ZINC

not another conference

2020 Zooming Innovation in Consumer Technologies Conference (ZINC)

Online, 26-27 May 2020

ISBN: 978-1-7281-8259-9

IEEE Catalog Number: CFP20ZIN-ART

www.GoZinc.org

The Concept of Consumer IP Address Preservation Behind the Load Balancer

Vladimir Ciric
University of Nis, Faculty of Electronic Engineering
Nis, Serbia
vladimir.ciric@elfak.ni.ac.rs

Nikola Vidojkovic
University of Nis, Faculty of Electronic Engineering
Nis, Serbia
nikola@vidojkovic.in.rs

Nadja Gavrilovic
University of Nis, Faculty of Electronic Engineering
Nis, Serbia
nadja.gavrilovic@elfak.ni.ac.rs

Ivan Milentijevic
University of Nis, Faculty of Electronic Engineering
Nis, Serbia
ivan.milentijevic@elfak.ni.ac.rs

***Abstract*— With the trend of moving all kind of services online, the number of globally connected services and, generally, devices had enormous growth in the last decade. In such an environment, a security becomes a big challenge. There are many network security architectures. However, most of them make a target service to suffer and struggle to maintain the functionalities behind all additional security layers. The first functionalities that are often lost are functionalities based on knowing the original consumer IP address (authentication, geo-location of consumers, etc). The preservation of the original IP addresses is a difficult task and requires advanced routing techniques at the service provider premises. In this paper the concept of advanced routing and network address translation which preserves the original consumer IP address is discussed. The concept involves load balancer, network intrusion detection, and proxy. The concept and the routing techniques will be described in detail. The results of the system implementation and evaluation will be given.**

Keywords*— *network security architecture, proxy service, routing techniques, intrusion detection systems

I. INTRODUCTION

The internet has grown rapidly in the last decade. Due to the increasing availability, affordability, and proficiency of network enabled devices, the number of globally connected hosts is steadily increasing [1]. With the large number of devices, services, and users the security becomes a challenge. The Internet of Things (IoT), which tends to connect all kinds of devices and sensors, often not well-tested because of time-to-market reasons, make the security even worse [2].

There are a lot of network security architectures and approaches which tend to lower the security risks and introduce additional network control and monitoring functions [3],[4]. Each security architecture provides specific access control when crossing network security perimeter, which is often deployed at the network boundary. Current designs for publically available services usually rely on packet filtering, proxy technology, and intrusion detection [5],[6]. With introduction of load balancers into the architecture, situation becomes even more complicated [7]. The first feature that is often lost is a transparency.

Transparency, in a broad sense, is a feature that makes the end devices in the communication, i.e. client and server, unaware of the existence of any additional layer [8]. In a direct communication, the destination end device (the server) receives the Layer-3 packet with the original IP address of the sender (the client). With robust network security perimeter, the transparency feature becomes more difficult to retain. Proxy servers and load balancers often act as intermediate devices, replacing the source IP address with their own while intercepting the packet for inspection. For the consumer they become "end device" that provide the service [8]. In such a scenario, the destination server loses the information about the original source of the packet, which can lead to lose of some service functionalities.

This problem is often addressed by the implementation of a Layer-2 bridge with extensions for Layer-3 security services [9]. Solutions that address the problem at Layer-3 give more flexibility, but they require source IP address spoofing at the destination. With source IP address spoofing at the security premises, routing becomes a challenge.

In this paper we present the concept of advanced routing and network address translation which preserves the consumer IP address, while the consumer is accessing the service through load balancer, proxy and the network intrusion detection system. The architecture and the routing techniques will be described in detail. In order to implement the routing that will support the communication we will utilize the application containerization. All used components will be described in detail. The results of system implementation and evaluation will be given.

This paper is organized as follows. In Section 2 we give a brief overview of proxy techniques, proxy protocol and routing techniques for IP address spoofing. Section 3 is the main section where we present and discuss the security architecture. In Section 4 the implementation results will be presented, while in Section 5 we give concluding remarks.

II. PROXY SERVICES AND LOAD BALANCING

A. Proxy services

In computer networking, a proxy is an entity that acts as an intermediary for requests from clients seeking resources from servers. Regarding their relative distance from the client there are forward and reverse proxy servers. The forward proxy is close to the client, while the reverse is close to the server. Regarding the awareness of the client of proxy existence, there are classical and transparent proxy servers [8]. Classical proxy server usually has two ends: the server and the client end. The server end acts as a server exposing same or modified communication protocol as a destination server. The client is aware of the existence of the proxy, and it connects to the proxy directly (Fig. 1a, message 1). The main problem with classical proxy is how the client delivers the information about the final destination to the proxy server. There are several approaches, but the simplest one (usually used for reverse proxies) is to configure the proxy in advance to be dedicated to one server only (Fig. 1a, message 2). The proxy connects to the final destination and fetches the data for the client, passing it back to the client (Fig. 1a).

978-1-7281-8259-9/20/$31.00 ©2020 IEEE

In the transparent scenario the client tries to contact the final destination directly, unaware of the existence of the proxy (Fig. 1b, message 1). The proxy usually has two interfaces and acts as a regular router on the way between the client and the server. However, when the regular request is passing through the proxy on its way, the proxy intercepts it based on the destination address and port, and neither forwards it nor drops it (Fig 1b). The proxy fetches the packet, and checks the source permissions for accessing the final destination. Then, the proxy contacts the destination server on behalf of the client, like in the classical proxy scenario [8]. Connections that go through a transparent proxy are actually made of two complete sessions: a session between the client and the IP address of the destination server (this session is "intercepted" by the proxy), and a session between the proxy and the server (Fig. 1b).

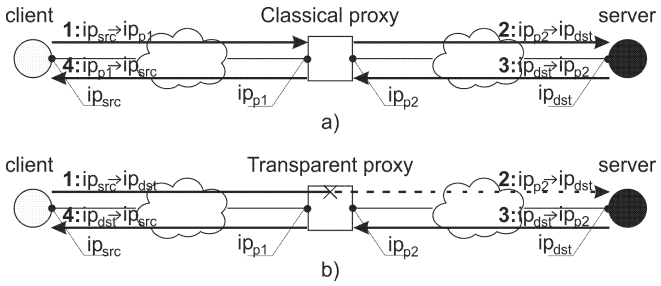

Fig. 1. The proxy servers operation: a) classical proxy, b) transparent proxy

Like classical proxies, transparent proxies hide the client IP address from the servers, since from the server perspective the session is initiated by the proxy [8].

B. Load balancing

Load balancing refers to the process of distributing a set of tasks over a set of resources, with the aim of making their overall processing more efficient. There are several approaches to load balancing [10]. Typical Layer-4 load balancer that acts as a service broker is shown in Fig. 2.

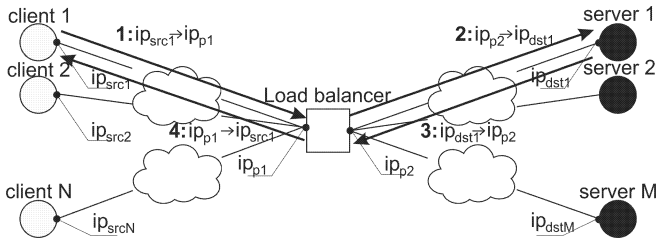

Fig. 2. Communication through load balancer

When the Layer 4 load balancer receives a request and makes load balancing decision, it changes the destination IP address from its own (Fig. 2, message 1) to that of the content server it has chosen (Fig. 2, message 2). Similarly, before forwarding server responses to clients, the load balancer changes the source address recorded in the packet header from the server's IP address to its own [10]. Load balancers can use different methods to choose the content server that will serve the request. These include the round-robin, least connections, etc.

From both the client's and the server's angle, the load balancer acts as a classical proxy, with predefined destination address (addresses). Eventually, it hides the source IP address from the server.

C. The Proxy protocol and IP address spoofing

The problem of source IP address preservation at the destination server can be addressed in several different ways. HTTP protocol defines the "Forwarded" extension, which replaces the 'X-Forwarded-For" header that carries information about the original source address. However, this technique requires knowledge of the underlying protocol to be implemented in intermediaries, and has specific implementation for each protocol. SMTP, for example, has XCLIENT protocol extension, etc [12].

In order to deal with the problem that each application layer protocol need to have its own extension, HAProxy published the Proxy Protocol (PP) [12]. The PP defines custom text or binary headers to carry the information about the source in the standard way through the cascade of proxy servers or to the final destination server. The idea behind the PP is simple: a packet carrying PP header precedes the regular (HTTP, SMTP, etc.) protocol communication (Fig. 3), and it is sent to the same port, to the same service as the main protocol packets. If the server is "PP ready", it will collect the information and fill the missing server structures: INET protocol, original source and destination IP addresses and ports. If the software is not "PP ready", the well-known protocols like HTTP, SMTP, FTP, etc., will cause a sort of "bad request" message, but the protocol itself will continue without the stall, with the drawback that the destination will not have the information about the source [12].

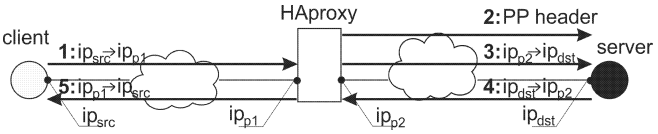

Fig. 3. HAproxy protocol communication sequence

In order to preserve the original source IP address at the destination server, the IP address in the message 3 from Fig. 3 should be spoofed. This is shown in Fig. 4. The spoofer component does that by accepting the proxy request with the PP header, and then spoofs the source information from the regular IP packet header before sending it to the destination server (Fig. 4).

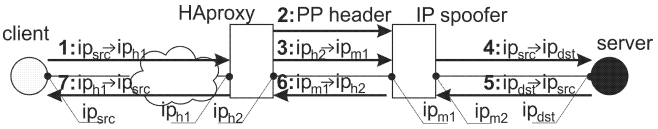

Fig. 4. HAproxy protocol communication sequence

From the server's perspective, the communication shown in Fig. 4 is exactly the same as in the case of a direct communication. However, this scenario requires the implementation of an advanced routing, because the reply from the server contains the destination address of the original source (Fig. 4, message 5), but it is actually destined for IP spoofer.

III. The Design of Network Security Architecture

In order to implement the routing that will support the communication shown in Fig. 4, we utilized the application containerization. The containers are primarily used to encapsulate the IP spoofer close to the server, so the return packet from the server (Fig. 4, message 5) can be easily intercepted within the container. The problem with the interception of the mentioned packet lies in the fact that the

packet destination address in message 5 from Fig. 4 is useless, because "the true" destination is the IP spoofer within the container.

The concept of consumer IP address preservation behind the load balancer is shown in Fig. 5. There are two application containers. The first container (LXC container 1) contains the load balancer and the network intrusion detection system. It has two network interfaces $ifc_{1\text{-}1}$ and $ifc_{1\text{-}2}$ (Fig. 5). The second container (LXC container 2) contains IP spoofer and the server, as well as *iptables* Linux module for advanced routing. The second container has only one entry point, i.e. ifc_2.

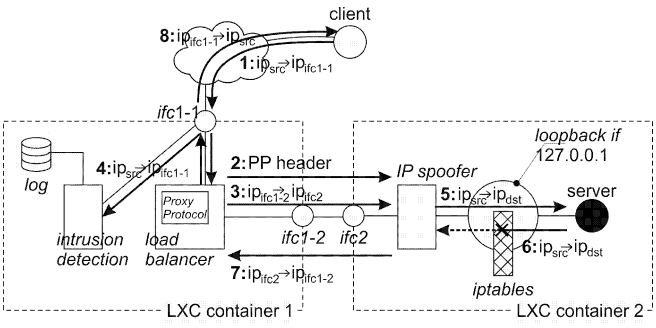

Fig. 5. The security perimiter architecture

From the client's perspective, the service is seen through the IP address of the load balancer, i.e. the address of the interface $ifc_{1\text{-}1}$ (Fig. 5, message 1). The load balancer should have the Proxy protocol support enabled. It should listen on the interface $ifc_{1\text{-}1}$, and send requests through interface $ifc_{1\text{-}2}$ (this depends on the IP address scheme). From the load balancer's perspective the container 2 abstracts the server, thus it forwards the request to the IP address defined by the ifc_2 in the same manner as it is shown in Fig. 4 (Fig. 5, messages 2 and 3). In addition, container 1 consists of the network intrusion detection which listens for the packets on the interface $ifc_{1\text{-}1}$(Fig. 5, message 4) and logs the intrusion attempts.

The second container implements advanced routing using Linux *iptables.* At the entry point of container 2, the spoofer accepts the proxy protocol header and the client request (Fig. 5). It prepares the message 5 from Fig. 5, but additionally it marks the packet, in order to be recognizable by the other subsystems.

The *iptables* from Fig. 5 enables advanced routing based on the packet marks. The code snippet from Fig. 6 shows the setup of the *iptables* that enforces routing table 100 for the marked packets, which routes these packets to loopback interface instead of routing them to the publically available interface. This applies to both IPv4 and IPv6.

```
iptables -t mangle -I PREROUTING -m mark --mark 123 -j
CONNMARK --save-mark

ip6tables -t mangle -I PREROUTING -m mark --mark 123 -j
CONNMARK --save-mark

iptables -t mangle -I OUTPUT -m mark --mark 123 -j
CONNMARK --restore-mark

ip6tables -t mangle -I OUTPUT -m mark --mark 123 -j
CONNMARK --restore-mark

ip rule all fwmark 123 lookup 100

ip route add local 0.0.0.0/0 dev lo table 100

ip -6 rule all fwmark 123 lookup 100

ip -6 route add local 0.0.0.0/0 dev lo table 100
```

Fig. 6. The code snippet that enables the adv. routing in the container 2

Based on the routing rules involved in Fig. 6, the packets marked with "123" will be routed to the loopback interface (Fig. 5). Thus, after the preparation of the message 5 from Fig. 5, the spoofer sends it to the loopback interface with the source IP address equal to the address of the client, and the destination IP address of the server. The server is setup to listen for the packets arriving over the loopback interface, thus it receives the packet. From the servers perspective it receives the real IP address of the source where it should be – in the header of the IP packet, and replays as in the case of the direct communication (Fig. 5, message 6). The *iptables* again reroutes the packet to the loopback interface, where it is recognized by the mark "123" and picked up by the spoofer. It replies to the load balancer, which does the network translation again and forwards the response to the client.

The containers 1 and 2 from Fig. 5 can be implemented either on single or separate nodes. In order to fully utilize the load balancer, the container 2 can be replicated many times.

IV. Implementation and Evaluation Results

For the sake of illustration and basic performance evaluation, the both containers from Fig. 5 are implemented on the server with i5-2410 3.1 GHz CPU, 6 GB 1333 MHz RAM, and SSD hard disk. Debian 9 and Proxmox virtualization environment were used to implement LXC containers, while on the container level we used CentOS 7.6.

For the load balancer Linux *nginx* tool v.1.14.2 is chosen. The well-known Snort IDS v2.9.12 is implemented as a network intrusion detection component, with the setup for malicious attempts logging. The server from Fig. 5 is Apache web server v.2.4.6. For the evaluation, the system is made publicly available and the online WebPageTest tool was used for measuring the response times (Fig. 7).

In order to be able to compare the results and estimate the overhead that the architecture from Fig. 5 introduces, we implemented the web server as a stand-alone server using the same versions of the components, as well. This communication is similar to scenario shown in Fig. 1a, but instead of a real proxy, we had static NAT address translation.

In both cases we used three different page sizes with different number of objects. We varied the page size from 2 to 1.000kB. Table 1 shows the results. For each measured parameter, two columns are given in Table 1 and denoted as

A and B. The columns denoted with A stand for the architecture shown in Fig. 5, while the columns denoted with B stand for the direct client server communication.

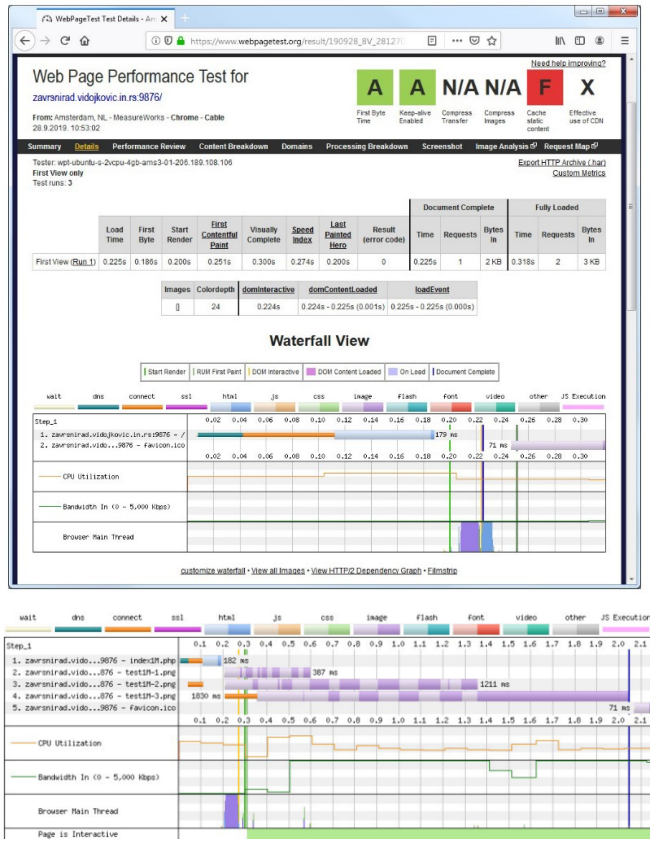

Fig. 7. WebPageTest evaluation of the system response times

TABLE I. SYSTEM RESPONSE TIMES: A) ARCHITECTURE FROM FIG. 5, B) THE STAND-ALONE WEB SERVER

The page size [kB]	The no. of requests	DNS time [ms]		Initial connection [ms]		The first byte [ns]		Total transfer time [ns]	
		A)	B)	A)	B)	A)	B)	A)	B)
2	2	35	30	64	65	71	70	243	200
100	3	31	31	64	64	74	74	497	464
1.000	5	30	34	66	66	71	70	2.004	1.998

From Table 1 it can be seen that no significant difference in the page load time exists in the case of the web server implementation. Furthermore, it can be concluded that for the smaller number of requests, the total page transfer time is longer. The difference is getting smaller as the number of the requests rises due to the route caching which is performed for the following packets.

The impact of the proposed concept to cyber security is twofold. Firstly, the architecture from Fig. 5 includes the common components which are used to lower the security risk in modern architectures. It provides the network intrusion detection (Fig. 5) which is used to actively monitor the traffic for possible intrusions. Also, it doesn't directly expose the IP addresses of the actual servers, thus it implicitly protects them. Secondly, the architecture from Fig. 5 preserves the IP address of the actual client, thus the server can perform all the regular authentication methods that rely on the client's IP address.

V. CONCLUSION

In this paper the concept of advanced routing and network address translation which preserves the original consumer IP address is discussed, while the consumer is accessing the service through load balancer, network intrusion detection, and proxy. The concept and the routing techniques are described in detail. The results of the system implementation and evaluation are given. In order to implement the routing that will support the communication we utilized the application containerization. The results of the system implementation and evaluation were given. The results show that no significant time overhead was introduced regarding the web page loading times due to the involvement of robust security perimeter.

ACKNOWLEDGMENT

This work was supported by the Serbian Ministry of Education, Science and Technological Development [grant number TR32012].

REFERENCES

[1] Tange, Koen, Michele De Donno, Xenofon Fafoutis, and Nicola Dragoni. "Towards a systematic survey of industrial IoT security requirements: research method and quantitative analysis." In Proceedings of the Workshop on Fog Computing and the IoT, pp. 56-63. 2019.

[2] Radanliev, Petar, Dave De Roure, Stacy Cannady, Rafael Mantilla Montalvo, Razvan Nicolescu, and Michael Huth. "Economic impact of IoT cyber risk-analysing past and present to predict the future developments in IoT risk analysis and IoT cyber insurance." (2018): 3-9.

[3] Varadharajan, Vijay, Kallol Karmakar, Uday Tupakula, and Michael Hitchens. "A policy-based security architecture for software-defined networks." IEEE Transactions on Information Forensics and Security 14, no. 4 (2018): 897-912.

[4] Zhou, Liang, and Han-Chieh Chao. "Multimedia traffic security architecture for the internet of things." IEEE Network 25, no. 3 (2011): 35-40.

[5] Kumar, Sailesh. "Survey of current network intrusion detection techniques." Washington Univ. in St. Louis (2007): 1-18.

[6] Ntuli, Nonhlanhla, and Adnan Abu-Mahfouz. "A simple security architecture for smart water management system." Procedia Computer Science 83 (2016): 1164-1169.

[7] Islam, Shafinaz. "Network Load Balancing Methods: Experimental Comparisons and Improvement." arXiv preprint arXiv:1710.06957 (2017).

[8] Chatel, M. "Classical versus transparent IP proxies." Network Working Group Request for Comments,(Mar. 1996) (1996): 1-35.

[9] Keromytis, Angelos D., and Jason L. Wright. "Transparent Network Security Policy Enforcement." In USENIX Annual Technical Conference, FREENIX Track, pp. 215-226. 2000.

[10] Cardellini, Valeria, Michele Colajanni, and Philip S. Yu. "Dynamic load balancing on web-server systems." IEEE Internet computing 3, no. 3 (1999): 28-39.

[11] Prasetijo, Agung B., Eko D. Widianto, and Ersya T. Hidayatullah. "Performance comparisons of web server load balancing algorithms on HAProxy and Heartbeat." In 2016 3rd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), pp. 393-396. IEEE, 2016.

[12] HAProxy, "The Proxy protocol Versions 1 & 2", March 2020. Link: http://www.haproxy.org/download/1.8/doc/proxy-protocol.txt