the ulogd2 is very small even in the network 100% load. As for the traffic between the ulogd2 and mysql daemon coming from the NFCT stack, it is also quite negligible.

At the collector machine, long-term security/traffic analyses are possible thanks to the network traffic events history stored in the database. For example, it is possible to include information about attacking IP address history in every report issued by the analyser component.

4. CONCLUSION

Although largely unnoticed by the Linux community, the ulogd2 provides a very effective way for strengthening security of servers and networks. This paper shows only one of the possible deployment scenarios without performance benchmarks. We are sure that the ulogd2 performance deserves detailed analysis and we are already working on it to be able to publish the results very soon.

5. ACKNOWLEDGEMENTS

This work has been supported by Ministry of Education, Youth and, Sports of the Czech Republic grant MŠMT 2C06008 "VLAM: Virtual Laboratory of Microprocessor Technology Application".

We also thank the whole Linux community and especially the iptables team for giving everyone the possibility to use, to learn and to contribute to such a great system.

6. REFERENCES

Bliznak, M. (2010). Virtual Laboratory of Microprocessor Technology Application, *Proceedings of the 21st DAAAM International World Symposium,* DAAAM International Vienna, Editor B. Katalinic, Zadar, Croatia, October 20-23, 2010

Kuna, L. (2009). Možnosti analýzy IP toků v OS Linux, TU of Ostrava, http://www.cs.vsb.cz/grygarek/TPS/ - final project report for "Advanced Computer Network Technologies" course, *Accessed on:* 2010-6-27

Neira, P. N. (2006). Netfilter's connection tracking system, *;login: The USENIX Magazine*, June 2006, Volume 31, Number 3, page numbers 34-39, http://people.netfilter.org/pablo/docs/login.pdf, Accessed on: 2010-6-27

Welte, H. (2005). Flow-based network accounting with Linux, pp. 265-270, *Proceedings of the Linux Symposium*, Ottawa, Ontario, Canada, July 20nd–23th, 2005