The testbed for definition of the exploit’s execution features to detect and score cyber attacks

Sergei Verevkin¹², and Elena Fedorchenko¹*

¹St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS), St. Petersburg Federal Research Center of the Russian Academy of Sciences (SPC RAS), 39, 14th Liniya, St. Petersburg, 199178, Russia
²A.F. Mozhaysky Military-Space Academy, 13, Zhdanovskaya str., St. Petersburg, Russia

Abstract. The paper considers the deployment of the testbed for definition of the exploit’s execution features to detect and score cyber-attacks. The paper describes the place of the proposed testbed in the approach to the cyber-attack detection and scoring. It defines the requirements to the testbed considering the introduced approach. The testbed infrastructure is defined and deployed considering the specified requirements. The introduced testbed infrastructure is justified. Finally, the technique of the testbed implementation for the feature extraction is introduced. The proposed testbed and technique allow combining advantages of the static and dynamic approaches to the exploits analysis and detecting and scoring known and previously unseen cyber-attacks.

1 Introduction

The testbeds, namely, sandboxes are widely used in malware analysis for unknown malware execution [1, 2]. There are best practice sandbox systems used for the behavioral malware analysis (or dynamic analysis) such as Cuckoo Sandbox [3], WSandbox [4], Zerowine [5], and others. One of possible applications of such testbed is definition of the exploit’s (or malware) execution features including system calls and API calls. The exploits should be executed within the controlled testbed environment for this goal. In this research the authors put forward a hypothesis that novel malware samples can include the elements of the already known exploits. Thus, outlining of the functional blocks of execution of different exploits and detection of indicators of their execution in real time can help detecting of both known and unknown exploits, and forecasting the information security risks.

To validate the hypothesis, the authors introduce the approach represented in Figure 1. It is based on the general semantic functional model of the exploits compiled code [6]. It reflects the order and dependencies of calls of imported "names" for the dataset of publicly known exploits: GSG = (V, E, λ), where V - nodes of the graph that correspond to "names" extracted from the executable code, E - edges representing the "names" usage dependencies from the importing modules, λ - the frequencies of dependencies between the vertexes of the graph.

* Corresponding author: doynikova@comsec.spb.ru
(namely, how often the transition between two "names" happen within the exploits compiled codes). The introduced approach incorporates the following stages: (1) generation of the general semantic functional model of the exploits compiled code (based on the static analysis); (2) step by step execution of the exploits within the controlled environment to define the dynamic characteristics based on the logs and network traffic. These dynamic characteristics correspond to the semantic functional model’s nodes generated on the basis of the static characteristics; (3) generation of the enhanced general semantic functional model of the exploits compiled code (based on the dynamic analysis); (4) mapping of the detected dynamic characteristics of exploits execution on the enhanced general semantic functional model in process of system operation and forecasting of the cyber-attack propagation and of the cyber security risks. The testbed is required to define the dynamic characteristics of the exploit’s execution.

Fig. 1. Approach for attack detection and scoring based on exploits analysis.

In this paper we consider the task of the testbed development and deployment. We propose using the testbed to detect external features of the exploit implementation. Namely, in scope of our research we specify the requirements to the testbed, analyse the alternatives of the testbed implementation, propose own testbed implementation, and we introduce the technique for the testbed exploitation to extract features of the exploits execution to detect cyber-attacks.

Thus, the contribution of this paper is as follows:

- The requirements to the testbed for definition of the exploit’s execution features to detect cyber-attacks.
- The testbed architecture and implementation.
- The technique for the testbed exploitation to extract features of the exploits execution to detect cyber-attacks.

The paper is organized as follows. Section 2.1 specifies requirements to the testbed for definition of the exploit’s execution features to detect cyber-attacks. Section 2.2 introduces the testbed architecture and implementation. The section 2.3 describes the technique for the
testbed exploitation to extract features of the exploits execution to detect cyber-attacks. The paper ends with conclusion and future work prospects.

2 The testbed and technique for definition of the exploit's execution features

In this section we specify the requirements to the testbed for definition of the exploit’s execution features to detect cyber-attacks (Section 2.1), we describe the testbed architecture and implementation (Section 2.2) and the technique for the testbed exploitation to extract features of the exploits execution to detect cyber-attacks (Section 2.3).

2.1 Requirements to the testbed

Considering the research goal, namely, definition of the exploit’s execution features to detect cyber-attacks, the testbed should provide the following functionality:

- Simulate real network infrastructure,
- Simulate the vulnerable nodes,
- Automate the cyber-attacks,
- Allow data gathering while cyber-attacks implementation (network traffic and logs).

Considering the existing solutions and the research tasks the authors specified the following non-functional requirements:

- Flexibility and scalability: it will provide the ability to generate various network topologies with support for multiple virtual machines and connections between them, allowing one to emulate real network environments and analyze malicious code under conditions that are as close as possible to the real-world conditions.
- Reproducibility: the ability to quickly restore the testbed's performance in case of a failure of the virtual hosts under attack.
- Isolation and security: isolation of network segments and virtual machines with different operating systems used to launch malicious code. This approach will ensure security by preventing the possible spread of malicious code.
- Extensibility: support for interoperability with malware analysis tools such as Wireshark, Snort, and others.

2.2 The testbed

The authors analyzed three alternatives to implement the testbed: building a real hardware and software stand; using virtualization tools to simulate a virtual network; using a network emulation environment. Let us consider the advantages and disadvantages of each method.

Building a real hardware and software testbed allows deploying real world vulnerable configurations, it allows taking into account the vulnerabilities of the software used, including firmware versions of network equipment. The key disadvantages of this approach are the cost of hardware components of the testbed and the lack of possibility to restore the performance of the testbed in case of failure of network nodes.

The use of virtualization tools for modeling the virtual network of the organization allows reducing the cost of hardware components and place it on a single server (if there is sufficient capacity). In particular, the organization of network communication, can be implemented by built-in functions of the virtualization environment, such as VMware Workstation or VirtualBox. However, such a solution excludes the possibility of determining the consequences of attacks on telecommunications equipment. Also, despite the ability to clone virtual machines, it takes considerable time to restore the testbed's operability.
The use of network emulation environment is the most preferable considering the specified requirements. This approach incorporates most of the advantages of the previous approaches, as it avoids the costs associated with building a real prototype, and also allows using firmware images of network equipment from different manufacturers to emulate the operation of telecommunication devices. Another advantage is the ability to create virtual machines and save their state for rapid deployment in case virtual nodes are damaged as a result of attacks.

The authors analysed the following network emulation environments: Cisco Packet Tracer, GNS3, OMNeT++, and EVE-NG. As a result, considering the need for exploit code analysis, the most appropriate network emulation environments are GNS3 or EVE-NG. Both solutions allow creating virtual networks, test various network devices, and analyze network security technologies to detect and investigate malicious code. We have chosen EVE-NG considering its popularity and availability of the materials. Free version of EVE-NG allows simulating a network with up to 64 devices, which fully meets the objectives of the study. It provides a wide choice of simulated network equipment from the most popular manufacturers, easy conversion of virtual machines for use as a part of the testbed, as well as the ability to collect telecommunication data at any point of the network. To deploy the EVE-NG environment, a PC based on Windows 10 Pro is used.

The authors developed the test network configuration represented in Figure 2. The vulnerable network configuration is extended by the equipment required for the exploits execution and analysis. The network nodes include virtual machines and network devices. The following types of nodes are identified: vulnerable machines (VPC 1-5), attacker's host (VPC 6), network equipment and database server with logs/scanner (VPC 7).

Virtual machines with a pre-installed vulnerable configuration are selected as the hosts for the attacks implementation. We use a set of vulnerable virtual machines Metasploitable v1,2 and 3, as well as popular virtual machines used for practicing penetration testing skills from such sources as Vagrant [7] and VulnHub databases. The selected virtual machines initially have a vulnerable configuration, which makes it possible to perform vulnerability testing of specific software and hardware. However, due to the daily updating of existing vulnerability databases, the vulnerabilities of each virtual machine within the testbed were repeatedly scanned using the OpenVAS vulnerability scanner. Based on the scanning results, a list of current vulnerabilities that have an exploit was generated. Exploit search was performed on the basis of open databases such as ExploitDB. The test results are presented in Table 1.

![Fig. 2. Configuration of the testbed in EVE-NG.](image-url)
Table 1. The results of analysis of the virtual machines for the vulnerabilities and exploits.

<table>
<thead>
<tr>
<th>Virtual Machines</th>
<th>CVE With exploit</th>
<th>Low severity</th>
<th>Mid severity</th>
<th>High severity</th>
<th>Total CVE</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metasploitable 1</td>
<td>13</td>
<td>2</td>
<td>21</td>
<td>11</td>
<td>72</td>
<td>13</td>
</tr>
<tr>
<td>Metasploitable 2</td>
<td>17</td>
<td>3</td>
<td>17</td>
<td>13</td>
<td>70</td>
<td>17</td>
</tr>
<tr>
<td>Metasploitable 3</td>
<td>13</td>
<td>7</td>
<td>13</td>
<td>16</td>
<td>57</td>
<td>13</td>
</tr>
<tr>
<td>spoonable/vulnerable-ubuntu-20.04</td>
<td>6</td>
<td>3</td>
<td>10</td>
<td>5</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>Vulnerable machine CTF (from HackTheBox)</td>
<td>8</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>27</td>
<td>8</td>
</tr>
<tr>
<td>ZeroLogonLab</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>13</td>
<td>4</td>
</tr>
</tbody>
</table>

Each virtual machine is set up once. Then the authors use the Qemu utility to convert it to the format used in EVE-NG. It allows avoiding the challenges in case of the testbed failure.

The database server with logsscanner is a virtual machine with a pre-installed vulnerability scanner to identify attacks used in experiments. Another functionality of this virtual machine is the aggregation of data from the attacked hosts - collecting log files, traffic, etc.

2.3 The technique

The technique for the testbed exploitation to extract features of the exploits execution to detect cyber-attacks incorporates the following stages: (1) collecting a dataset for the experiments; (2) compiling and validating the collected exploits; (3) running exploits sequentially step-by-step within the testbed; (4) collecting and labelling the data; (5) extracting the features mapped to the corresponding code fragments.

We use ExploitDB [8] as a dataset of the exploits for the experiments at the first stage. The second stage involves compilation and validation of exploits.

The authors execute the exploits step-by-step on the third stage due to the proposed approach and to the hypothesis put forward in the introduction that different malicious software may include elements of already known exploits, and thus, by identifying the functional blocks of execution of different exploits and determining the features of their execution in real time, it is possible to detect the execution of both known and unknown exploits and forecast the security risks to the system. Thus, it is required to execute the exploit step by step for the subsequent construction of the semantic model. This will also allow labeling the data such as logs and traffic in the fourth stage.

The gathered labelled data are used to extract the features of the exploits execution for further development of the algorithms for mapping the detected features on the extended general semantic functional model of the exploits compiled code and forecasting and scoring the cyber-attacks in the system operation mode.

3 Conclusion

The paper proposed a combined static and dynamic approach to detecting and scoring cyber-attacks. To implement the proposed approach, a testbed was developed to dynamically analyze exploits used in cyber-attacks. The requirements to the testbed were specified,
modeling tools were selected, and a list of vulnerabilities and exploits to be tested to obtain the initial data was defined. As a result of this work, a virtual testbed of the network was deployed, containing the required vulnerabilities, as well as the means to conduct attacks and collect data. The technique for definition of the exploit's execution features using the deployed testbed was introduced. In future work it is planned to conduct experiments on the testbed and to develop an automated system of information security risks assessment.

This research is being supported by the grant of RSF #23-21-00498 in SPC RAS, https://rscf.ru/en/project/23-21-00498/.

References