WEB Platform for Modeling Network Activity Based on Docker Container Virtualization Technology

Bukhanov D.G.
db.old.stray@gmail.com

Polyakov V.M.
p_v_m@mail.ru

Kalgov I.V.
calgovilya@gmail.com

Redkina M.A
ritushhaa@gmail.com

Belgorod State Technological University named after V.G. Shukhov

Abstract

The paper proposes an approach to creating a system for modeling network activity. To demonstrate the proposed approach, a web-based modeling platform for network activity was developed based on the container virtualization technology. Docker containers were used as virtual machines. Demonstration and testing of the developed system was carried out. The results of the temporal characteristics of modeling the behavior of network activity in the following scenarios: ping, syn-flood and WinFreeze.

1 Introduction

The development of information systems has led to an increase in cybercrime [Sha, Cyb]. To ensure safety, there are a set of legal, economic and technical measures. Technical measures includes following: a system broadcast address and port, intrusion detection systems (IDS) [JNSA16], tunneling traffic systems, firewalls (FW). The tendency to an increase in information security breaches suggests that attackers are one step ahead of the developers of computer security enhancement systems (CS) [Ora, Unp].

One of the aspects of improving security is penetration testing and security threat modeling [dJ16, OS11]. They are designed to identify new and most dangerous threats in terms of protection systems. According to [Cis], the number of network attacks using distributed bot services increased last year. The modeling of such security threats is dealt with in [SBL$^+$17]. These systems are based on KVM technology, which is a Linux kernel loadable module designed to provide virtualization on the Linux platform. In [PR14, ASH17], Netkit is used to emulate network activity. In [BNNK17], open source software CORE is used to simulate network interaction. The disadvantages of these systems are: lack of cross-platform support; problems with adding support for new hardware and software environments; the complexity of setting up a virtualization system; a resource consumption of virtual machines.

Simulation of network interactions allows to test security tools, such as firewall and IDS, and also allows you to demonstrate the peculiarity of a particular network attack.

To simulate network activity, physical or virtual data networks can be used. The advantage of the first is a complete approximation to real conditions, in turn, the disadvantage is the high resource intensity. The second

In: S. Hölldobler, A. Malikov (eds.): Proceedings of the YSIP-3 Workshop, Stavropol and Arkhyz, Russian Federation, 17-09-2019–20-09-2019, published at http://ceur-ws.org

approach require, significantly less resources and use for this purpose virtualization and emulation at the host level.

Table 1 provides a brief overview of the main existing systems for modeling the behavior of network activity.

Table 1: A brief overview of the mainstream virtualization systems

Title	Supported OS	Emulated OS	Limitations	License
VMWare	Windows, Linux, macOS	Windows, Linux, macOS	Price, small number of supported guest operating systems	Commercial
QEMU	Windows, Linux, macOS	Windows, Linux, macOS	Slowness due to lack of hardware acceleration	Open source
Kathara	Windows, Linux, macOS	Windows, Linux	Lack of user-friendly interface to configure gateways	Open source
Xen	Linux, BSD	Windows, Linux	High cost of deployment guest systems due high consumption of RAM	Open source

One of the approaches to virtualization is the use of virtual containers, each of which runs the corresponding operating system. This approach to virtualization uses the Docker development platform. It shields the user from complex virtualization mechanisms, and provides a convenient interface for using.

2 Development of a common structure for a system for modeling network activity of the CS

As a development platform, it is proposed to use web technologies. The advantage of this approach is the possibility of cross-platform use of the system and the provision of an open service without installing specialized software. Figure 1 shows the general scheme of interaction between the elements of the system emulating the behavior of a CS.

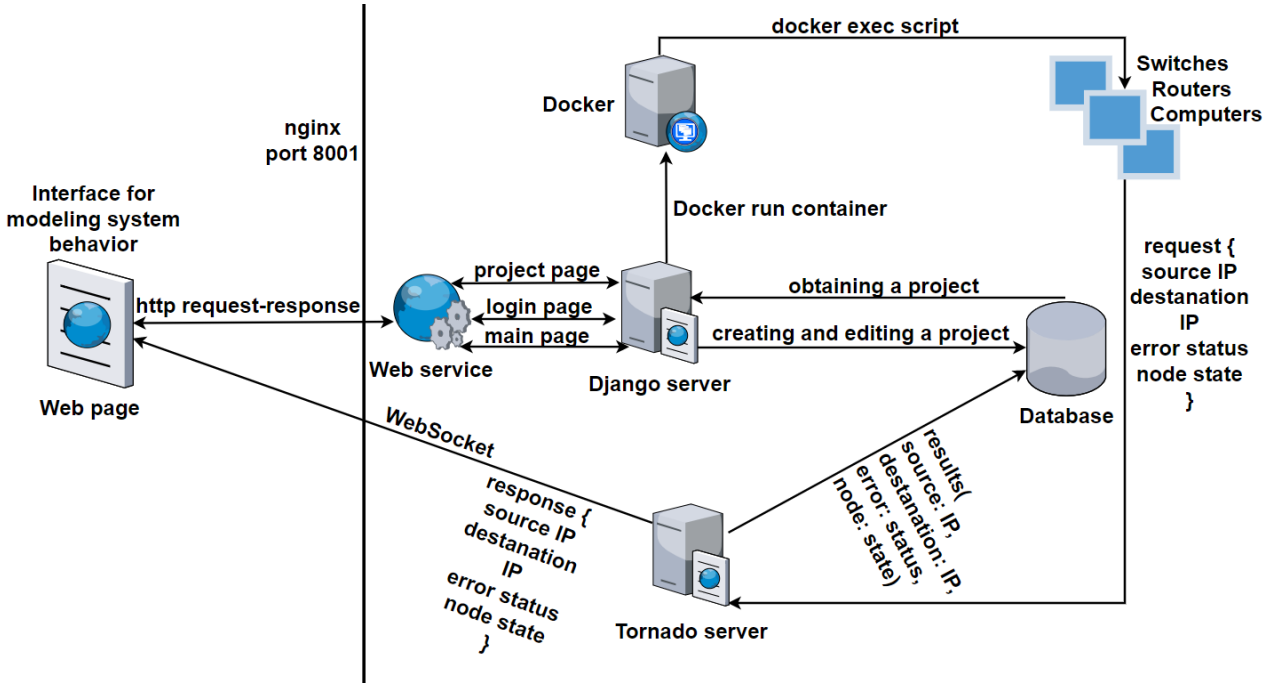

Figure 1: The general scheme of the system of modeling network activity of the CS

Connection to the developed system occurs via the http protocol through port 8001. Nginx is used as a web server. Django was used as a framework for developing server-side applications. Upon successful authorization and transition to the simulation page, the application server creates a new virtualization project and saves it to the database (DB). Adding a new node to the workspace binds it to the current project and puts the selected parameters for it in the database. The following components are available for building a network topology: computers, switches and routers, and communication lines. All components are emulated on Ubuntu 18.04 LTS with the creation of appropriate network interfaces. After building the computer network topology, the server software sends the Docker creation commands for configuring the containers with the parameters specified for them. Each Docker-container contains all possible networking scenarios.

After the configured computer network is started, one of the predefined network interaction scripts is selected and executed on the already running Docker containers that are already running. Figure 2 shows a diagram of how to run networking scenarios through a web interface.

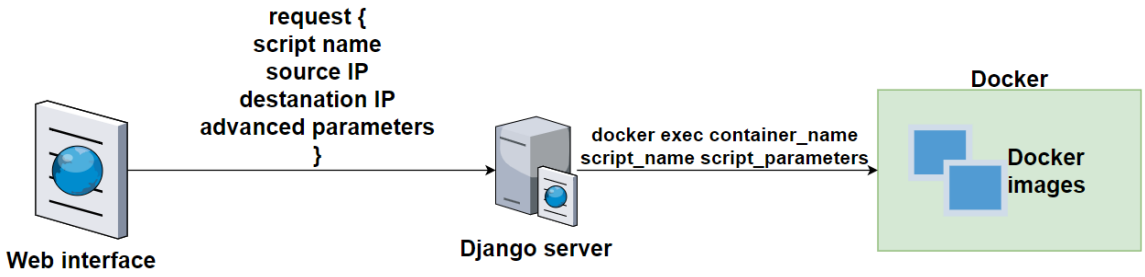

Figure 2: The scheme of running network interactions scenarios

The user in the web interface select the script to run, enter the required parameters and send an http request to the Django-server. The request contains: script name, source address (IP of the container from which script will be run), destination IP and additional parameters. The server retrieves the script startup parameters, configure the docker exec command according to the received parameters, and then execute. Executing the command will run a network communication script on one of the containers.

Here is an example of running the script ping. Select the script "ping in the web interface, enter 192.168.0.2 in the "source Address" and 192.168.0.3 in the "destination Address". Additionally, you can fill in the "number of packages" and "package Size" fields, initially they contain the values 3 and 64 respectively. After filling in the required fields, click "Run". Clicking this button generates an http packet containing the specified script launch parameters and send it to the Django server. The server receives the packet, extracts the parameters run a script that generates the command "docker exec container02 python3 ping.py -c 3-s 64 192.168.0.3 &" and execute it. Executing the command activate the "ping" script on the container that corresponds to the IP address 192.168.0.2. The container sends 3 ICMP at 64K on the container with an IP address of 192.168.0.3, then the script exits.

During the execution of scripts, Docker-containers send a request about their state, containing the following fields: IP address of the sender, IP address of the recipient, error status, structure state of the node, containing information about the state of the node on the server. Tornado server and WebSockets are used as a receiver of this data. Tornado returns a response structure, whose fields match the request, to the network activity modeling page to visualize the process. Tornado saves intermediate results to the database. Results contains the following fields: source address, destination address, error status, and site status structure. In the error status field, 0 is written if the packet was successfully delivered, otherwise, the error code, the node state structure is abstract, because of the script that was selected will depend on which fields it will contain. Figure 2 shows the system state diagram.

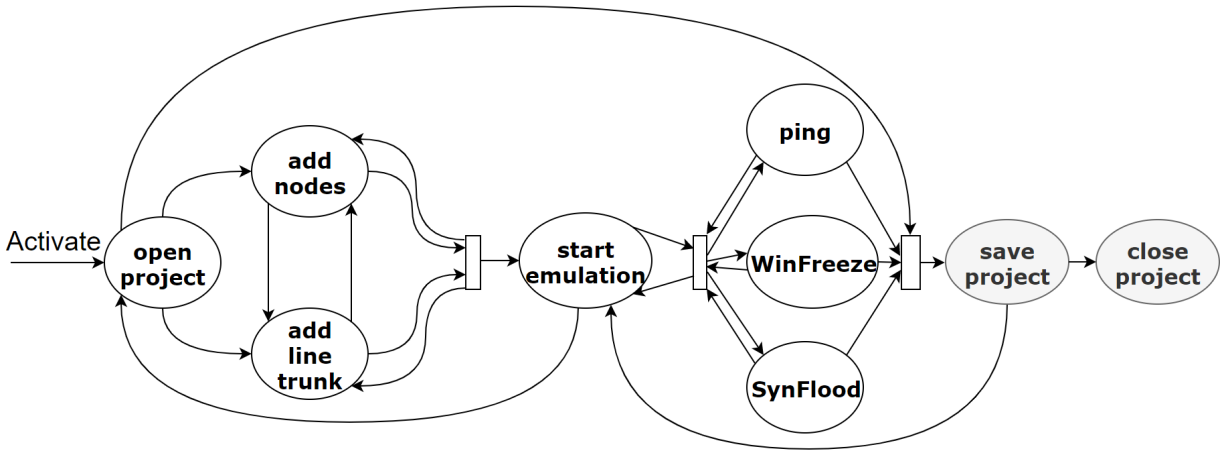

Figure 3: State diagram of the system modeling the behavior of network activity

The initial state is the open project state. From it you can go to the following states: add node, add link, start emulation, save project, close project. From the add node state, a transition to the add link state is possible, a reverse transition is also available. From these two states there is a transition to the states start emulation, close project, save project. From the start emulation state you can return to the initial open project state in order to reconfigure the network topology and restart the project. Also from this state there are transitions to the ping, SYN-flood and WinFreeze states. These states assume the execution of the corresponding python scripts. After executing the corresponding script, the system returns to the start emulation state, or goes to the save project or close project state. From the save project state, it is possible to switch to the start emulation and close project states. The close project status means the shutdown of the network activity modeling system.

For testing the developed system, three types of scripts are implemented: ping, SYN-flood and WinFreeze. The ping script is designed to verify the integrity of the connection between two nodes. To perform it, the user enters the following parameters: the source address, the destination address, the number of packets, the size of each packet. The script collects ICMP packets with the parameters specified by the user, calculates a checksum, encapsulates them into IP packets, sends them and waits for an echo reply or timeout. Returned responses are sent to the Tornado server and contain the fields for the destination address, the source address, error status, packet size, sequence ID, TTL, response time. In the event of a timeout, only the direction, destination and error status are sent to the server.

The SYN flood script implements a denial of service attack. The user must specify the IP address of the machine from which the attack will be conducted, the IP address and port of the victim. The attacker's computer overflows the connection queue on the victim's computer, sending SYN requests. SYN + ACK packets received from the victim are ignored, keeping its ports in a half-open state. The script keeps the queue full so that the connection to the ports of the victim is difficult or impossible [BSR13]. When receiving a SYN + ACK packet, the script sends a packet to the Tornado server containing the attacker's IP, the victim's IP and port number.

The WinFreeze script causes the victim's computer to attack itself. The user specifies the attacker's IP and attack target IP. The attack is performed inside the victim's network. The attacker sends ICMP redirect messages on behalf of the router to the target address, informing about the choice of a non-optimal route and the need to add the best route to the routing table. All victim packets are redirected back at the victim by attacker. Due to the large number of such messages, the victim will spend most of his resources processing packets and attempting to modify the routing table. With each packet received, the script sends to the Tornado server a packet containing the attacker's IP, the victim's IP and the number of packets already forwarded. Figure 3 shows a web interface with an example of network topology.

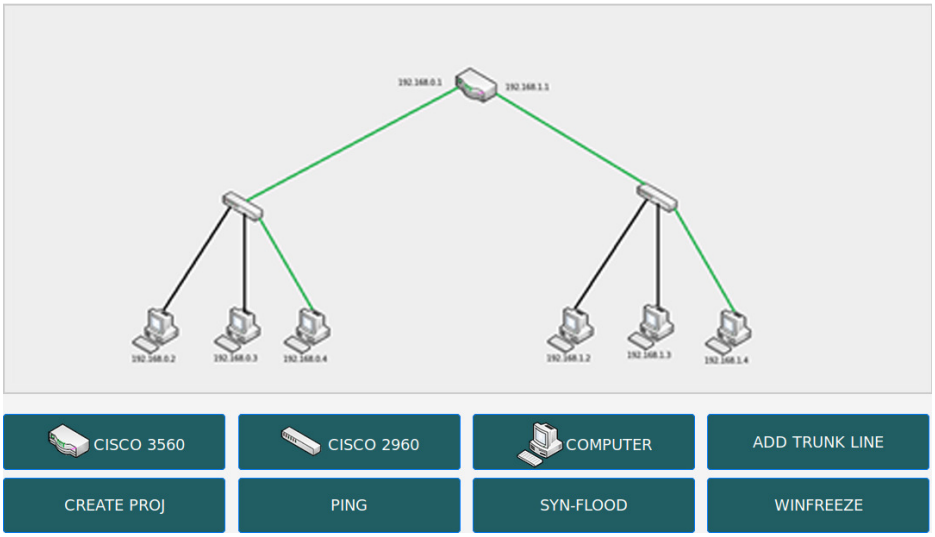

Figure 4: Web-interface with an example of network topology

The figure shows a topology with two subnets with the following addresses: 192.168.0.0/24 and 192.168.1.0/24. Green lines indicate routes that ping between nodes 192.168.0.4 and 192.168.1.4. Under the working area there are buttons for adding switches, routers, computers and communication lines, a button for starting the emulation, as well as buttons for emulating network activity: ping, syn-flood and WinFreeze.

3 Experimental results of the time characteristics of the developed system for modeling the behavior of network activity

Figure 4 shows the graphs of the delay between the experiment performed and the visualization time.

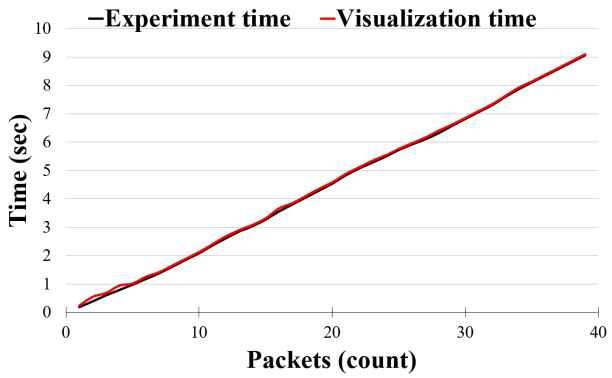

Figure 5: Time to visualize the results of experiments

The time of visualization of the experiment coincides with the time of its holding. The graph shows that the time between the experiment and the transfer of results to the user almost coincides. Due to the lack of desynchronization of time intervals on the main OS and containers, it allows the user to connect to the corresponding container to conduct a more detailed analysis of the script.

Figure 5 shows the graphs of time when a packet is transmitted in a real physical network and within the developed modeling system. During testing packet size of 62535 bytes was used.

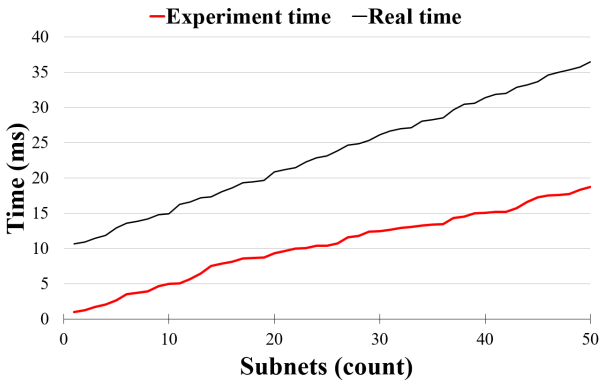

Figure 6: Temporal characteristics of packet transmission between IP subnets

Experiments show that the use of virtualization significantly reduces the simulation time. This is due to the following factors: packets are not sent outside the network interface of the host, containers of the same subnet are connected to a common virtual interface such as a bridge. As the number of devices grows, and consequently the number of containers deployed by Docker, the simulation time starts to slow down. The limit value of the maximum number of running containers is determined by the resources of the server running the network activity modeling system.

The results of the work in comparison with the Kathara system are show in table 2.

Table 2: A brief overview of the mainstream virtualization systems

Number of Subnets	Kathara (ms)	WEB platform (ms)
1	1.394	0.976
2	1.957	1.253
5	3.009	2.669
10	5.582	4.989
25	12.198	10.391
50	22.446	18.74

As a comparison tool was used the standard utility "ping" with the help of which packets of 62535 bytes were transmitted between different number of subnets.

4 Conclusion

The paper proposed an approach to creating a system for modeling network activity. A web platform for modeling network activity was developed based the Docker containers. A study of the developed system for compliance with the requirements. The study showed the possibility of using the Docker platform for the implementation of a network activity modeling system. All network attacks carried out in the created virtual network work equivalently to real network. The time delay between the experiment being conducted and the virtualization time does not exceed one second. The proposed approach and the developed web-platform can be used in testing and developing mechanisms for countering network threats. A feature of the developed system is the ability to package in a single Docker image for rapid deployment.

References

[ASH17] Yuri Ariyanto, Yan Watequlis Syaifudin, and Budi Harijanto. Performance analysis of network emulator based on the use of resources in virtual laboratory. In *2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*. IEEE, September 2017.

[BNNK17] Cristian Hernandez Benet, Robayet Nasim, Kyoomars Alizadeh Noghani, and Andreas Kassler. OpenStackEmu — a cloud testbed combining network emulation with OpenStack and SDN. In *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, January 2017.

[BSR13] Mitko Bogdanoski, Tomislav Shuminoski, and Aleksandar Risteski. Analysis of the SYN flood DoS attack. *International Journal of Computer Network and Information Security*, 5(8):15–11, June 2013.

[Cis] Cisco 2018 annual cybersecurity report.

[Cyb] Cybersecurity threatscape 2018: trends and forecasts.

[dJ16] Rina Elizabeth Lopez de Jimenez. Pentesting on web applications using ethical - hacking. In *2016 IEEE 36th Central American and Panama Convention (CONCAPAN XXXVI)*. IEEE, November 2016.

[JNSA16] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. ACM, 2016.

[Ora] Oracle weblogic deserialization rce vulnerability (0day).

[OS11] Xinming Ou and Anoop Singhal. Security risk analysis of enterprise networks using attack graphs. In *Quantitative Security Risk Assessment of Enterprise Networks*, pages 13–23. Springer New York, October 2011.

[PR14] Maurizio Pizzonia and Massimo Rimondini. Netkit: network emulation for education. *Software: Practice and Experience*, 46(2):133–165, May 2014.

[SBL+17] Nils Schmidt, Lars Baumgartner, Patrick Lampe, Kurt Geihs, and Bernd Freisleben. MiniWorld: Resource-aware distributed network emulation via full virtualization. In *2017 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, July 2017.

[Sha] Internet issues & availability report 20182019.

[Unp] Unpatched zero-day vulnerability in social warfare plugin exploited in the wild.