BUILDING A CONTINUOUSLY INTEGRATING SYSTEM WITH HIGH SAFETY

Tuan Nguyen Kim[1], Ha Nguyen Hoang[2,*], Vy Huynh Trieu[3]

[1]School of Computer Science, Duy Tan University, Da Nang, Viet Nam
[2]University of Sciences, Hue University, Vietnam
[3]Phạm Van Dong University, Quang Ngai, Viet Nam

ABSTRACT

In this paper, we propose and implement an internal continuous integration system, based on two open-source tools Jenkins and GitLab, taking into account the safety factor for servers in the system. In the proposed system, we use a combination of firewall function and reverse proxy function to protect Jenkins server itself and reduce the risk of this server against attacks on the CVE-2021-44228 security vulnerability, may exist in plugins of Jenkins. This system is highly practical, and it can be applied to immediately protect service servers when a vulnerability in it has been discovered but the corresponding patch has not been found or the condition to update the patch is not allowed yet.

KEYWORDS

Continuous Integration, Continuous Delivery, CI/CD, CVE-2021-44228, Firewalls, Jenkins, Gitlab.

1. INTRODUCTION

Continuous Integration (CI) and Continuous Delivery (CD) are two core processes in DevOps software development methodology, a modern software development methodology that has many advantages compared with the traditional software development method and is deployed in most enterprises and software development groups today. The CI process and the CD process are seen as two sides of the same coin, but they are used in conjunction to make the development of a software product, from the moment a new version of the source code is released. commit until the latest product is deployed, making it simpler, faster and more efficient. Full, detailed information, related to DevOps, CI and CD, is not within the scope of this study, so we do not present it in detail here, it can be found in [1-3].

Computer network infrastructure, to implement related services, is necessary so that an enterprise, or a group, of software development can deploy CI, and CD processes automatically into the development cycle. software towards their DevOps approach. The servers that perform the CI server and CD server functions play an important role in this network infrastructure, so they need to be protected to be able to resist attacks on themselves, or on the service being used. deployed on it. This network infrastructure can be deployed in the Internet space (based on Cloud services) or in the intranet (Intranet/Extranet), depending on the requirements, conditions and scale of each software development enterprise... CI/CD network infrastructure on the Internet or in the intranet has its own advantages and disadvantages. In this paper, we propose a network infrastructure that only ensures the automation of the CI process in the intranet, it is called "Internal Continuous Integration System", because we are interested in speed, availability, and proactiveness in its security deployment.

The two main components of any continuous integration system are the server that performs the continuous integration function (CI server) and the server that stores and manages the source code (Repository server). If these servers are "taken down" by an attacker, the system will not only stop working but the data stored in it can be intentionally exploited and stolen. This danger is potential, because these servers, whether located on the Internet or inside an intranet, must have sharing, availability, and simultaneous multi-connectivity. In addition, attackers not only attack the CI servers or Repository servers themselves but also find ways to exploit security vulnerabilities that may exist in the software used to build these servers. Thus, ensuring the security of the CI server and Repository server is necessary, many different techniques and tools can be used for this task, but we choose to use the firewall for the system suggested in this article. In our system, the firewall not only performs the main task of protecting the CI server (using Jenkins) and the Repository server (using Gitlab) but can also prevent attacks that exploit CVE-security vulnerabilities- 2021-44228 exists in several plugins of the open source Jenkins tool. In this case, we would like to introduce another way to 'close' a vulnerability that exists in a service server, here is Jenkins, without needing, or without, using the corresponding patch. Of course, using a patch to "seal" the vulnerability is the most often chosen solution, but in the condition that the patch is not ready or the time to perform the "patching" of the vulnerability has not yet allowed, the use of firewall to temporarily "seal" the discovered vulnerability is a proposal that should be considered for implementation.

2. RELEVANT KNOWLEDGE

2.1. Continuous Integration System based on Jenkins and Gitlab

In this section, we only present the most basic information related to a continuous integration system and the main functions of two open-source tools Jenkins and GitLab. The reason for choosing to use Jenkins and Gitlab to build the recommendation system is also mentioned here. More details on these topics have been found in [4-7].

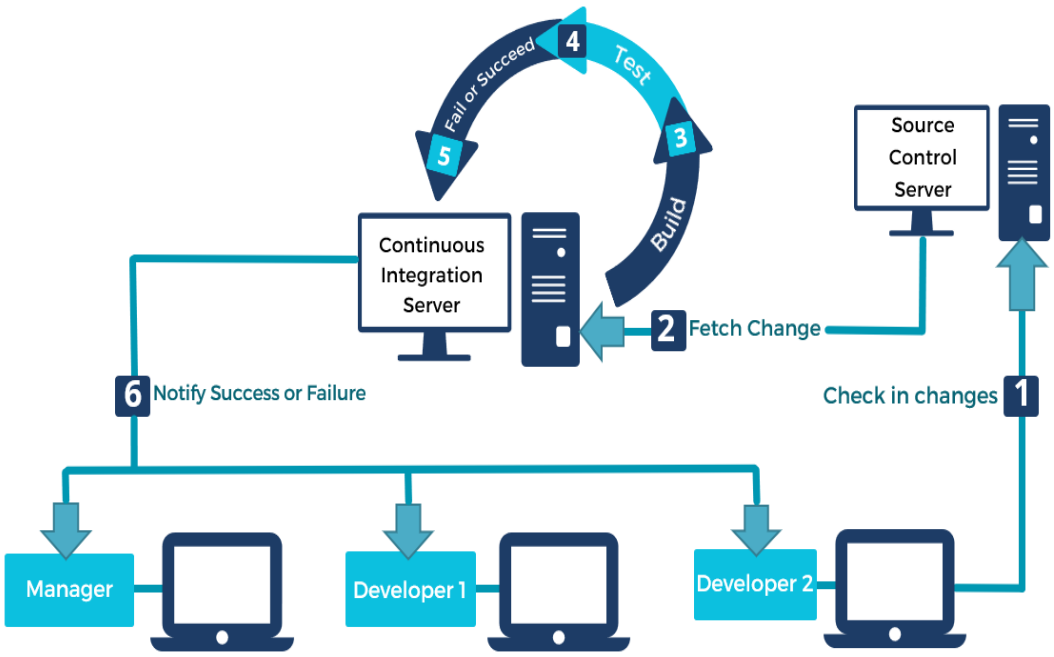

Figure 2.1: Components and operating principles of an internal CI system

2.1.1. Regarding Continuous Integration Systems

Figure 2.1 shows the main components, and operating principles, of an internal CI system. Which, Continous Integration Server (CI server) plays the role of controlling the continuous integration process and makes this process work automatically. i) The Source Control Server (Repository server) is responsible for storing and managing the source code files, which it receives from the Manager and/or from the Developers. This server implements a reliable solution to centralize and maintain code changes made by developers (1: Check-in Changes). ii) CI server continuously, or periodically, detects code changes on the Repository server, performs receiving the latest code version from the Repository server (2: Fetch changes) to perform the assigned tasks of the Repository server. it (3: Build; 4: Test; 5: Fail or Succeed), and then send the result (6: Notify success or failure) back to project members, managers and developers.

Thus, maintaining a high level of connectivity, both in terms of speed and availability, between CI server, Repository server, Manager computers, and Developer computers is required with this system. Our recommendation system is particularly concerned with the availability of the CI server and the Repository server.

2.1.2. Regarding the reason for choosing Jenkins to build CI server

Currently, on the market, there are many tools available to support building CI servers for CI/CD systems, such as Buddy, TeamCity, Jenkins, GoCD, CodeShip, etc. According to our research, the open-source Jenkins tool has more advantages and is more widely used, compared to similar products, because: Jenkins is developed from Java, so its cross-platform compatibility is very high; the number of Jenkins plugins is large and diverse in functionality; Jenkins users get great support from the developer community; the Jenkins administrator can perform configuration operations on it through both the pseudo-GUI interface and the console; This is why we use Jenkins to build CI server for recommendation system.

The issue of ensuring the security of the operation of the Jenkins server is also focused on by Jenkins, it allows the administrator to implement the available security options, such as Access Control; CSRF Protection; Controller Isolation; Content Security Policy; etc Easily. However, in this study, we only mention the solution to protect the Jenkins server, and the countermeasures to the CVE-2021-44228 vulnerability of Jenkins, in the direction of using the Firewall.

2.1.3. Regarding the reason for choosing GitLab to build Repository server

GitLab is also an open-source tool, it is often used to build storage servers, and source code repository servers (Source code repository servers) for CI/CD systems. We can use one of the following available products as an alternative to GitLab, like Git, Github, Apache Subversion, Bitbucket Server, Team Foundation Server, etc., as they all provide basic support for the version a repository server needs, such as supporting a mechanism to track changes in a stored source code files; support mechanism that allows multiple members (developers) to access the source code file concurrently; support conflict resolution mechanism when many developers simultaneously want to change the content of a source code file; support a web interface for developers to see the change between two versions of source code, the previous version and the recently committed version, visually; etc.

In the proposed system, we use Gitlab to build a Repository server because of its outstanding advantages, such as the unlimited free, ability to integrate with Git, allowing dynamic security checks, allowing projects to be carried out according to a plan, etc. In particular, GitLab allows us to restrict access to projects and view the compliance status of each participant; In the scope of

this study, we did not address security issues related to GitLab, which will be clarified in a future publication by us.

2.2. Security vulnerability CVE-2021-44228

The Log4Shell vulnerability [8], discovered in 2021 by Alibaba's Chen Zhaojun and identified as CVE-2021-44228, belongs to the group of software security vulnerabilities, that exists in Log4j2. Log4j2 is an open-source library based on the Java language, which supports the logging of error messages, and related information, generated from Java application programs. Log4j2 is used quite commonly, especially, built into the Apache web server.

The CVE-2021-44228 security vulnerability is considered one of the most serious vulnerabilities discovered, up to now, it allows attackers to perform a remote attack to take control of the target. vulnerable, usually a device on the Internet, if that device is running an instance of Log4j2, without authentication. To perform this attack, an attacker just needs to send a request containing the exploit code (payload), via one of the protocols LDAP, RMI, DNS, CORBA, etc., to the server using it. Log4j2, to enable the vulnerability in Log4j2.

Then, through one of the services controlled by the attacker, the server will make a request on JNDI (Java Naming and Directory Interface) to send back to the attacker. Upon receiving this request, the attacker returns a path to a remotely hosted Java class file, which is then injected into the stream by the attacked machine, allowing the attacker to execute optional remote code execution (RCE: Remove Code Execute). Figure 2.2 shows the operation of the attack on the CVE-2021-44228 security vulnerability.

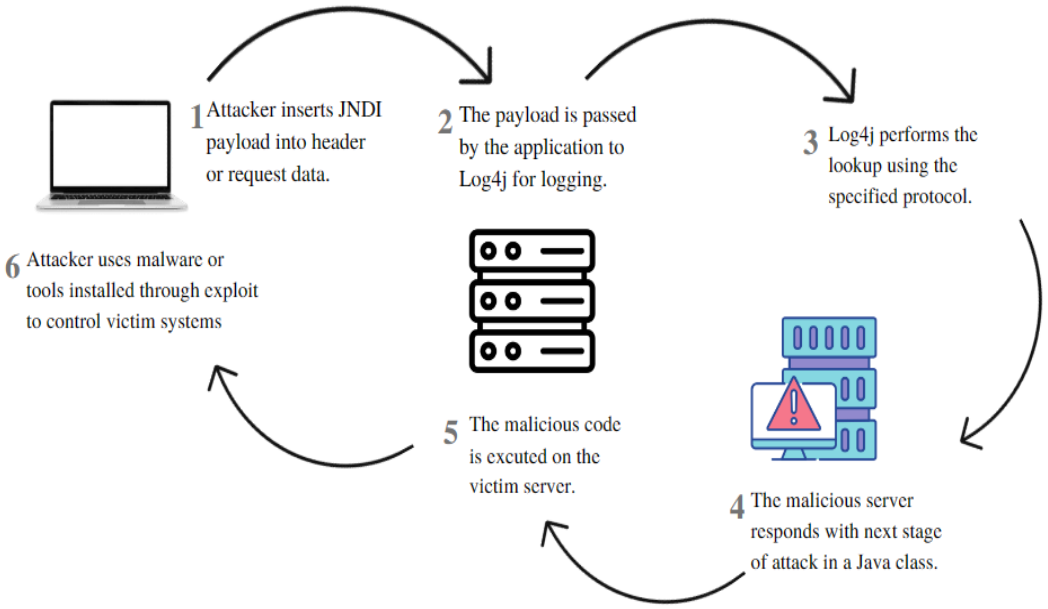

Figure 2.2. Operation of the attack on the CVE-2021-44228 security vulnerability

As such, the huge potential harm in the Log4Shell security vulnerability is clear, but unfortunately, the Log4j library is used in quite a few Jenkins plugins (Log4j doesn't exist in the core of Jenkins), like Audit- log; bootstraped-multi-test-results-report; make builder; cucumber-reports; It is for this reason that ensuring the safety of CI servers using Jenkins in CI/CD systems is always of special interest to system administrators.

Currently, there are many solutions offered to deal with the CVE-2021-44228 security vulnerability such as Update to the latest version of the Log4j library (v2.17.0); disabling the search function in messages; removing the JndiLookup class from the classpath; using the rules table in the packet filtering firewall to limit connections to servers and services that use the Log4j library; Since an attacker can inject Log4j exploit code into any HTTP headers and HTTP body components, in this study we use a packet filtering firewall to both protect the CI server itself and reduce the risk. caused by a Log4Shell vulnerability that exists in this server.

2.3. The Role of Packet Filtering Firewall in Network Security

Detailed information on the operation of firewalls and packet filtering firewalls is presented in [9-10], so in this section, we only present the characteristics of this type of firewall.

The firewall filter packets based on a network access policy, designed in the form of a Packet Filter Rule Table, and the related information contained in the traffic stream, usually IP Address, Protocols, and Port number, to decide to allow a flow of traffic, or a certain packet, is allowed to pass through it. The packet filtering rule table is set up and installed by the network security administrator in the firewall, it is the basis for the packet filter to make the decision to allow the packet to the firewall to pass through it, to reach the destination of the packet, or not.

Figure 2.3 is a diagram illustrating an enterprise network, in which using a packet-filtering firewall as a gateway to the network, all packets from the Internet to the internal network and from the inner network to the Internet must go through the firewall.

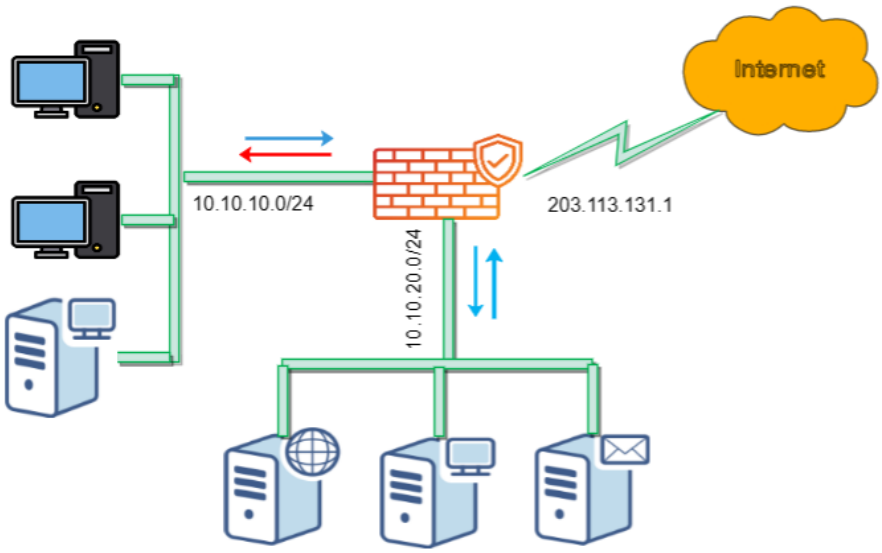

Figure 2.3. Network diagram with the presence of packet filtering firewall

The network access policy of this enterprise network is as follows: Only packets from the Internet are allowed to enter network zone 192.168.2.0/24 - the DMZ zone of the enterprise network - are not allowed to enter network zone 10.10.10. /24, the internal user network area. However, all traffic from the user network area and the DMZ network area can go to the Internet.

In this study, we use the Iptables tool combined with the Nginx tool (Reverse proxy) to build firewalls for the proposed system [11], it both protects the servers inside the network and can prevent Block attacks on the Log4j vulnerability.

Again, a reverse proxy server is a server that acts as a bridge between client computers and servers. This server controls the requests sent from the clients, and coordinates these requests to the appropriate server so that the requests are processed. When the server has finished processing, it will return a response to the Reverse proxy, and the Reverse proxy will return that response to the client sending the request.

3. THE PROPOSED INTERNAL CONTINUOUS INTEGRATION SYSTEM

3.1. System Diagram

In this section, we propose an internal continuous integration system that takes into account the safety factor of the CI server and the Repository server. Figure 3.1 is the network diagram of this system.

In this diagram: The Jenkins server machine plays the role of CI Server; The Gitlab server machine plays the role of Server Repository; the Web server and Mail server are two basic components of an Intranet; Routers are used for connecting the internal network to the Internet; Firewall is used to protect servers, protect the internal network area and perform the function of a Reverse proxy server for http, https and some other protocols (serves to prevent attacks on CVE-2021-44228 vulnerability).

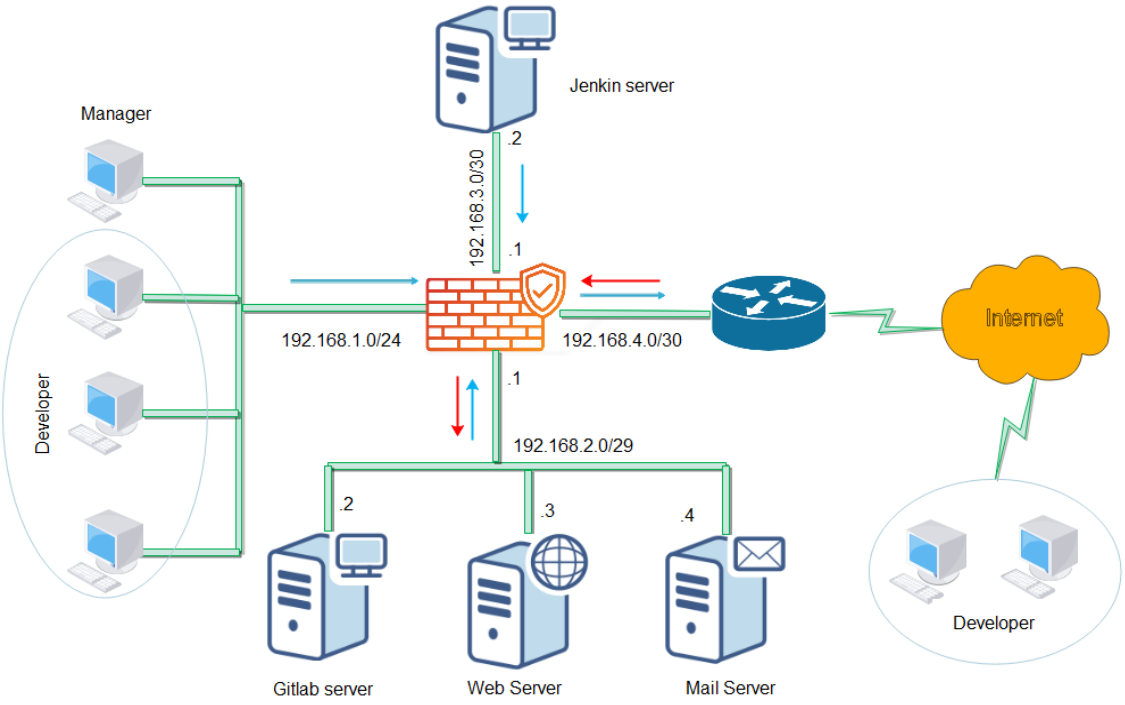

Figure 3.1: Diagram of the proposed internal continuous integration system

The network area containing the Web server and the Mail server is often referred to as the DMZ zone. Normally, any connection from the Internet to servers in the DMZ is allowed, so the Gitlab server located in the DMZ is for the purpose of allowing developers to connect and commit source code to the Server Repository even when they are outside the network. internal. Users on the Internet, including Manager and Developer, are not allowed to directly access the Jenkins server, this is why the Jenkins server is placed in a separate network area so that it is protected by both the Router and Firewall.

3.2. Configuration for the System

Configuration requirements for the proposed system include:

Set the IP address for each computer and device in the network according to the specified IP address scheme (Figure 3.1). The designated Developer's computer receives a dynamic IP address (a DHCP server inside the local network will take care of this dynamic IP address distribution).

Install Jenkins, and the necessary plugins, on the computer selected as the Jenkins server, which must have Java installed before, and then set the configuration parameters for this server. Pay attention to select the appropriate service port number (service port number) for Jenkins (Jenkins's default port number is 8080, but we can change this port after installing Jenkins).

Install Gitlab, and related plugins, on the computer selected as the Repository server, and then set the configuration parameters for this server. Remember to save the information related to the Gitlab admin account created during the installation process to change the password or login to Gitlab to perform the system administrator's functions later. The default port number of Gitlab is 80, but we can change this port after we have finished installing Gitlab. The FirewallD service on Gitlab will be turned off because we use a separate firewall, the IpTables firewall in combination with Nginx.

Connecting Jenkins server to Gitlab server: First of all, you must create an authentication key pair, so that from Jenkins server you can connect to GitLab server.

Connect the Jenkins server to the Web server: First of all, you must create an authentication key pair, to be able to create a connection from the Jenkins server to the Web server.

With the network diagram of the proposed system, it is very important to configure the Iptables firewall and set up the packet filtering rule table for it. This configuration step must be performed correctly for the system's "Network Access Policy", also known as "Network Security Policy", to take effect. At that time, the Firewall will rely on packet filtering rules to perform the function of protecting the network, protecting the servers and especially preventing attacks on the CVE-2021-44228 security vulnerability on the Jenkins server.

Table 3.1: Table of packet filtering rules on the Iptables firewall

Order	IP Source	Port Source	IP Destination	Port Destination	Allow /Deny
1	Internet-dev	Any	192.168.2.2	80, 443	Allow
2	Any	Any	192.168.2.3	80, 443	Allow
3	Any	Any	192.168.2.4	587	Allow
4	Manager’s IP	Any	192.168.3.1	8080	Allow
5	192.168.3.2	Any	192.168.2.2	22	Allow
6	192.168.1.0/24	Any	192.168.2.2	80, 443	Allow
7	192.168.1.0/24	Any	Internet	Any	Allow
8	192.168.2.4/24	Any	192.168.1.0/24	Any	Allow
9	192.168.2.0/29	Any	Internet	Any	Allow
10	Any	Any	Any	Any	Deny

The network access policy is designed as a rule table (Table 3.1) and will be configured as a filter on the Iptable firewall. The firewall will rely on this rule table to only allow connections from the Internet to servers in the DMZ to enter the network. In particular, to connect to the Gitlab server,

the traffic flow carried by the https protocol and the destination port 443 will be allowed by the Firewall. The same goes for connections that come from within the intranet.

Finally, install Nginx on the machine running the Iptables firewall, disable the Default virtual host function of Nginx, and then deploy the Reverse proxy function on it (by editing the nginx.conf configuration file). The following rules need to be set up on Nginx to require it to monitor and block traffic whose payload contains the Log4j exploit on the Jenkins server.
Here are some filter commands of the rules:

```
server {
[...]
   ## Block Log4j
   set $block_log4j 0;
   if ($query_string ~ "\$\{jndi\:") {
      set $block_sql_injections 1;
   }
   if ($query_string ~ "${jndi:ldap:") {
      set $block_sql_injections 1;
   }
                        if                    ($query_string                    ~
"\$\{jndi\:(ldap[s]?|rmi|dns|nis|iiop|corba|nds|http):/[\/]?[^\n]+") {
      set $block_sql_injections 1;
   }
[...]
   if ($block_sql_injections = 1) {
      return 403;
   }
```

The above are just some basic filtering commands, for the purpose of blocking attacks on the Log4j vulnerability, of the code that needs to be set up on Ngnix. Obviously, if the payload of traffic directed to the Jenkins server, networked by the http or https protocol, contains one of the character strings "jndi\:", "jndi:ldap:", "jndi\:(ldap). [s]?|rmi|)",… then Ngnix will return error code 403: Website is blocked.

3.3. Operation of the System

In this section, we do not discuss the operation of the proposed continuous integration system, but only the operation of the packet filtering firewall, combined with reverse proxy, in the system. How the packet filtering firewall is used to protect the internal network and the servers in the proposed system; and how the reverse proxy server prevents attacks on the Log4j vulnerability are shown here. As follows:

With the packet filtering rule table set up, in general, the firewall only allows connection streams oriented to the DMZ to pass through it. However, to reach the Gitlab server, the traffic must be carried by https protocol and the incoming port is 443. Thanks to this, developers both inside the intranet and outside the Internet can connect and commit the source code. to the Gitlab server. The firewall also only allows direct connection and data exchange between Jenkins server and GitLab server. The flow of traffic carrying messages, https protocol and port 8080, from the Jenkins server to the manager and developer computers is also allowed through the firewall.

Any traffic, exactly requests, directed to the Jenkins server, carried by the http or https protocol is controlled by the Nginx server. This server intercepts traffic streams whose payload contains content for the purpose of exploiting Log4j vulnerabilities that may exist in the plugins that Jenkins is using. Nginx proxies rely on established rule sets to do this. As such, the Jenkins server is not only protected against possible attacks on the server itself but is protected against attacks on the Log4j vulnerability that may exist in its plugins.

4. DISCUSSION ABOUT THE PROPOSED SYSTEM

The internal continuous integration system proposed and deployed in this study has ensured the basic requirements of the Intranet network of software enterprises, oriented to develop software products according to the DevOps method and apply the CI/CD process.

Because the GitLab server is located in the local network, the speed and availability of the system are quite high. This also contributes to reducing the cost of system operation and saving enterprise workspace (because Manager and Developer can work from home, connecting to the Gitlab server and/or Jenkins server via the Internet); The entire internal system, especially the Jenkins server and Gitlab server, is protected by Firewall and Reverse proxy. All access from the Internet to the internal network is controlled, only the traffic directed to the DMZ is allowed to go through the Firewall. Users on the Internet, who are not managers or developers of a certain software project of the enterprise, can hardly connect to the Gitlab server; Only two-way connections between the Jenkins server and the Gitlab server are allowed. That is, any attack against the Jenkins server, a key component of the CI system, is difficult to bypass the control and prevention of the Firewall, Reverse proxy. Manager and Developer computers can only receive messages from the Jenkins server, it cannot make a direct connection to this server.

In particular, the traffic that intentionally exploits the Log4Shell vulnerability cannot go through the firewall, it is blocked by the reverse proxy according to the filtering commands set up in the suite. This is an important task of the firewall, it is also our main contribution to this study.

However, in order to install, configure and operate this system, enterprises must have qualified personnel in the field of network administration, network security and network consolidation.

5. CONCLUSION

In this study, we have successfully proposed and installed an internal continuous integration network system with three characteristic elements: i) Both the CI server and the Repository server are located in the internal network, so the speed and system availability is always guaranteed at a high level; ii) Ensuring the security of the CI server and Repository server by the approach of using a packet filtering firewall in combination with a reverse proxy server and iii) Firewall not only performs the function of protecting the internal network, protecting the CI server, protecting the network. Repository server against external attacks but also protects Jenkins server when the security vulnerability CVE-2021-44228 exists in it has not been patched in time.

We use open-source tools Jenkins, Gitlab, IpTables and Nginx for the proposed model. Still, the features of this model are highly general so when we use similar open-source tools to develop this model, the practicality and feasibility of the model are still guaranteed.

REFERENCES

[1] Mojtaba S., Muhammad A. B. and Liming Z., "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices", IEEE Access, vol. 5, 2017, DOI: 10.1109/ACCESS.2017.2685629.

[2] Donca C., Stan O. P., Misaros M., Gota D. And Miclea L., "Method for Continuous Integration and Deployment Using a Pipeline Generator for Agile Software Projects", Sensors (Basel), vol. 22, no. 12, 2022, DOI: 10.3390/s22124637.

[3] Sree P. K., Rajkumar P., "Survey on Continuous Integration, Deployment and Delivery in Agile and DevOps Practices", International Journal of Computer Sciences and Engineering, vol. 4, iss. 4, pp. 213-216, 2016.

[4] Ioannis K. M., Imtiaz H., Claudia A., Fred H., Yann A., Luc D., Xian Z., Christopher J. W., Jeremy L. J., Nicholas H., John T., and Christian N. P., "Jenkins-CI, an Open-Source Continuous Integration System, as a Scientific Data and Image-Processing Platform", Society for Laboratory Automation and Screening, SLAS Discovery, vol. 12, pp. 1-12, 2016, DOI: 10.1177/1087057116679993.

[5] Pranoday P. D.,"CI/CD Pipeline Using Jenkins Unleashed - Solutions While Setting Up CI/CD Processes" (Book), Apress Berkeley, CA, Number 1, 2022, DOI: 10.1007/978-1-4842-7508-5.

[6] Sriniketan M., Vaibhav B., "Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible", IEEE Explore, International Conference on Emerging Trends in Information Technology and Engineering, 2020, DOI: 10.1109/ic-ETITE47903.2020.239.

[7] Rayanagoudar S.F., Hampannavar P. S., Pujari J.D., Parvati V. K., *"Enhancement of CICD Pipelines with Jenkins BlueOcean",* International Journal Of Computer Sciences And Engineering, vol. 6, iss. 6, pp.1048-1052, 2018, DOI: 10.26438/ijcse/v6i6.10481053.

[8] Behind Java, *"Log4J Vulnerability (Log4Shell) Explained - for Java developers*", Available at: https://www.incibe-cert.es/en/blog/log4shell-analysis-vulnerabilities-log4j, 2021 [Accessed 27 Feb 2023].

[9] Mohammad I., Abdulrahman A. A., Bilal A., *"Role of firewall Technology in Network Security",*International Journal of Innovations & Advancement in Computer Science, vol. 4, iss. 12, pp. 3-6, 2015.

[10] Sahithi D,, Tarik E., *"Firewalls Implementation in Computer Networks and Their Role in Network Security"*, Journal of Multidisciplinary Engineering Science and Technology, vol. 2, iss. 3, pp. 408-411, 2015.

[11] Alex Tatistcheff, *"Protecting against Log4j with Secure Firewall & Secure IPS"*,Available at: https://blogs.cisco.com/security/protecting-against-log4j-with-secure-firewall-secure-ips?scid=6WCg7b7JiR3&id=1RqLEkeW8BL, 2021 [Accessed 27 Feb 2023].

AUTHORS

Tuan Nguyen Kim (First Author) was born in 1969, received B.E., and M.E from Hue University of Sciences in 1994, and from Hanoi University of Technology in 1998. He has been a lecturer at Hue University since 1996. From 2011 to the present (2021) he is a lecturer at School of Computer Science, Duy Tan University, Da Nang, Vietnam. His main research interests include Computer Network Technology and Information.

Ha Nguyen Hoang (Corresponding Author) was born in 1976 in Vietnam. Working in the Faculty of Information Technology, Hue University of Sciences. 1995-1999 Bachelor of Science (B.S) in Science (majoring in COMPUTER SCIENCE), Hue University of Sciences. 2003-2005 Master of Science (M.S) in Computer Science, Hue University of Sciences.2012-2017 Doctor of Science (D.S) in Computer Science, Hue University of Sciences. His main research interests include Cloud Computing, parallel processing, and distributed processing.

Vy Huynh Trieu was born in 1979 and is currently a lecturer at Pham Van Dong University. He received a bachelor's degree in information technology in 2002 and a master's degree in computer science in 2006 from the Hue University of Science, Hue University. He successfully defended his doctoral thesis in computer science in April 2023 at the University of Science and Technology – The University of Da Nang. His main research interest is privacy-preserving in data mining.