EXPERIMENTAL STUDIES OF THE FEATURES OF USING WAF TO PROTECT INTERNAL SERVICES IN THE ZERO TRUST STRUCTURE

LAKHNO V. [1], **BLOZVA A.** [2], **KASATKIN D.** [3],
CHUBAIEVSKYI V. [4], **SHESTAK Y.** [5], **TYSHCHENKO D.** [6], **BRZHANOV R.** [7]

[1,2,3] National University of Life and Environmental Sciences of Ukraine, Kyiv, Ukraine,
[4,5,6] Kyiv National University of Trade and Economics, Department of Software Engineering and Cybersecurity, Kyiv, Ukraine,
[7] Caspian State University of Technology and Engineering named after Sh. Yesenov, Aktau, Kazakhstan

E-mail: [1]lva964@gmail.com, [2]andriy.blozva@nubip.edu.ua, [3]d.kasatkin@nubip.edu.ua,
[4]chubaievskyi_vi@knute.edu.ua, [5]shestak@knute.edu.ua, [6]tyshchenko@knute.edu.ua, [7]brzhanov@mail.ru

ABSTRACT

With the growth of web applications popularity, the need to protect them from hacking and unauthorized access is growing even faster. More than 75% of hacker attacks are aimed at vulnerabilities in web applications and corporate websites. The consequences of such malicious actions are quite obvious and not very pleasant for companies (especially their customers): the loss of personal data, including payment information, the ability to access trade secrets and confidential documents via enterprise networks. Traditional firewall methods do not prevent attacks on web services. Firewalls target threats at the network and transport layers, while web applications operate at the application layer. A Web Application Firewall (WAF is a type of firewall that is used to protect web applications. While a forward proxy server protects the client computer's identity using an intermediary, WAF deploys in front of web applications (in reverse proxy mode) and analyzes bi-directional HTTP / HTTPS traffic to entice malicious traffic and block it. WAFs are not the ultimate security solution, rather they are intended to be used in conjunction with other network perimeter security solutions such as next-generation firewalls (NGFW) and intrusion prevention systems (IPS).

Keywords: *Security, Firewalls, OWASP, WAF*

1. INTRODUCTION

A web application architecture is a mechanism that defines how the components of a program interact with each other. Other words, a web application architecture is a model for the interaction between the various components of a web application. Most of the web applications consist of two parts: client (front-end) and server (back-end). The server-side code (back-end) is responsible for rendering the page requested by the user, as well as for storing various data, including user-profiles and data entered by the user. This code is always hidden from the user. To write client-side code (front-end), a combination of technologies such as HTML, CSS, JavaScript is used. Client-side code specifically designed for user interaction [1].

Most web applications are developed by dividing the core functionality into 3 tiers: presentation tier, business tier, and resilience tier.

The presentation layer reflects the interface and is designed for user interaction. It is developed using three main technologies: HTML is the markup language that defines the structure of the site, CSS allows you to control the appearance of the application, and JavaScript with supporting frameworks make the site interactive.

The business layer, also called business logic, accepts requests from the user, processes them and determines the routes along which data will be accessed. For example, if the application provides hotel booking functionality, the business layer will be responsible for the sequence of actions to be taken when booking a room.

The resiliency tier, or storage tier, is a centralized location that accepts all data requests and provides access to application storage.

The storage infrastructure includes a server and a database management system, the software that communicates with the database [12-14].

Some components are parts of a web application, but separated from the main layers - end-to-end code and third-party integrations. End-to-end code handles application functionalities such as communications, operational management and security. It affects all parts of the system, but should never mix with them. Third-party integrations are integrations that are connected to the back-end of the application using senippets of code called APIs [2].

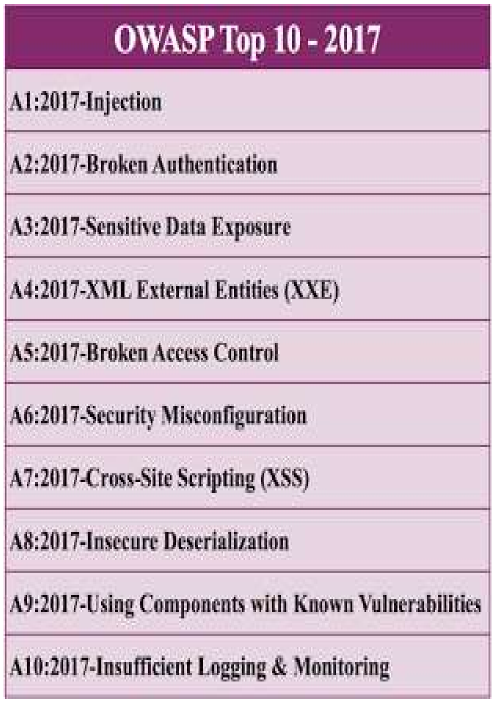
Today, there are several types of web application architecture, depending on how the interaction between the client and server-side takes place. The most common of these are single-page applications (SPA) and multi-page applications (MPA).

Single Page Applications - A type of web application that uses a single HTML page to display all of the information. In practice, it means that the user observes the main content of the page in the browser, but when scrolling or switching to another page, all the necessary elements are dynamically updated instead of reloading the page and sending a new request to the server. Examples of single-page applications include Gmail, Facebook, Twitter.

Multi-Page Applications is a web application consisting of multiple pages that are loaded every time the user visits them. Each time a new page is requested, a request is sent to the server and all data is completely refreshed. This is a traditional web application development pattern that is used on sites with a lot of content. As a rule, MPA applications have a complex structure, with a large number of levels and links. The content of such web applications is divided into several sections and subsections. An example of a multi-page application would be Amazon or eBay [3].

Today, a large number of people use web applications to find the products and services they want. Customers that provide their names, payment system data, can become a gold mine for hackers who seek to get hold of confidential information. That being said, protecting a site is also a matter of protecting physical equipment. Hackers can not only steal sensitive client information but also infect a website with malware that can affect physical hardware. Website security is critical to the longevity of a business, as unauthorized access can have a significant impact on reputation, downtime, and also result in decreased performance.

The OWASP (Open Web Application Security Project) community is responsible for classifying attack vectors and vulnerabilities. It is an international non-profit organization focused on analyzing and improving software security. OWASP has created a list of the 10 most dangerous attack vectors for Web applications, this list is called OWASP Top Ten, it contains the most dangerous vulnerabilities.



OWASP Top 10 - 2017
A1:2017-Injection
A2:2017-Broken Authentication
A3:2017-Sensitive Data Exposure
A4:2017-XML External Entities (XXE)
A5:2017-Broken Access Control
A6:2017-Security Misconfiguration
A7:2017-Cross-Site Scripting (XSS)
A8:2017-Insecure Deserialization
A9:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring

Figure 1. OWASP Top Ten

Today, the issue of web application security is very acute, since web applications are tightly integrated into the modern world. The OWASP "Top 10" is a recognized global methodology for assessing vulnerabilities, reflecting modern trends in web application security. While the global web application security policy has slowly changed in the right direction over the past few years. During 2020 and the COVID-19 pandemic, this process has stopped and, in some cases, the situation has slightly deteriorated.

2. THE AIM OF THE STUDY

In this paper, we aim to apply WAF to the internal web resources of the Free Economic Zone in systems with zero trust in users. With a possible response to the threats posed by OWASP. We strive to ensure the effectiveness of such an approach to the most common threats and increase the effectiveness of protection of end users and free services.

We set ourselves the opportunity to improve the algorithm for protecting services in the local networks of universities to protect end users and services.

3. MODELS AND METHODS

A Web Application Firewall (WAF) is a device that protects web applications from most common attacks (including OWASP Top Ten).

The WAF sits between external users and web applications and analyzes all HTTP / HTTPS traffic, identifying and blocking malicious requests before they can affect users or the web application. As a result, WAFs protect business-critical web applications and web servers from attacks.

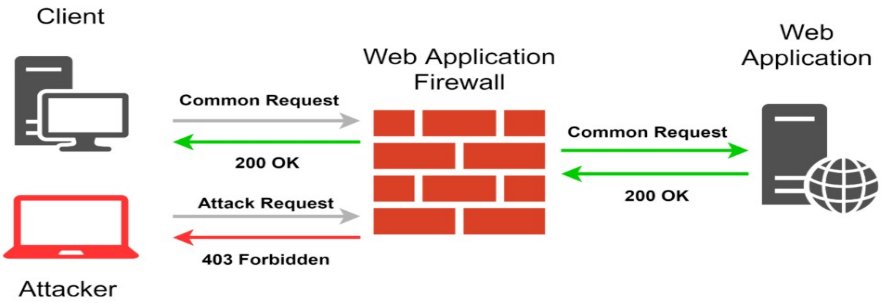


Figure 2. Scheme of WAF

Traditional network firewalls protect the local network from unauthorized access. Their main purpose is to separate the protected area from the less secure one and further control the communication between them. The key technical difference between application layer firewalls and network layer firewalls is the layer at which they operate, as defined by the Open Systems Interconnection Model, which characterizes and standardizes communication functions in telecommunications and computing systems. WAF protects against attacks at layer 7 of the OSI model - the application layer. The main threats at this level are attacks on various frameworks, cookie manipulation, SQL injection exploitation and cross-site scripting attacks. Traditional network firewalls operate at layers 3 and 4 of the OSI model to protect network traffic. For this reason, a traditional standalone network firewall will not protect businesses from attacks on web pages [9].

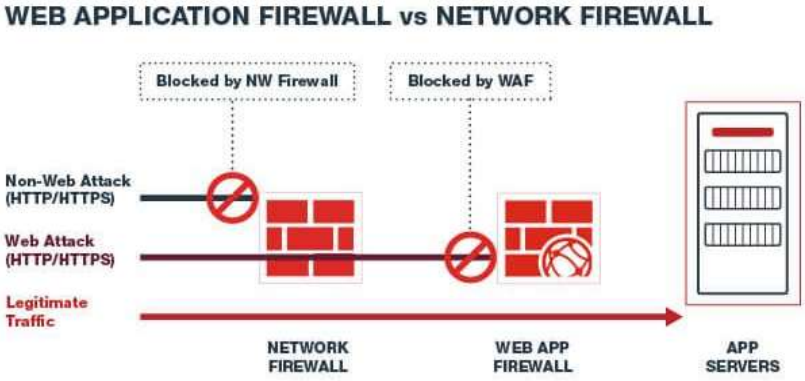


Figure 3. Comparison of the traditional firewall with web application firewall in countering web attacks

WAF operates on a set of rules called policies, which are used to filter most of the attacks known today. Many WAF services provide a default set of rules that are updated periodically.

WAFs can operate on negative security (blacklist), positive security (whitelist) or hybrid model. The blacklist model uses predefined signatures to block harmful web traffic, as well as signatures designed to prevent attacks that exploit specific vulnerabilities in websites and applications. For example, if multiple IP addresses are sending many more packets than is typical, a blacklisted firewall can prevent a DDOS attack.

The whitelist model allows web traffic to meet specially configured criteria. For example, a firewall can be configured to only allow HTTP GET requests from specific IP addresses. Whitelisted firewalls are best for intranet web applications that are intended to be used only by a limited group of people, such as company employees [11].

WAF can be implemented in one of the following ways, each with its advantages and disadvantages:

- ✓ Network WAF, usually hardware. Installing locally minimizes latency, but is a more expensive option.
- ✓ Host-oriented WAF. This solution is cheaper than network WAF and offers more customization options. The disadvantages of a host-based WAF are local server resource consumption, implementation complexity and maintenance costs.
- ✓ Cloud WAF. Provides the simplest implementation, has the lowest initial cost, and offers a solution that is constantly updated to protect against new threats without additional work or cost on the customer side. Flaw cloud WAF is a third-party liability. One of the most popular cloud web application firewalls is Cloudflare WAF.

In this project, WAF is implemented in a host-based mode on the Nano Pi R1 hardware platform (appearance, the layout of board elements are given in Appendix A, technical specifications are given in Appendix B).

There are many free WAFs that are capable of securing web applications. The biggest advantage of the open-source WAF is the freedom to modify the code to suit the needs of the projects. The most famous open-source WAFs include:

- ✓ ModSecurity. This firewall is well equipped with many features and offers complete freedom to expand its capabilities. Among the main features of this firewall are the following: application security monitoring and real-time access control, HTTP traffic logging, continuous passive security assessment. The ModSecurity community is actively and constantly releasing updates.
- ✓ NAXSI. The acronym comes from Nginx Anti XSS & SQL Injection. The main purpose of this firewall is to protect against SQL injection and cross-site scripting.
- ✓ WebKnight. Designed for Microsoft IIS. The toolkit checks all requests and filters them according to the policies set by the administrator. The firewall aims to prevent buffer overflow attacks, SQL injection, character encoding [5].

WAF can be integrated into the network in the following ways: network monitoring mode via SPAN port, bridge mode, reverse proxy).

In monitor mode, packets do not go through the web application firewall. The Switched Port Analyzer (SPAN) feature forwards a copy of traffic on a port to another port on the same switch. In this mode, the firewall analyzes the copy of the monitored traffic, rather than the packets that are being sent. The advantage of operating in this mode is that WAF does not affect traffic, avoiding performance and latency issues. The disadvantage of working in this mode is that WAF works with a copy of the traffic and cannot prevent attacks on web applications.

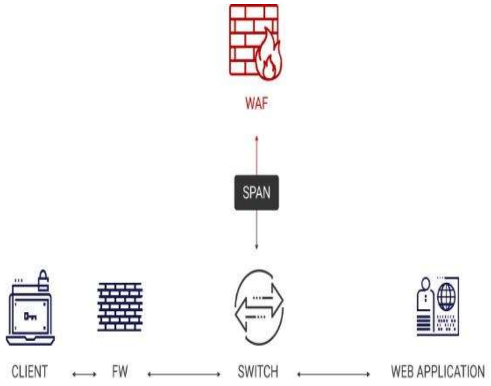


Figure 4. An example of WAF implementation in monitoring mode

In bridge mode, WAF sits on the same line between the firewall and web servers and acts as a Layer 2 bridge.

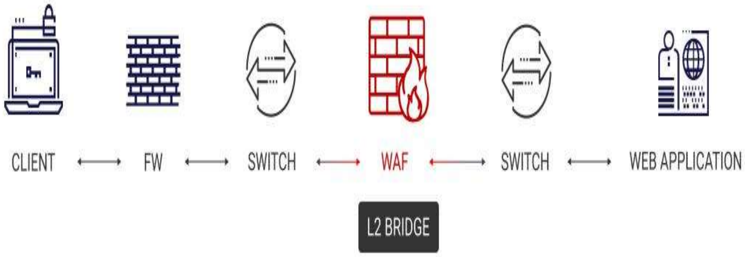


Figure 5. An example of WAF implementation in monitoring mode

Reverse proxy server. Typically, proxy servers act as intermediaries for online connections. Proxies can be divided into types according to various criteria. The type of proxy depends on the type of device acting as a proxy server, the level of anonymity of the client when using the proxy, and the method of data management. According to another criterion - location in the network structure - the proxy server is divided into reverse and forward.

Direct proxy server - when using the term "proxy", most often they mean a direct proxy server. Forward proxies are types of proxies that clients use to hide their IP addresses and maintain anonymity when browsing the Internet. When using a forward proxy server, the device sends a normal request that the proxy server does not exist, but all requests to the target system will go through the proxy server. The proxy accepts requests and redirects them through its IP address, hiding the user's real IP address. Most often, direct proxy servers are used by ordinary users to bypass blocked services.

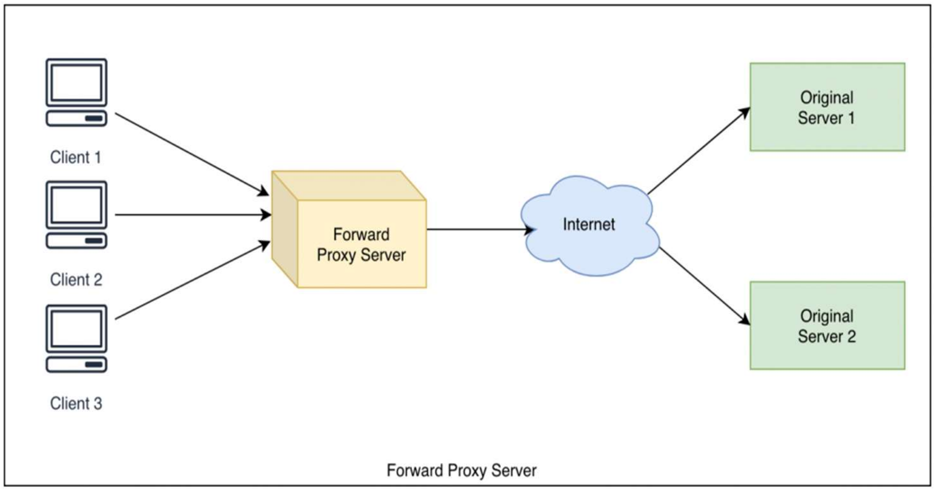


Figure 6. An example of a direct proxy server

A reverse proxy is a proxy that accepts requests on behalf of web servers. A reverse proxy does not work for clients, but web servers. Whereas a forward proxy is designed to provide anonymity to clients, a reverse proxy is designed to provide anonymity for web servers. They hide the reallocation of the servers from clients.

The reverse proxy accepts requests from the Internet and determines whether to forward the request to a real server [7].

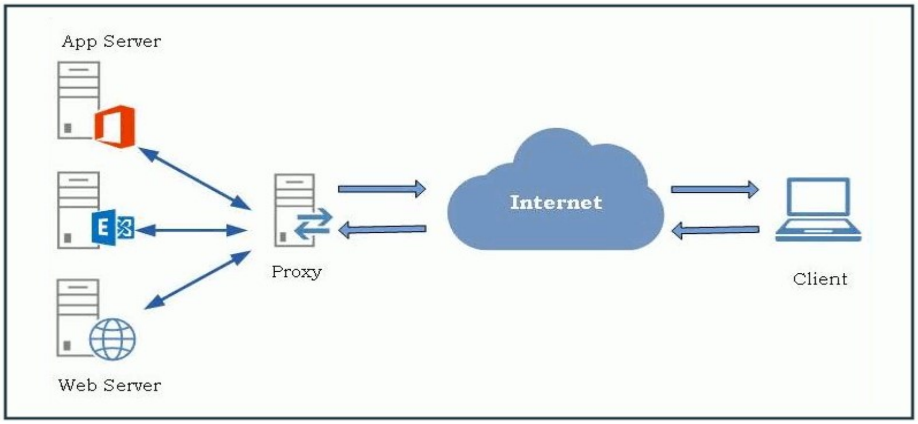


Figure 7. An example of a reverse proxy server

Reverse proxies can be used to:

- ✓ load balancing. Typically, sites with a lot of daily users cannot handle all traffic with a single egress server. In this way, the reverse proxy can evenly distribute the load among the back-end servers;
- ✓ additional security of internal servers. If the web uses a reverse proxy, its address is hidden, and users can only access the IP address of the reverse proxy. This introduces an additional element of security. For example, it is much more difficult to conduct a denial of service attack;
- ✓ caching. It is the process of keeping a copy of files in the cache for faster re-access. Caching allows sites to efficiently reuse previously acquired data. This allows web applications to run more efficiently;
- ✓ SSL encryption. Encrypting and decrypting connections for each user can be ineffective for the egress server. A reverse proxy server can do this job by encrypting and decrypting all requests [8-11].

Implementing a firewall as a reverse proxy server is by far the most popular and widely used.

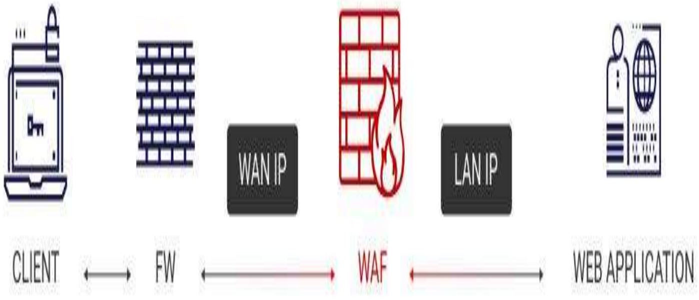
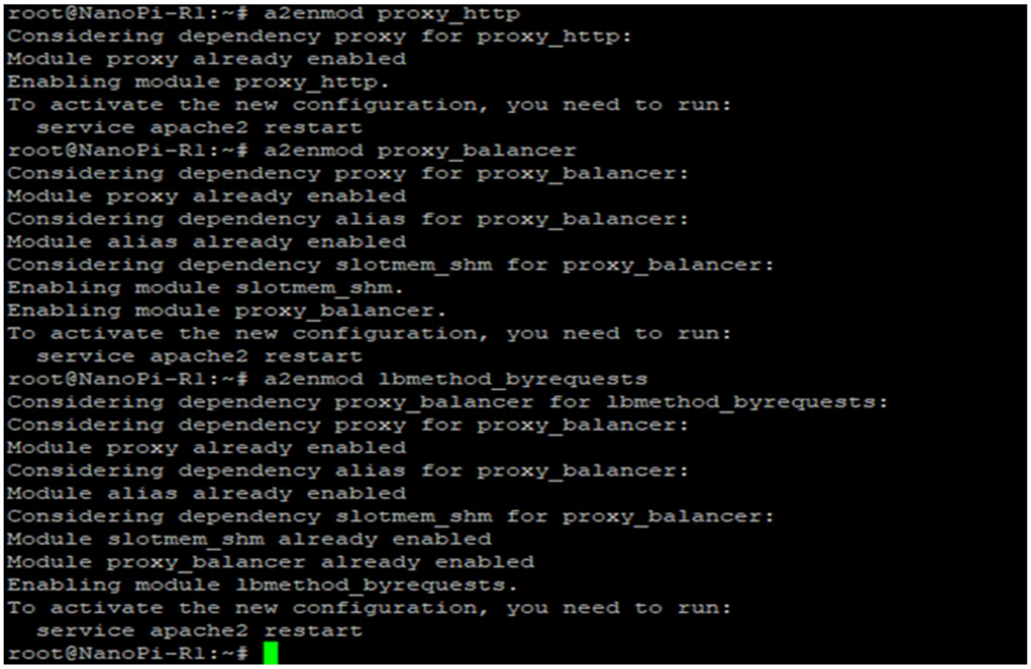


Figure 8. Implementing WAF as a reverse proxy server

A firewall is a device that secures networks by monitoring network traffic based on established sets of security rules. A Web Application Firewall (WAF) is a device that protects web applications from most attacks today (including OWASP Top Ten). The WAF sits between external users and web applications and analyzes all HTTP / HTTPS traffic, identifying and blocking malicious requests before they can affect users or the web application. Traditional network firewalls operate at layers 3 and 4 of the OSI model to protect network traffic. For this reason, a traditional network firewall alone will not protect businesses from attacks on web pages. WAF protects against attacks at layer 7 of the OSI model - the application layer. The main threats at this level are attacks on various kinds of frameworks, cookie manipulation, exploitation of SQL injection, cross-site scripting attacks as a result of WAF protecting business-critical web applications and web servers from attacks. WAF operates on a set of rules called policies, which are used to filter most of the attacks known today. Many WAF services provide a default set of rules that are updated periodically. The most commonly used method for implementing WAF on the web is as a reverse proxy.

The firewall function was assigned to the ModSecurity program. WAF budo is deployed in reverse proxy mode. To do this, you need to configure the Apache webserver in reverse proxy mode. We enable additional modules that are required for the Apache webserver to function as a reverse proxy server (mod_proxy is the main Apache proxy module that manages and redirects connections, mod_proxy_http is the proxy server functions for HTTP and HTTPS protocols, mod_proxy_balancer and mod_lbmethod_byrequests)



```
root@NanoPi-R1:~# a2enmod proxy_http
Considering dependency proxy for proxy_http:
Module proxy already enabled
Enabling module proxy_http.
To activate the new configuration, you need to run:
  service apache2 restart
root@NanoPi-R1:~# a2enmod proxy_balancer
Considering dependency proxy for proxy_balancer:
Module proxy already enabled
Considering dependency alias for proxy_balancer:
Module alias already enabled
Considering dependency slotmem_shm for proxy_balancer:
Enabling module slotmem_shm.
Enabling module proxy_balancer.
To activate the new configuration, you need to run:
  service apache2 restart
root@NanoPi-R1:~# a2enmod lbmethod_byrequests
Considering dependency proxy_balancer for lbmethod_byrequests:
Considering dependency proxy for proxy_balancer:
Module proxy already enabled
Considering dependency alias for proxy_balancer:
Module alias already enabled
Considering dependency slotmem_shm for proxy_balancer:
Module slotmem_shm already enabled
Module proxy_balancer already enabled
Enabling module lbmethod_byrequests.
To activate the new configuration, you need to run:
  service apache2 restart
root@NanoPi-R1:~#
```

Figure 9. Connecting additional Apache modules

4. EXPERIMENTAL STUDIES

Let's edit the default configuration file 000-default.conf to enable the proxy function.

During the experiments, they were limited by the small amount of statistical data accumulated for a relatively small segment of the institute's network - the university.

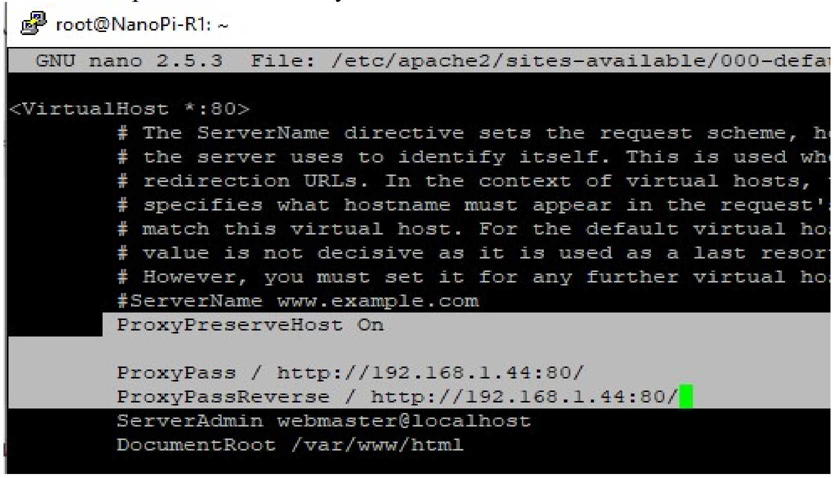
In this experimental part of the work, we set ourselves the goal of applying WAF to the internal web resources of the university in a system with zero trust to users. With the possible response to the threats that OWASP relies on, we strive to make sure that this approach is effectively applied to the most common threats and increases the effectiveness of protecting end users and university services.

Three directives are used to configure the proxy:

- ✓ ProxyPreserveHost forces Apache to pass the output Host header to the backend server. This is useful because it allows the backend server to know the address used to access the application.
- ✓ ProxyPass is the main directive for proxy configuration. In this case, it specifies that everything after the root URL (/) should be sent to the server at the given address.
- ✓ ProxyPassReverse - must-have settings similar to ProxyPass. It tells Apache how to change the headers in the response from the backend server. Thus, it is guaranteed that the client's browser will be redirected to the proxy address and not to the backend server address.

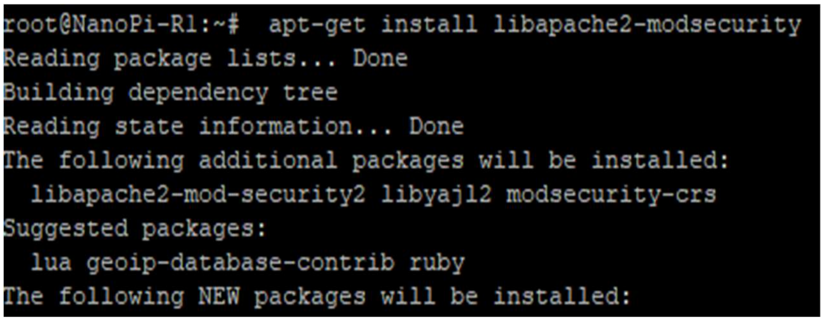
As a result of the proxy settings, when accessing the address http://192.168.1.251/, a page will be opened located on the server with the address 192.168.1.44/.

After you finish configuring the Apache web server, you need to install ModSecurity using the command apt-get install libapache2-modsecurity.



```
root@NanoPi-R1: ~
  GNU nano 2.5.3  File: /etc/apache2/sites-available/000-defa

<VirtualHost *:80>
        # The ServerName directive sets the request scheme, h
        # the server uses to identify itself. This is used wh
        # redirection URLs. In the context of virtual hosts, 
        # specifies what hostname must appear in the request'
        # match this virtual host. For the default virtual ho
        # value is not decisive as it is used as a last resor
        # However, you must set it for any further virtual ho
        #ServerName www.example.com
        ProxyPreserveHost On

        ProxyPass / http://192.168.1.44:80/
        ProxyPassReverse / http://192.168.1.44:80/
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
```

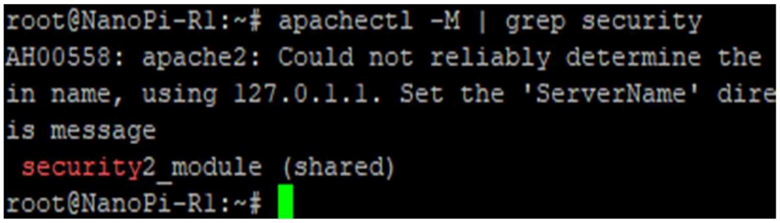
Figure 10. Editing the 000-default.conf file



```
root@NanoPi-R1:~#  apt-get install libapache2-modsecurity
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libapache2-mod-security2 libyajl2 modsecurity-crs
Suggested packages:
  lua geoip-database-contrib ruby
The following NEW packages will be installed:
```

Figure 11. Installing ModSecurity

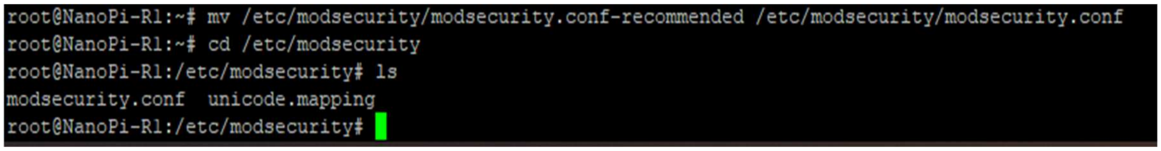
To check if the installation is correct, use the command apachectl -M | grep security. If the installation was successful, the command should output security2_module (shared).



```
root@NanoPi-R1:~# apachectl -M | grep security
AH00558: apache2: Could not reliably determine the
in name, using 127.0.1.1. Set the 'ServerName' dire
is message
 security2_module (shared)
root@NanoPi-R1:~# 
```

Figure 12. Checking the correctness of the ModSecurity installation

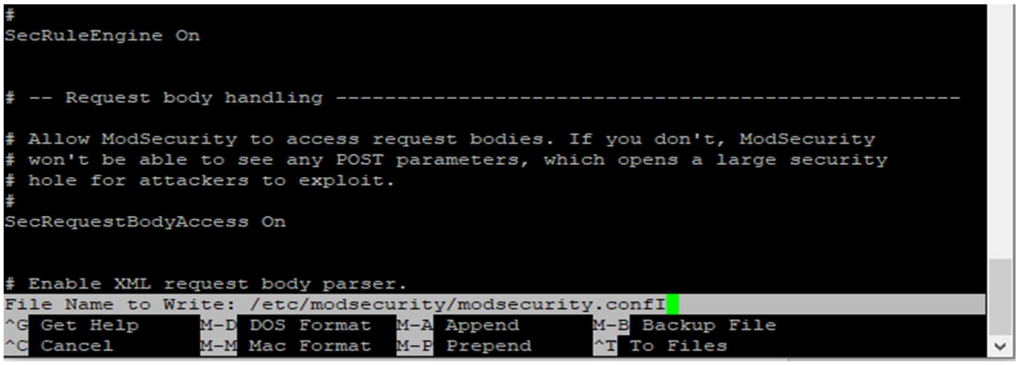
ModSecurity includes a recommended modsecurity.conf-recommended configuration file located in the / etc / modsecurity directory. In order for this file to work with ModSecurity, you must rename it using the mv /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf command.



```
root@NanoPi-R1:~# mv /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf
root@NanoPi-R1:~# cd /etc/modsecurity
root@NanoPi-R1:/etc/modsecurity# ls
modsecurity.conf  unicode.mapping
root@NanoPi-R1:/etc/modsecurity#
```

Figure 13. Renaming the configuration file

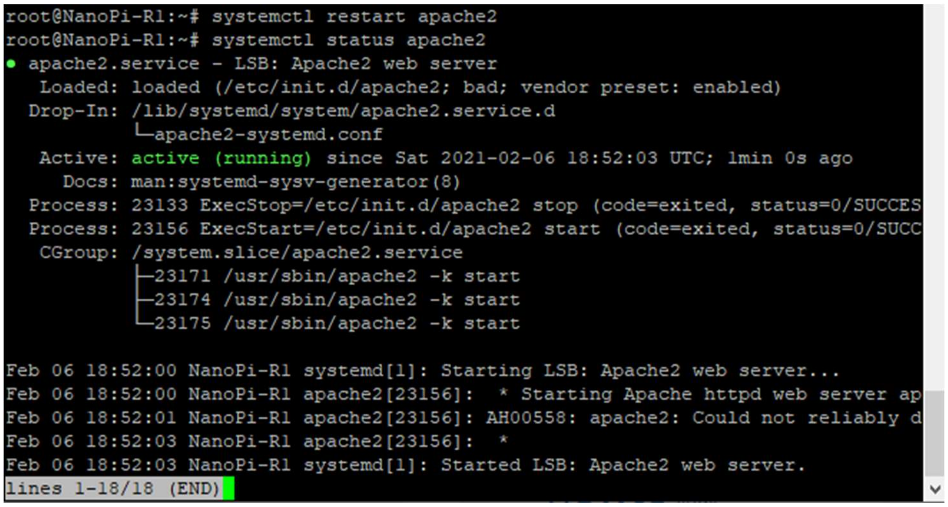
Using any text editor, edit the contents of the modsecurity.conf file. Change "SecRuleEngine Detection Only" to "SecRuleEngine On", save changes and exit the text editor.



```
#
SecRuleEngine On


# -- Request body handling ---------------------------------------------------

# Allow ModSecurity to access request bodies. If you don't, ModSecurity
# won't be able to see any POST parameters, which opens a large security
# hole for attackers to exploit.
#
SecRequestBodyAccess On


# Enable XML request body parser.
File Name to Write: /etc/modsecurity/modsecurity.confI
^G Get Help     M-D DOS Format   M-A Append      M-B Backup File
^C Cancel       M-M Mac Format   M-P Prepend     ^T To Files
```

Figure 14. Editing the modsecurity.conf file

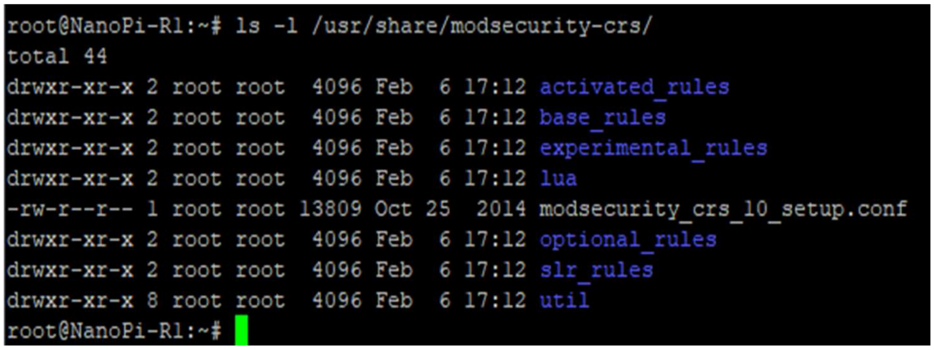
After editing the file, restart the Apache webserver.



```
root@NanoPi-R1:~# systemctl restart apache2
root@NanoPi-R1:~# systemctl status apache2
● apache2.service - LSB: Apache2 web server
   Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since Sat 2021-02-06 18:52:03 UTC; 1min 0s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 23133 ExecStop=/etc/init.d/apache2 stop (code=exited, status=0/SUCCES
  Process: 23156 ExecStart=/etc/init.d/apache2 start (code=exited, status=0/SUCC
   CGroup: /system.slice/apache2.service
           ├─23171 /usr/sbin/apache2 -k start
           ├─23174 /usr/sbin/apache2 -k start
           └─23175 /usr/sbin/apache2 -k start

Feb 06 18:52:00 NanoPi-R1 systemd[1]: Starting LSB: Apache2 web server...
Feb 06 18:52:00 NanoPi-R1 apache2[23156]:  * Starting Apache httpd web server ap
Feb 06 18:52:01 NanoPi-R1 apache2[23156]: AH00558: apache2: Could not reliably d
Feb 06 18:52:03 NanoPi-R1 apache2[23156]:  *
Feb 06 18:52:03 NanoPi-R1 systemd[1]: Started LSB: Apache2 web server.
lines 1-18/18 (END)
```

Figure 15. Rebooting the webserver

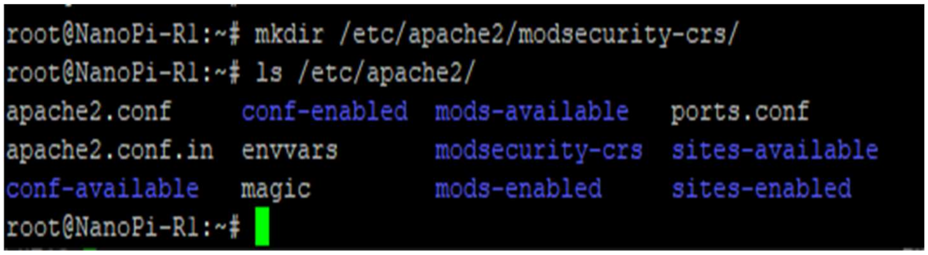
ModSecurity comes with many Core Rule Set. CRS aims to protect meb applications from a wide range of attacks (including OWASP Top Ten), with a minimum of false positives. CRS rules are stored in the / usr / share / modsecurity-crs directory



```
root@NanoPi-R1:~# ls -l /usr/share/modsecurity-crs/
total 44
drwxr-xr-x 2 root root  4096 Feb  6 17:12 activated_rules
drwxr-xr-x 2 root root  4096 Feb  6 17:12 base_rules
drwxr-xr-x 2 root root  4096 Feb  6 17:12 experimental_rules
drwxr-xr-x 2 root root  4096 Feb  6 17:12 lua
-rw-r--r-- 1 root root 13809 Oct 25  2014 modsecurity_crs_10_setup.conf
drwxr-xr-x 2 root root  4096 Feb  6 17:12 optional_rules
drwxr-xr-x 2 root root  4096 Feb  6 17:12 slr_rules
drwxr-xr-x 8 root root  4096 Feb  6 17:12 util
root@NanoPi-R1:~#
```

Figure 16. Set of basic CRS rules

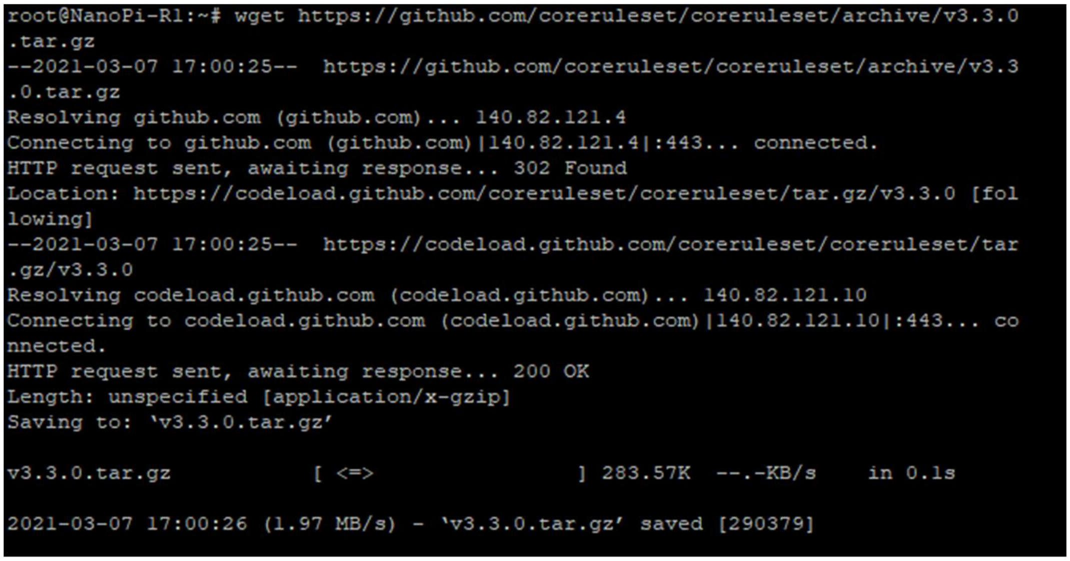
For further work, the set of rules downloaded from Github will be used. Remove the default ruleset with the rm -rf / usr / share / modsecurity-crs command. Create a new directory in the Apache directory using the command:



```
root@NanoPi-R1:~# mkdir /etc/apache2/modsecurity-crs/
root@NanoPi-R1:~# ls /etc/apache2/
apache2.conf     conf-enabled  mods-available   ports.conf
apache2.conf.in  envvars       modsecurity-crs  sites-available
conf-available   magic         mods-enabled     sites-enabled
root@NanoPi-R1:~#
```

Figure 17. Creating the modsecurity-crs directory

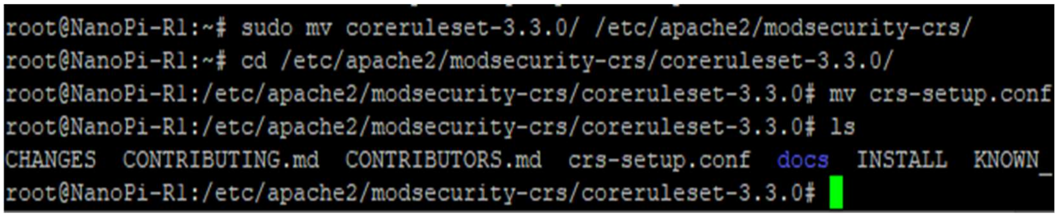
Download the basic Modsecurity ruleset using Github and unpack it with tar xvf v3.3.0.tar.gz



```
root@NanoPi-R1:~# wget https://github.com/coreruleset/coreruleset/archive/v3.3.0
.tar.gz
--2021-03-07 17:00:25--  https://github.com/coreruleset/coreruleset/archive/v3.3
.0.tar.gz
Resolving github.com (github.com)... 140.82.121.4
Connecting to github.com (github.com)|140.82.121.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/coreruleset/coreruleset/tar.gz/v3.3.0 [fol
lowing]
--2021-03-07 17:00:25--  https://codeload.github.com/coreruleset/coreruleset/tar
.gz/v3.3.0
Resolving codeload.github.com (codeload.github.com)... 140.82.121.10
Connecting to codeload.github.com (codeload.github.com)|140.82.121.10|:443... co
nnected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-gzip]
Saving to: ‘v3.3.0.tar.gz’

v3.3.0.tar.gz            [ <=>                 ] 283.57K  --.-KB/s    in 0.1s

2021-03-07 17:00:26 (1.97 MB/s) - ‘v3.3.0.tar.gz’ saved [290379]
```

Figure 18. Downloading the core Github ruleset

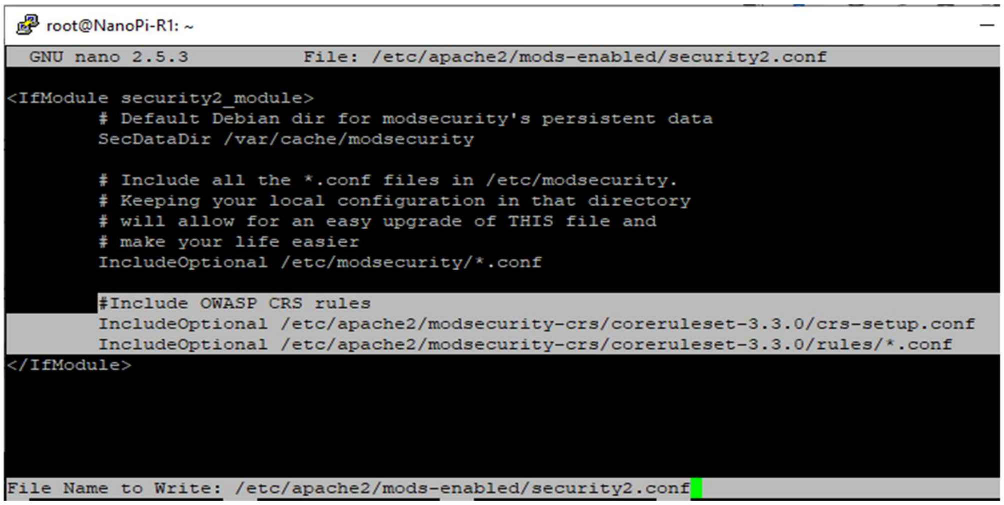
Move the unpacked directory to / etc / apache2 / modsecurity-crs /. Go to the directory again and change the name of the crs-setup.conf.example file.



```
root@NanoPi-R1:~# sudo mv coreruleset-3.3.0/ /etc/apache2/modsecurity-crs/
root@NanoPi-R1:~# cd /etc/apache2/modsecurity-crs/coreruleset-3.3.0/
root@NanoPi-R1:/etc/apache2/modsecurity-crs/coreruleset-3.3.0# mv crs-setup.conf
root@NanoPi-R1:/etc/apache2/modsecurity-crs/coreruleset-3.3.0# ls
CHANGES  CONTRIBUTING.md  CONTRIBUTORS.md  crs-setup.conf  docs  INSTALL  KNOWN_
root@NanoPi-R1:/etc/apache2/modsecurity-crs/coreruleset-3.3.0#
```

Figure 19. Setting up directories for storing Core Rule Set

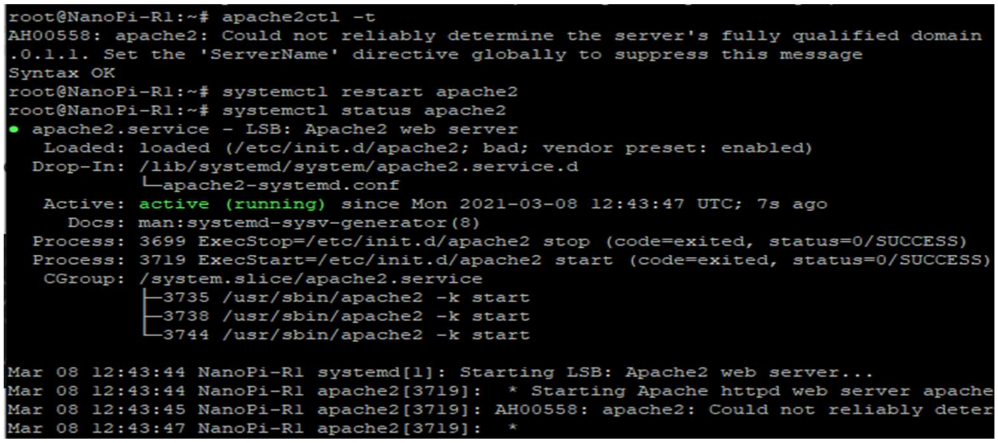
For Apache to read the .conf files in directories, you need to edit the security2.conf file.



```
root@NanoPi-R1: ~
  GNU nano 2.5.3            File: /etc/apache2/mods-enabled/security2.conf

<IfModule security2_module>
        # Default Debian dir for modsecurity's persistent data
        SecDataDir /var/cache/modsecurity

        # Include all the *.conf files in /etc/modsecurity.
        # Keeping your local configuration in that directory
        # will allow for an easy upgrade of THIS file and
        # make your life easier
        IncludeOptional /etc/modsecurity/*.conf

        #Include OWASP CRS rules
        IncludeOptional /etc/apache2/modsecurity-crs/coreruleset-3.3.0/crs-setup.conf
        IncludeOptional /etc/apache2/modsecurity-crs/coreruleset-3.3.0/rules/*.conf
</IfModule>



File Name to Write: /etc/apache2/mods-enabled/security2.conf
```

Figure 20. Editing the security2.conf file

We check the Apache configuration and restart the webserver.

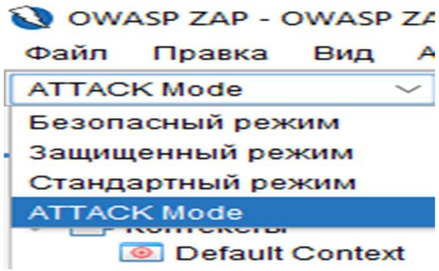


```
root@NanoPi-R1:~# apache2ctl -t
AH00558: apache2: Could not reliably determine the server's fully qualified domain
.0.1.1. Set the 'ServerName' directive globally to suppress this message
Syntax OK
root@NanoPi-R1:~# systemctl restart apache2
root@NanoPi-R1:~# systemctl status apache2
● apache2.service - LSB: Apache2 web server
   Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since Mon 2021-03-08 12:43:47 UTC; 7s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 3699 ExecStop=/etc/init.d/apache2 stop (code=exited, status=0/SUCCESS)
  Process: 3719 ExecStart=/etc/init.d/apache2 start (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/apache2.service
           ├─3735 /usr/sbin/apache2 -k start
           ├─3738 /usr/sbin/apache2 -k start
           └─3744 /usr/sbin/apache2 -k start

Mar 08 12:43:44 NanoPi-R1 systemd[1]: Starting LSB: Apache2 web server...
Mar 08 12:43:44 NanoPi-R1 apache2[3719]:  * Starting Apache httpd web server apache
Mar 08 12:43:45 NanoPi-R1 apache2[3719]: AH00558: apache2: Could not reliably deter
Mar 08 12:43:47 NanoPi-R1 apache2[3719]:  *
```

Figure 21. Checking the configuration and restarting the Apache server

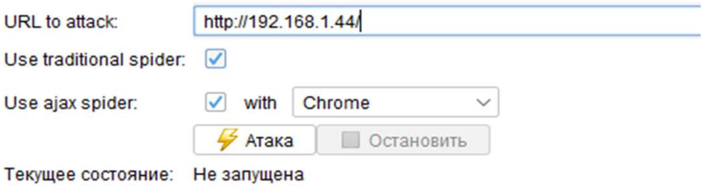
WAMP server with installed WordPress and configured standard page was chosen as a test server. For access, the IP address is 192.168.1.44 without WAF, and 192.168.1.251 - through WAF.

1. OWASP ZAP



OWASP ZAP - OWASP ZA
Файл Правка Вид А
ATTACK Mode
Безопасный режим
Защищенный режим
Стандартный режим
ATTACK Mode
Default Context

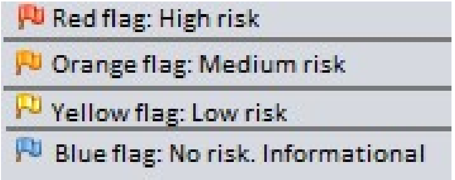
Figure 22. Selecting the scan mode



URL to attack: http://192.168.1.44/
Use traditional spider: ☑
Use ajax spider: ☑ with Chrome
Атака Остановить
Текущее состояние: Не запущена

Figure 23. URL for scan

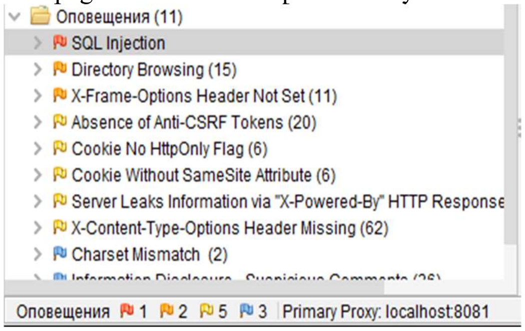
Once the scan is complete, you can view the results on the Notifications tab. Alerts are represented by 5 types of alerts, the severity of which is indicated by a specific color of the checkbox.



⚑ Red flag: High risk
⚑ Orange flag: Medium risk
⚑ Yellow flag: Low risk
⚑ Blue flag: No risk. Informational

Figure 24. Alert severity

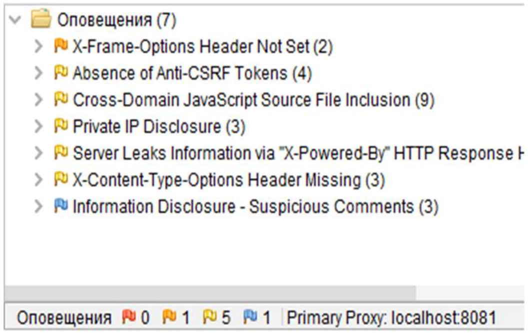
Web page scan results not protected by WAF:



Оповещения (11)
- ⚑ SQL Injection
- ⚑ Directory Browsing (15)
- ⚑ X-Frame-Options Header Not Set (11)
- ⚑ Absence of Anti-CSRF Tokens (20)
- ⚑ Cookie No HttpOnly Flag (6)
- ⚑ Cookie Without SameSite Attribute (6)
- ⚑ Server Leaks Information via "X-Powered-By" HTTP Response
- ⚑ X-Content-Type-Options Header Missing (62)
- ⚑ Charset Mismatch (2)
- ⚑ Information Disclosure - Suspicious Comments (26)

Оповещения ⚑ 1 ⚑ 2 ⚑ 5 ⚑ 3 Primary Proxy: localhost:8081

Figure 25. Testing a web page that is not protected by WAF

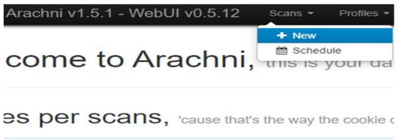


Оповещения (7)
- ⚑ X-Frame-Options Header Not Set (2)
- ⚑ Absence of Anti-CSRF Tokens (4)
- ⚑ Cross-Domain JavaScript Source File Inclusion (9)
- ⚑ Private IP Disclosure (3)
- ⚑ Server Leaks Information via "X-Powered-By" HTTP Response H
- ⚑ X-Content-Type-Options Header Missing (3)
- ⚑ Information Disclosure - Suspicious Comments (3)

Оповещения ⚑ 0 ⚑ 1 ⚑ 5 ⚑ 1 Primary Proxy: localhost:8081

Figure 26. The result of testing a page that is protected by WAF

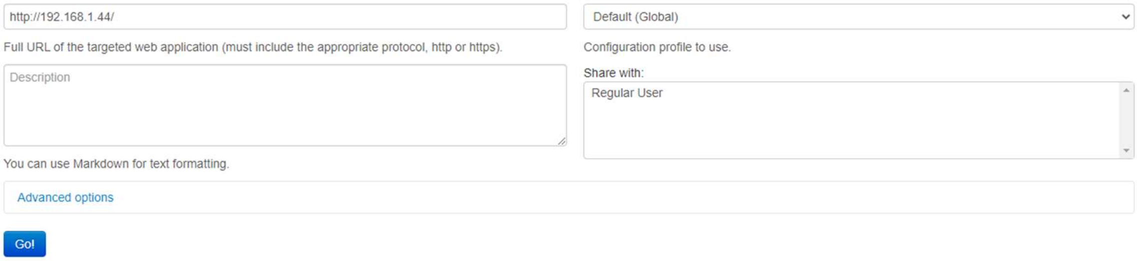
2. ARACHNI



Arachni v1.5.1 - WebUI v0.5.12 Scans Profiles
+ New
Schedule
come to Arachni, this is your da
es per scans, 'cause that's the way the cookie c

Figure 27. New scan

In the next window, select the Default scan type, set the URL for scanning and click "Go".



http://192.168.1.44/

Full URL of the targeted web application (must include the appropriate protocol, http or https).

Description

You can use Markdown for text formatting.

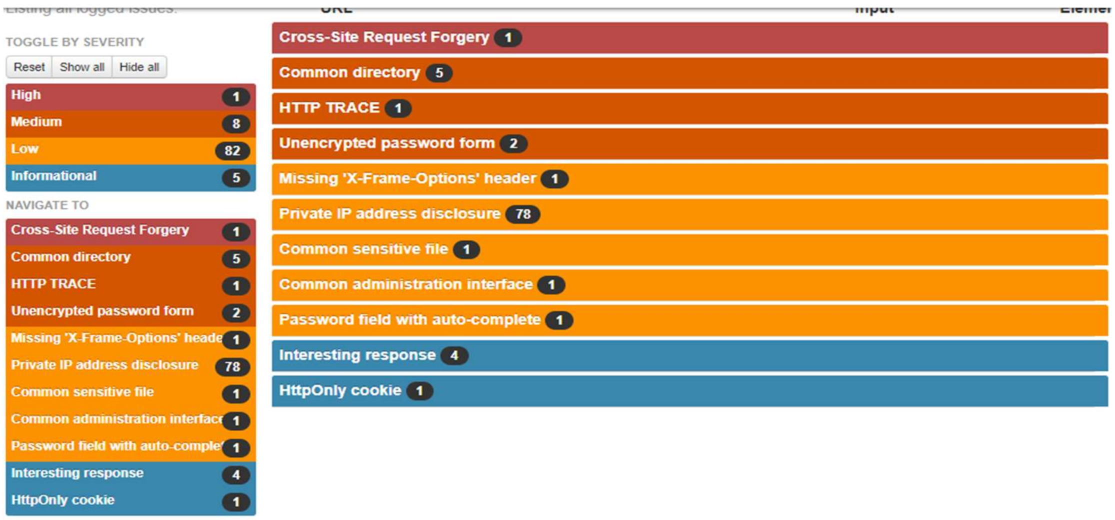
Default (Global)

Configuration profile to use.

Share with:

Regular User

Results of a 4-hour scan without WAF:



Listing all logged issues. URL Input Elemer

TOGGLE BY SEVERITY

High	1
Medium	8
Low	82
Informational	5

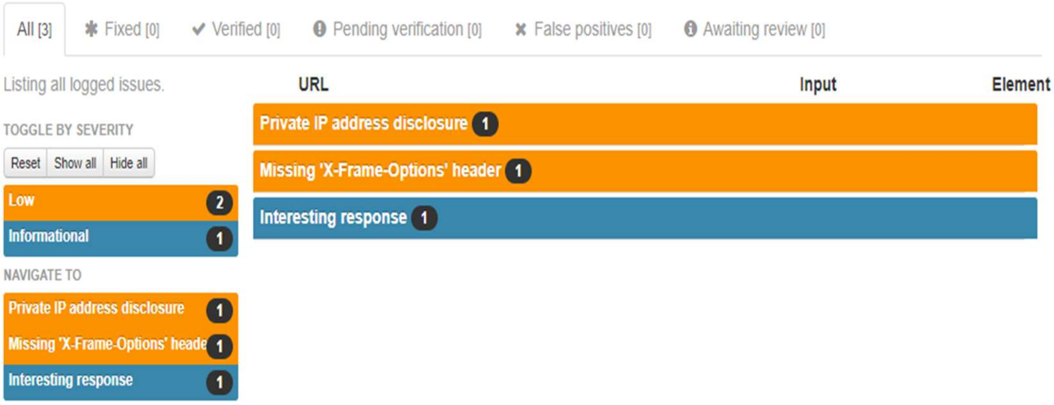
NAVIGATE TO

Cross-Site Request Forgery	1
Common directory	5
HTTP TRACE	1
Unencrypted password form	2
Missing 'X-Frame-Options' heade	1
Private IP address disclosure	78
Common sensitive file	1
Common administration interfac	1
Password field with auto-comple	1
Interesting response	4
HttpOnly cookie	1

Cross-Site Request Forgery 1
Common directory 5
HTTP TRACE 1
Unencrypted password form 2
Missing 'X-Frame-Options' header 1
Private IP address disclosure 78
Common sensitive file 1
Common administration interface 1
Password field with auto-complete 1
Interesting response 4
HttpOnly cookie 1

Figure 28. Scan without WAF

No vulnerabilities were found during scanning by XSS and SQL profiles.
Scanning via WAF lasted only 2 minutes, results:



Listing all logged issues. URL Input Element

TOGGLE BY SEVERITY

Low	2
Informational	1

NAVIGATE TO

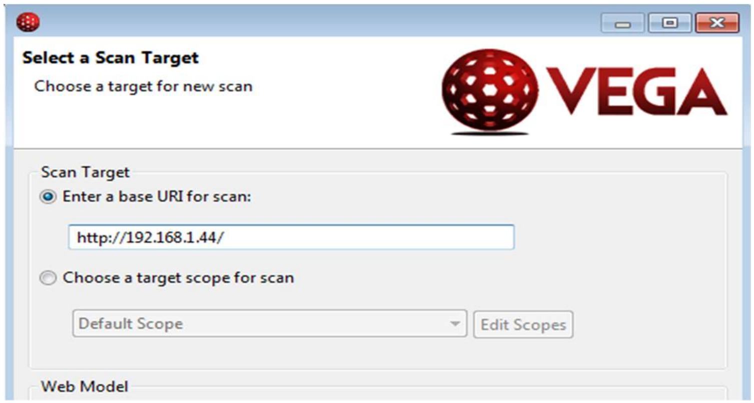
Private IP address disclosure	1
Missing 'X-Frame-Options' heade	1
Interesting response	1

Private IP address disclosure 1
Missing 'X-Frame-Options' header 1
Interesting response 1

Figure 29. Scanning via WAF

3. VEGA

Select the "Start New Scan" option. In the window that will open, enter the URL for scanning and click Finish.



Select a Scan Target

Choose a target for new scan

Scan Target

● Enter a base URI for scan:

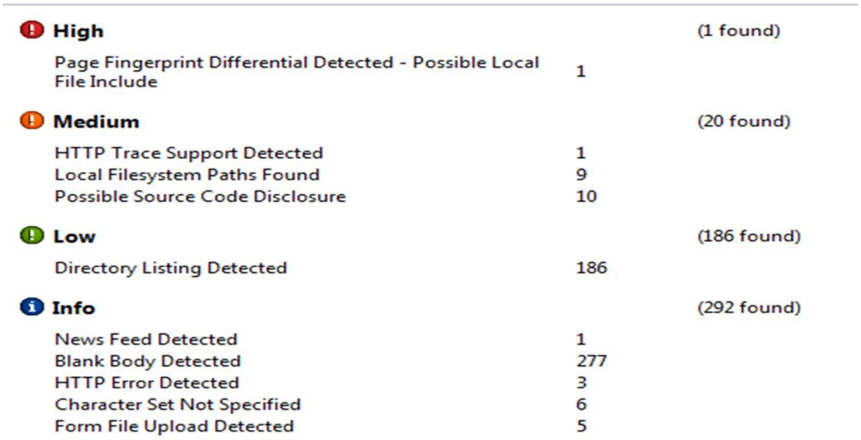
http://192.168.1.44/

○ Choose a target scope for scan

Default Scope | Edit Scopes

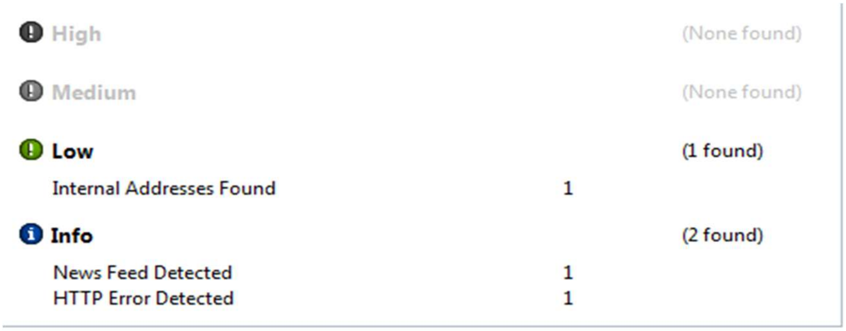
Web Model

Figure 30. URL to scan



High		(1 found)
Page Fingerprint Differential Detected - Possible Local File Include	1	
Medium		(20 found)
HTTP Trace Support Detected	1	
Local Filesystem Paths Found	9	
Possible Source Code Disclosure	10	
Low		(186 found)
Directory Listing Detected	186	
Info		(292 found)
News Feed Detected	1	
Blank Body Detected	277	
HTTP Error Detected	3	
Character Set Not Specified	6	
Form File Upload Detected	5	

Figure 31. Scanning without WAF



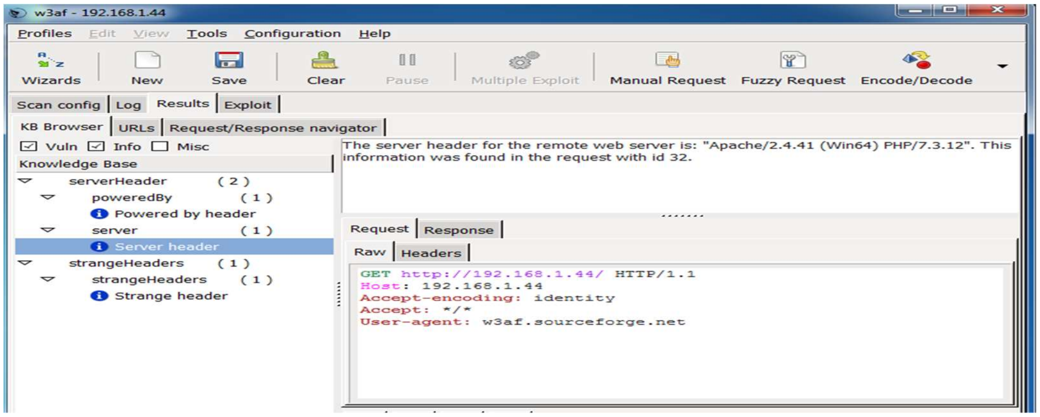
High		(None found)
Medium		(None found)
Low		(1 found)
Internal Addresses Found	1	
Info		(2 found)
News Feed Detected	1	
HTTP Error Detected	1	

Figure 32. Scanning via WAF

4. W3AF

Select the OWASP Top 10 scan profile, enter the URL and click the Start Scan button

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

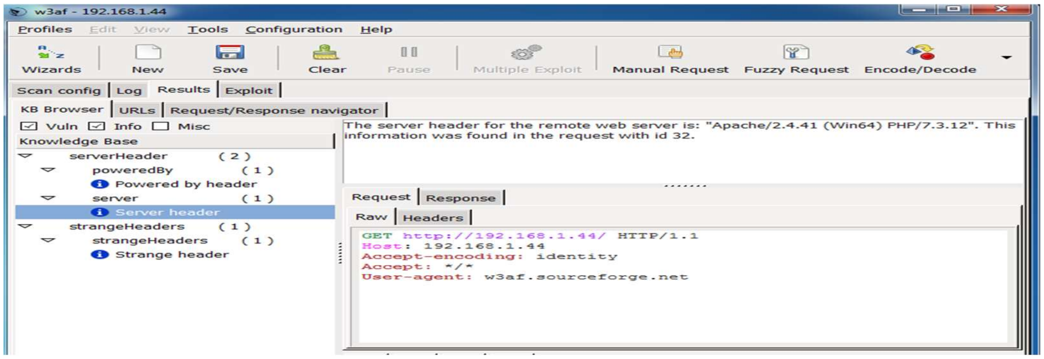


The server header for the remote web server is: "Apache/2.4.41 (Win64) PHP/7.3.12". This information was found in the request with id 32.

```
GET http://192.168.1.44/ HTTP/1.1
Host: 192.168.1.44
Accept-encoding: identity
Accept: */*
User-agent: w3af.sourceforge.net
```

Figure 33. Adjusting the scanner settings

The scan ended at 1:30 after the start. The scanner could only find information about the version of the operating system, Apache and PHP.

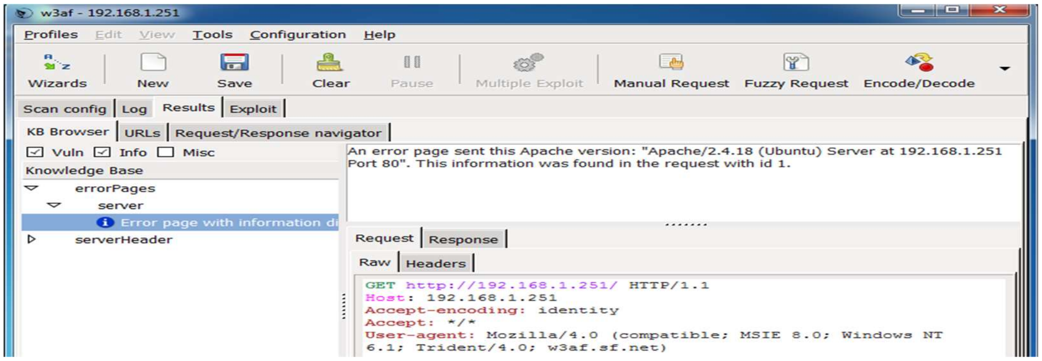


The server header for the remote web server is: "Apache/2.4.41 (Win64) PHP/7.3.12". This information was found in the request with id 32.

```
GET http://192.168.1.44/ HTTP/1.1
Host: 192.168.1.44
Accept-encoding: identity
Accept: */*
User-agent: w3af.sourceforge.net
```

Figure 34. Scan results without WAF

WAF scans ended in 1:17. As a result, the scanner was able to find one page with an Apache webserver error.


w3af - 192.168.1.251
Profiles Edit View Tools Configuration Help
Wizards New Save Clear Pause Multiple Exploit Manual Request Fuzzy Request Encode/Decode
Scan config Log Results Exploit
KB Browser URLs Request/Response navigator
☑ Vuln ☑ Info □ Misc
Knowledge Base
errorPages
server
Error page with information di
serverHeader
Request Response
Raw Headers

An error page sent this Apache version: "Apache/2.4.18 (Ubuntu) Server at 192.168.1.251 Port 80". This information was found in the request with id 1.

```
GET http://192.168.1.251/ HTTP/1.1
Host: 192.168.1.251
Accept-encoding: identity
Accept: */*
User-agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT
6.1; Trident/4.0; w3af.sf.net)
```

Figure 35. Scanning via WAF

A Web Application Firewall (WAF) is a device that protects web applications from most of today's attacks (including OWASP Top Ten). The WAF sits between external users and web applications and analyzes all HTTP / HTTPS traffic, identifying and blocking malicious requests before they can affect users or the web application. As a result, WAFs protect business-critical web applications and web servers from attacks. WAF operates on a set of rules called policies, which are used to filter most of the attacks known today. Many WAF services provide a default set of rules that are updated periodically. The most commonly used method for implementing WAF on the web is as a reverse proxy. The firewall function was assigned to the ModSecurity program. To install the ModSecurity web application firewall, the Apache webserver was installed and configured for further operation in the reverse proxy server mode. To block attacks, the most current version of the OWASP CRS rules downloaded from GitHub was uploaded to the webserver. To protect against denial of service attacks, distributed denial of service (DoS, DDoS) and brute-force attacks, the mod_evasive module was installed. The main settings for this module are located in the /etc/apache2/mods-enabled/evasive.conf file.

5. DISCUSSION OF RESEARCH RESULTS

Testing of the system was carried out in two stages: at the first stage, tools for automating the search for web vulnerabilities (web vulnerability scanners) were used. No high severity vulnerabilities were found when scanning a vulnerable application through a firewall. At the same time, there is a significant decrease in the number of vulnerabilities of medium and low severity levels.

At the second stage, manual testing of applications for vulnerabilities of SQL injection, cross-site scripting, and Path Traversal attacks was carried out. When an attempt was made to attack an application protected by a firewall, the response was "403 Forbidden", which indicates the impossibility of carrying out attacks. ModSecurity uses two types of logs to track webserver attacks: the error log (error.log) and the modsec_audit.log audit log. An error log is generated when an error is encountered or when an attack is attempted. Since ModSecurity is paired with Apache, all error logs (Apache error logs + ModSecurity error logs) are generated in one file. The audit log begins to fill up after an event is recorded in the error log. The audit log records more detailed information about the blocked attack. ModSecurity audit logs are generated according to the unique identifiers of the error log.

The issues of using neural networks to study traffic from IoT devices to determine possible threats from such devices need to be studied.

Using WAF in zero-trust systems is a fairly common option for securing services within an organization. But the use of open solutions in this approach allows you to customize protection more flexibly and personally according to the corresponding needs. It is also an opportunity to work with the introduction of elements of artificial intelligence in the approach to protection. In future studies, we plan to consider approaches to using neural networks to study traffic passing through the WAF and use the data obtained to improve the artificial intelligence algorithm for identifying threats [14-16].

6. CONCLUSIONS

A pilot study was carried out Using WAF to protect internal services in the Zero Trust structure. The system was tested in two stages. First, we used tools to automate the search for web vulnerabilities (web vulnerability scanners). No high severity vulnerabilities were found when scanning a vulnerable application through a firewall. At the same time, there is a significant decrease in the number of vulnerabilities of medium and low severity levels. At the second stage, manual testing of applications for vulnerabilities of SQL injection, cross-site scripting, and Path Traversal attacks was carried out. When an attempt was made to attack an application protected by a firewall, the response was "403 Forbidden", which indicates the impossibility of carrying out attacks. ModSecurity uses two types of logs to track webserver attacks: the error log (error.log) and the modsec_audit.log audit log. An error log is generated when an error is encountered or when an attack is attempted. Since ModSecurity is paired with Apache, all error logs (Apache error logs + ModSecurity error logs) are generated in one file. The audit log begins to fill up after an event is recorded in the error log. The audit log records more detailed information about a blocked attack. ModSecurity audit logs are generated according to the unique identifiers of the error log.

Also we need to study the use of neural networks to study traffic from IoT devices to identify possible threats from such devices. But we have the opportunity to improve the protection and protection algorithm of services in local area networks for the protection of end users and services.

The use of WAF in zero-trusted systems is a fairly common option for protecting services within the organization. But the use of open solutions in this approach makes it possible to more flexibly and personally adjust the protection to the appropriate needs. It is also an opportunity to work on the introduction of elements of artificial intelligence in the approach to protection. In future research, we plan to consider approaches to using neural networks to study traffic passing through WAF and use the data to improve the artificial intelligence algorithm to identify threats.

REFERENCES:

[1] D. APPELT, C. D. NGUYEN, A. PANICHELLA and L. C. BRIAND, "A Machine-Learning-Driven Evolutionary Approach for Testing Web Application Firewalls," in *IEEE Transactions on Reliability*, vol. 67, no. 3, pp. 733-757, Sept. 2018, doi: 10.1109/TR.2018.2805763.

[2] APPELT, Dennis & NGUYEN, Cu & BRIAND, Lionel & ALSHAHWAN, Nadia. (2014). Automated Testing for SQL Injection Vulnerabilities: An Input Mutation Approach. 2014 *International Symposium on Software Testing and Analysis, ISSTA 2014* - Proceedings. 10.1145/2610384.2610403.

[3] Chen, ZHIYU & Yan, LONGCHUAN & Lü, ZITONG & Zhang, YANLING & Guo, YONGHE & Liu, WENJING & Xuan, JIAXING. (2021). Research on Zero-trust Security Protection Technology of Power IoT based on Blockchain. *Journal of Physics: Conference Series.* 1769. 012039. 10.1088/1742-6596/1769/1/012039.

[4] DEMERTZIS, Konstantinos; ILIADIS, Lazaros. Cognitive Web Application Firewall to Critical Infrastructures Protection from Phishing Attacks. *Journal of Computations & Modelling*, 2019, 9.2: 1-26.

[5] GARBIS, Jason; CHAPMAN, Jerry W. Zero Trust Architectures. In: *Zero Trust Security*. Apress, Berkeley, Ca, 2021. P. 19-51.

[6] JINGYAO, Sun, et al. Securing a Network: How Effective Using Firewalls and VPNs Are?. In: *Future of Information and Communication Conference*. Springer, Cham, 2019. p. 1050-1068.

[7] MACDONALD, Neil; ORANS, Lawrence; SKORUPA, Joe. The Future Of Network Security Is In The Cloud. *Gartner. Viitattu*, 2019, 1: 2021.

[8] RATH, Annanda, et al. Security Pattern for Cloud SaaS: From System and Data Security to Privacy Case Study in AWS and Azure. *Computers*, 2019, 8.2: 34.

[9] STEFANOVIC, Vladimir; KATINSKI, Milos. Network Traffic Management. In: *Pro Azure Administration and Automation*. Apress, Berkeley, CA, 2021. p. 215-245.

[10] Y. SUN, S. NANDA and T. JAEGER, "Security-as-a-Service for Microservices-Based Cloud Applications," *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, 2015, pp. 50-57, doi: 10.1109/CloudCom.2015.93..

[11] TORRANO-GIMENEZ, Carmen, et al. An Anomaly-based Web Application Firewall. In: *SECRYPT*. 2009. p. 23-28.

[12] KHOROLSKA K., LAZORENKO V., BEBESHKO B., DESIATKO A., KHARCHENKO O., YAREMYCH V. (2022) Usage of Clustering in Decision Support System. Intelligent Sustainable Systems. Lecture Notes in Networks and Systems, vol 213. Springer, Singapore. https://doi.org/10.1007/978-981-16-2422-3_49

[13] BEBESHKO, B., KHOROLSKA, K., KOTENKO, N., KHARCHENKO, O., & ZHYROVA, T. (2021). Use of neural networks for predicting cyberattacks. Paper presented at the CEUR Workshop Proceedings, 2923 213-223

[14] LAKHNO, V., MALYUKOV, V., AKHMETOV, B., KASATKIN, D., PLYSKA, L. (2021). Development of a model for choosing strategies for investing in information security, Eastern-European Journal of Enterprise Technologies, 2 (3-110), pp. 43-51.

[15] LAKHNO, V., AKHMETOV, B., MAZARAKI, A., KRYVORUCHKO, O., CHUBAIEVSKYI, V., DESIATKO, A. Methodology for assessing the effectiveness of measures aimed at ensuring information security of the object of informatization, (2021) Journal of Theoretical and Applied Information Technology, 99 (14), pp. 3417-3427.

[16] Buriachok, V., Ageyev, D., Zhyltsov, O., Skladannyi, P., & Sokolov, V. (2020). Invasion detection model using two-stage criterion of detection of network anomalies. Paper presented at the CEUR Workshop Proceedings, , 2746 23-32.