

Network Security final Project

1st Akram Hamdi 2nd Arunjunai Rajan Senthil Kumar
3rd Suchdeep Singh Juneja 4th Tanguy Marbot

Abstract—When having a given system, concerns about security always arise as one of the first known problems. Security on perhaps the storage of passwords or other sensitive data. When using a said system usually the assumption is that the maintainers of the system have your interests at heart, and are committed to keeping the service safe for any intruders. One category of such a safeguard is an intrusion prevention system, a form of security that works to prevent identified threats by detecting them before they take place. For the course of Network security we, the team GroupFour, were tasked with implementing an alternative monitor to the widely known prevention system known as fail2ban. The goal was to create these monitors for the panels of Wordpress, SSH, Joomla and phpMyAdmin. This report will go into depth on the background on these systems and platforms and explain our findings.

I. INTRODUCTION

In the current day, a lot of research on intrusion prevention systems has already been performed. An intrusion prevention system, as is already mentioned, is known as a security form that wants to prevent threats to the system before they can take place and do any harm. One easy example of this is DDos protection, where you, for example, block traffic from a certain sender for a period of time when suspecting them of trying to take down the system. One famously known example of an intrusion prevention system is known as the system of fail2ban [3]. Fail2ban is a program that is able to scan log files in, for example, Apache servers and is able to ban IP's that show some malicious signs. These signs include, but are not limited to, too many faulty logins or seeking for certain exploits in the server with certain inputs. After Fail2ban detects that something went wrong or a threat is imminent, the IP from which the signs came from is banned and all traffic coming from that IP is blocked completely. The amount of time and other rules for IP banning can be customized as well for the specific server depending on what the users preferences are. A lot of research in the current day has been performed on these systems, and some significant steps have been made in improving these systems. More on the improvements in the field and the state of the art techniques on intrusion prevention systems can be

found in the related work section.

For the course of Network Security we were tasked with implementing an alternative fail2ban monitor for some platforms, such as Joomla or Wordpress. We should be able to block all traffic from certain IP addresses depending on what they try to do, the main focus on filtering many login attempts and block traffic depending on that. In addition to this, we were tasked with creating a web interface that is able to change the ban time and also the amount of requests it takes to get a certain IP address banned. Additionally we should also be able to undo bans and see a list of banned IP's. And lastly, this article is the final part of this project.

II. BACKGROUND

In this section we will discuss the background on the relevant parts of the project. We will go over the process of banning IP's, as well as go over the different platforms we were tasked with implementing the alternative detection system for.

A. Fail2Ban in depth

Fail2Ban consists of two parts, the client, and the server [3]. The server is multi threaded and listens to comments on a UNIX socket, while the client is the front end of Fail2Ban. Fail2Ban aims to ban IP addresses by scanning certain log files from the server. These log files are always generated by the server in order to keep track of who and what is trying to get into the server and for other purposes. Fail2ban is mainly focused on preventing brute force attacks, and makes it easier to ban the adversaries since going through a log file that possibly contains thousands of log in attempts can be a big chore. Then after the log file is scanned and some malicious activity is detected, Fail2ban enters and bans the IP address usually by adding a firewall rule. Fail2Ban defines an `actions.d` folder containing multiple different commands that are executed when, for example, banning an IP address. Consider the figure on the next page:

```
# Fail2Ban action file for firewall-cmd/ipset
#
# This requires:
# ipset (package: ipset)
# firewall-cmd (package: firewallld)
#
# This is for ipset protocol 6 (and hopefully later) (ipset v6.14).
# Use ipset -V to see the protocol and version.
#
# IPset was a feature introduced in the linux kernel 2.6.39 and 3.0.0 kernels.
#
# If you are running on an older kernel you may need to patch in external
# modules.
[INCLUDES]
before = iptables-blocktype.conf
[Definition]
actionstart = ipset create fail2ban-<name> hash:ip timeout <bantime>
               firewall-cmd --direct --add-rule ipv4 filter <chain> 0 -p <protocol> -m multiport
               --dports <port> -m set --match-set fail2ban-<name> src -j <blocktype>
actionstop = firewall-cmd --direct --remove-rule ipv4 filter <chain> 0 -p <protocol> -m multiport
               --dports <port> -m set --match-set fail2ban-<name> src -j <blocktype>
               ipset flush fail2ban-<name>
               ipset destroy fail2ban-<name>
actionban = ipset add fail2ban-<name> <ip> timeout <bantime> -exist
actionunban = ipset del fail2ban-<name> <ip> -exist
```

Fig. 1. Snippet from the `firewallcmd-ipset.conf` file, note that the commands that are defined here are performed when banning or unbanning a certain IP address

Here in this figure we can see that Fail2ban already predefined a lot of commands to make the banning process go automatically. The Fail2ban server listens to commands on a Unix sockets, so it's perfectly compatible with these Unix terminal commands.

B. SSH

The *Secure Shell Protocol*, or **SSH** for short, is a cryptographic network protocol where an SSH instance can connect to an SSH server [14]. It is most widely used for remote login to another system and executing command line instructions on that same system from a remote location. These applications are based on a client-server architecture, and comes with 3 layers. The transport layer, which handles authentication of the server, integrity and confidentiality of the data. The user authentication protocol, which validates the user. And the connection protocol, which expands the encrypted tunnel of communication into multiple communication channels [14]. The main purpose of SSH was to be a replacements of the Telnet and other unsecure remote Unix shell protocols. For a more simplified view of the SSH protocol, consider the figure below: Whilst this

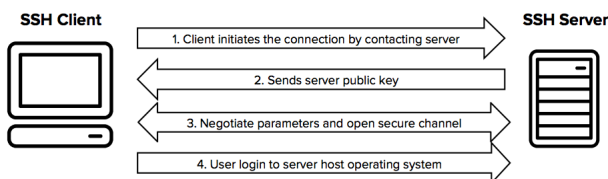


Fig. 2. The simplified SSH protocol, note that this connection between client and server must remain confidential [11].

protocol was believed to be a reasonably safe protocol of communication, it may depend on the context as

well. However, older cryptographic schemes such as the Diffie Hellman Key Exchange protocol [2] have been proven to be vulnerable to so called man in the middle attacks, where a router between the protocol of key exchange can examine all communication, but may not be able to alter it [7]. Whereas newer protocols like the group diffie hellman protocol may be sufficient, using older versions of this key exchange may prove to be an issue, and the newer SSH protocol don't allow these unsafe protocol configurations by default anymore. It specifically prints a warning message saying a man in the middle attack may be performed when running ssh with the current configuration.

Fail2Ban natively supports securing SSH connections, and is able to completely block traffic from your IP address if you fail to log into the server properly after multiple tries [4]. Fail2Ban manages configurations and rules by using a `jail.local` file. In this file you can add the specific security configurations, like for example, setting the type of connection to secure to be SSH. With the Fail2Ban service running in the background and restarting the SSH service too, it will use the configuration for the jailing automatically, and will monitor the service on faulty login attempts. Fail2Ban will keep track of the SSH log file and scan it for the failed login attempts made by the client. Since it does this in realtime, Fail2Ban can quickly add the rule to the firewall, and block the traffic altogether. However do note that for this to be able to happen, the `passwordAuthentication` setting in the `/etc/ssh/sshd_config` must be set to yes! An example of how this looks like in action can be seen in the figure below:

```
ssh dev@192.168.1.107
dev@192.168.1.107's password:
Permission denied, please try again.
dev@192.168.1.107's password:
Permission denied, please try again.
dev@192.168.1.107's password:
dev@192.168.1.107: Permission denied (publickey,password).
ssh dev@192.168.1.107
dev@192.168.1.107's password:
Permission denied, please try again.
dev@192.168.1.107's password:
Connection closed by 192.168.1.107 port 22
sh dev@192.168.1.107
ssh: connect to host 192.168.1.107 port 22: Connection refused
```

Fig. 3. By using faulty login attempts whilst having Fail2Ban active, you can block communication from that IP completely [4]

C. Wordpress

Another platform we had to consider for our project was Wordpress. Wordpress is a platform that allows the building of webpages using pre made templates or using

your own ones. It was initially supposed to be for blog publishing, although it supports many other web content types now as well. It also allows for request routing so you can have an easier URL for your website. It is mainly written in PHP along with a database in either MYSQL or MariaDB. In order for Wordpress to work it has to be installed on a web server, either locally or remote. And with hosting a web server a multitude of different security related risks arise. When making a, for example, forum based website with a login protocol, this already creates a ton of issues when considering the storage of user accounts, passwords, etc.

Because any arbitrary wordpress site may be used for a wider public, it's important that these security measures are taken to prevent abuse and maintain the stability of the server and site. Many different Wordpress security guides exist, providing plenty of information and documentation for the not so experienced developer in the world of webserver hosting and webpage engineering. Wordpress themselves also provide documentation on security on their website [15]. They mention the so called OWASP top 10 list, which Wordpress has security measures against. Although protection against DDOS attacks for example are not specifically mentioned in this post. However, Wordpress does have a specific plugin for Fail2ban, allowing protection against these attacks if your web app uses a certain authentication protocol. This plugin provides a custom filter made specifically for Wordpress log scanning and filtering, and allows you to point Fail2Ban to that specific configuration file. Afterwards you can edit the jail file of Fail2Ban to add the specific rules you want the protocol to enforce, such as the max amount of retries allowed, or the ban time.

D. Joomla

Joomla is quite a similar system compared to Wordpress in that it allows a user friendly way to publish websites with your own content. These include, but are not limited to, discussion forums or blogs. It is written in PHP just like Wordpress and uses a MariaDB database as well. These two systems of making websites are quite familiar, but there are some differences however. Where Wordpress is more focused for the general public and easy to use tools, Joomla already comes with more complex systems for user access control and user management [6]. In addition to that, Joomla allows the usage of multiple languages and also multiple templates. Joomla is overall considered a better option for the more experienced developer, with a multitude of options

that can be utilized better by developers compared to non developers or very low skilled ones.

Luckily however Joomla also supports Fail2Ban and all of its benefits [5]. The administrator panel of Joomla defines a certain path to the log folder where all the logging takes place. This logging includes, but isn't limited to, login attempts. Although this log file themselves doesn't contain much information, using the user logging plugin does allow for more elaborate loggings of users, such as the IP address which we want to use to block traffic in the case of malicious behavior. Another difference compared to using Wordpress is the fact that the filter isn't provided by a plugin for Joomla, but has to be made yourself. By filtering a certain regex and looking for the `joomlafailure` keyword in the logs created by the user log plugin, the filter can be created and added to the Fail2Ban filters. Then the developer needs to create the jail file with the necessary rules, the rules structure differs a bit from the one wordpress uses, but is quite similar. Consider the figures below for a comparison:

```
[joomla-login-errors]
enabled = true
filter = joomla-login-errors
port = http,https
maxretry = 6
findtime = 18000
bantime = 18000
logpath = /var/www/*/tmp/*error.php
          /var/www/*/htdocs/tmp/*error.php
          /var/www/*/htdocs/logs/*error.php
          /var/www/*/apps/logs/*error.php
          /var/www/*/htdocs/administrator/logs/*error.php
          /var/www/*/apps/*/administrator/logs/*error.php
```

Fig. 4. Joomla's jail file [5]

```
[wordpress-hard]

enabled = true
filter = wordpress-hard
logpath = /var/log/auth.log
maxretry = 3
port = http,https
```

Fig. 5. Wordpress' Jail file [10]

As can be seen in the two figures, the jail file configuration for both Wordpress and Joomla are very similar. Although Joomla has additional fields for the ban time or the find time, and also multiple log paths (since the Joomla server configuration may allow for logging to multiple files). This is because the error logs of faulty attempts aren't being kept track of with additional IP information by default, so it's best to scan multiple log files to search for the specific IP and flag that says a faulty login has been performed, also known as the before mentioned `joomlafailure` flag. By filtering on this specific key word in the conf file the scanning can start.

E. *phpMyAdmin*

The last platform we were tasked with implementing the Fail2Ban monitor for is the platform of phpMyAdmin, a free software tool written in PHP to handle the administration of MySQL over the Web [9]. In addition to this MariaDB is also supported. The clear difference between Wordpress, Joomla and this platform is the fact that with the other two platforms it's also aimed at making your own webpages, whereas phpMyAdmin only handles the administrative tasks of the database server. Therefore phpMyAdmin can be combined with either Wordpress or Joomla in order to handle all the database related tasks that Wordpress or Joomla rely on. On the host control panels of Wordpress, like cPanel or Plesk, it's even pre-installed.

Relevant functionalities of phpMyAdmin include, but are not limited to, multiple operations on databases, and its tables and columns, administration of multiple servers and the tracking of changes within these databases. Databases are very important for the storage of valuable data that makes the web applications and web servers work. Anything ranging from a blog posting site to a forum site, or even your regular news site all rely on databases for the storage of the data it wants to show, or the customer data it wants to store. Because of course, when a user creates an account with all the relevant credentials, it's important that this data is stored for later authentication. Having this data compromised can be devastating for the safety of customer data, such as passwords, and also for the reputation of safety of the maintainer of the service.

Luckily, Fail2Ban is also compatible with this service of phpMyAdmin. Since Fail2Ban runs on a server with a web interface, it also requires a login authentication in order to be able to access the tools of changing

the database and perform all the operations. The way the implementation works of the Fail2Ban program for phpMyAdmin is very similar to the one of Joomla which was described before. It mainly comes down to adding a custom filter specifically altered for the regex of phpMyAdmin [13]. In order to trigger this however, log files need to be present, which only require a change in the Apache configuration in order to enable the logging in the certain format that you want. After this has been performed, and the filter has been created, the jail can be built and after reloading the Fail2Ban and Apache web server services, the configuration is enabled and Fail2Ban will protect phpMyAdmin against faulty login attempts. For the sake of completion, the jail of phpMyAdmin can be seen in the figure below:

```
[phpmyadmin]
enabled = true
port = http,https
filter = phpmyadmin
logpath = /var/log/apache2/phpmyadmin_access.log
```

Fig. 6. The Jail file for the phpMyAdmin [13]

For the final project of the Network Security course, our task was to create an alternative algorithm which performs the same tasks and fulfills the same purposes as Fail2Ban does: the prevention of brute force faulty login attempts and providing choices for configuration and options that can be tweaked to the wishes of the users. In the following sections we will dive deeper into our own projects, methods, and ways of tackling this subject.

III. METHODOLOGY

The first idea, was to separate the tasks into two main scripts. The first one would take care of configuring the settings, used in our fail2ban procedure. The second one implements the core logic of the fail2ban. It was also first thought to separate the process of parsing the file and getting the parameters for each jail. It was later decided that the script responsible for reading the parameters of the fails, would also take care of the banning of IP. During our investigation, we followed two main strategies.

A. First Idea

In our first idea, we separated the project into 4 different scripts. The first one is a script that gets arguments from the user. The arguments are one of the four jail names, the ban time in seconds, the maximum number of failed authentication attempts, the time window to monitor the number of failed authentications. This configuration script would create or append the local jail fail. If the config for the given jail already exists it will modify it. This script keeps as constant, logpath and regex filters associated to each jail.

The second script doesn't get arguments. It will constantly monitor the local jail file, create a thread to monitor to configured jails. The thread essentially checks for the regex in the logpath, if the regex is found within the failure window for a maximum number of retry times, then add the IP table and start a timer for the ban time. When the timer is expired, the IP table is removed. When a user enters settings to the configuration script, a thread can either continue and change the parameters to monitor for the banning, or get created if it didn't exist. When a user disables a ban, the thread is killed.

A third python script can be used by the user to disable a jail.

The final script concerns the web hosting. It should allow the user to change the settings of the fail2ban as described earlier but through a graphical user interface. Essentially this script will serve to configure the fail2ban in a more convenient way for the user. The reference of [8] is used as the base for our web host GUI

B. Second Idea

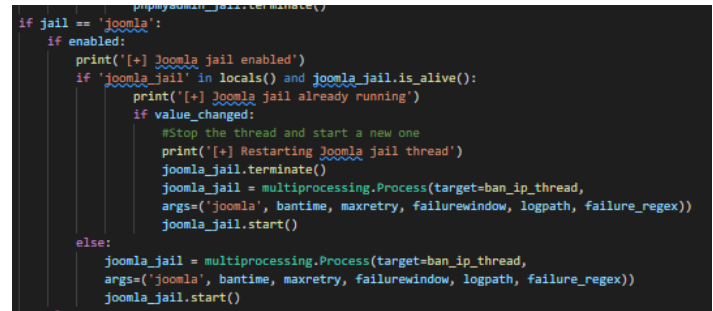
The second idea is in the same vein. But 3 scripts are used instead of 4, because a new setting can be entered from the user which is enable(true) or disable(false), for a particular jail.

The second file, is essentially the same, but makes sure to use the external ConfigParser python module to read the configuration file continuously. Handles killing on

disabling of jail properly. If thread killing is not possible, we need to keep the configuration file always there with some default values for all the jails and just update the new values as needed.

C. Third Idea

Eventually, used multiprocessing python module instead of multithreading, which enabled the process killing easier using terminate routine. A snippet performing process termination in the case of Joomla can be seen in the figure below.



```

def joomla_jail_terminate():
    if jail == 'joomla':
        if enabled:
            print('[+] Joomla jail enabled')
            if 'joomla_jail' in locals() and joomla_jail.is_alive():
                print('[+] Joomla jail already running')
                if value_changed:
                    #Stop the thread and start a new one
                    print('[+] Restarting Joomla jail thread')
                    joomla_jail.terminate()
                    joomla_jail = multiprocessing.Process(target=ban_ip_thread,
                                                         args=('joomla', bantime, maxretry, failurewindow, logpath, failure_regex))
                    joomla_jail.start()
            else:
                joomla_jail = multiprocessing.Process(target=ban_ip_thread,
                                                         args=('joomla', bantime, maxretry, failurewindow, logpath, failure_regex))
                joomla_jail.start()

```

Fig. 7. Snippet from the source code performing thread termination for the Joomla process

Additionally, the processes were restarted every time there is config change noticed in the jail file. This is to make sure there is

IV. EXPERIMENTS

A. Setup used to experiment and test

Host computer running in Windows 11 and a Virtual Machine running Ubuntu 20.04 LTS was used to test the whole system and developed scripts to be an alternative to the fail2ban. A NAT connection was established between Virtual machine and Host machine, so that both could be in same subnet. Then the server and database configuration needed to host Wordpress, Joomla and phpMyAdmin was installed and setup in Virtual machine. Then the scripts to ban IP and Configure through Webpage UI was also run on virtual machine. The brute attacks were then performed from Host windows machine.

B. Base functionality with tweaking

One of the requirements of the system we were tasked with implementing was that it should be able to be tweaked. When using Fail2Ban this is an option as well, since rules can be added in the jail file which contain the specific rules for the filter that it's designed for. Rules like the amount of permitted login attempts within

a given time or the time someone is banned can all be changed, and should be able to be changed in our approach as well. When trying to test our local self made system, we tried to get into the system by first testing the program normally. This means that depending on the configuration and the specific jail, trying to get the system to ban us out. Whilst testing this, it seemed like that we actually did get blocked out, and weren't able to get in or connect at all. By examining the log file that the web services output we can see that failed login attempts are logged, which our system can then detect and handle the rule adding to the firewall. It would of course already be problematic if the basic functionalities of the prevention system would not work, since that would mean that the server could be easily vulnerable against brute force attacks on the login files, leading to an unwanted access to your server.

C. Threading

Whilst using this program locally usually requires only one web server instance (and therefore jail instance) to be running, this is of course not the only way of approaching this issue. Fail2Ban allow, for example, running multiple jail configurations at once. When trying to run multiple jails for multiple web services, it doesn't really make sense to stick to a single threaded program. Therefore an interesting experiment to conduct is on the experimentation on threading and the termination of these threads during run time. In normal circumstances, when disabling the jail for the chosen system (and therefore the thread), the thread should instantly terminate in order for the security to be maintained. It would be extremely problematic if the protection in the jail would just instantly disappear with the server still running, since that would mean that the login authentication on the server could easily just be brute forced without any action being taken, leading to potential sensible data to be compromised. With this experiment we could test if our own system handles this properly, and if the thread termination happens fast enough.

V. RESULTS

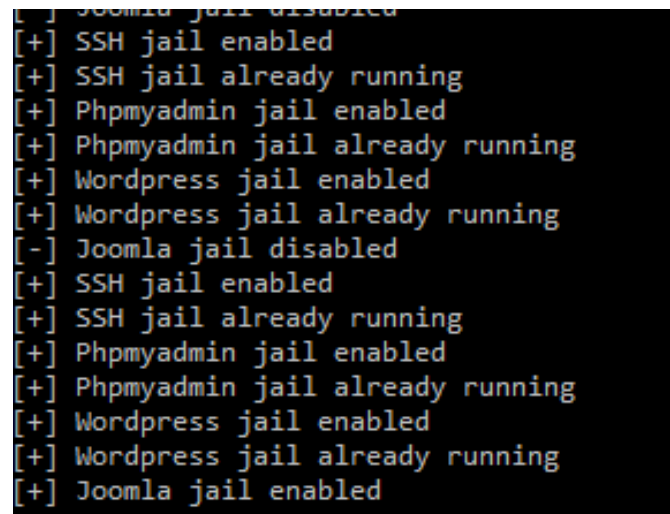
A. Blocking Traffic

This experiment did prove to be a success. When trying to block out the traffic from the IP address when the jail was running and too many faulty login attempts were performed, it could be observed that connection attempts afterwards just failed entirely. In addition to this, it could also be observed that those faulty logins were indeed logged away, therefore confirming that the reading of the log files was a success. The collaborative

jail having the multiple rules was working, and it scanned multiple logs if possible, although we did also add functionality of disabling or enabling certain jails when desired.

B. Process termination

As seen before in the methodology section of the article, we talked about the termination of processes if necessary. If the process of the jail is not needed anymore, due to either the server shutting down or the changing of the settings, the joomla jail will be disabled and the thread in which it's running will be terminated. When observing the terminal output we can see that in fact there is a change in the jail statuses and that the thread actually gets enabled again when it happens, see figure 8.



```

[+] Joomla jail disabled
[+] SSH jail enabled
[+] SSH jail already running
[+] Phpmyadmin jail enabled
[+] Phpmyadmin jail already running
[+] Wordpress jail enabled
[+] Wordpress jail already running
[-] Joomla jail disabled
[+] SSH jail enabled
[+] SSH jail already running
[+] Phpmyadmin jail enabled
[+] Phpmyadmin jail already running
[+] Wordpress jail enabled
[+] Wordpress jail already running
[+] Joomla jail enabled

```

Fig. 8. Output of whether or not the threads are active, note that the status of joomla thread changes when we change the configurations and enable the jail again!

As we can clearly see from this snippet from the output of the system, the joomla jail gets re-enabled, therefore making our process termination and reviving successful.

C. Webpage UI to configure

As explained in the methodology section, a web interface was developed using Flask module to allow the users to enable/disable the Jails as per their need. And also it allows users to configure Bantime, Maxretry and Failure window to monitor through GUI. This web hosting script using config script to configure the 'custom_jail.conf' file. Another script responsible for running the main logic to ban the IPs would monitor the conf file and act as expected. Figure 9 shows the screen grab of Home page where users can configure the parameters of config file.

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/config'. The page has a dark header with 'Secure Admin Dashboard' and a 'Config' tab. The main content area has a light green background and contains four configuration sections:

- ssh** (green header): Includes a 'Disable' button, a 'Max Retry' input field with the value '10', a 'Ban Time(In Seconds)' input field with the value '300', a 'Failure Window(In Seconds)' input field with the value '600', and a 'Submit' button.
- joomla** (pink header): Includes an 'Enable' button.
- wordpress** (pink header): Includes an 'Enable' button.
- phpmyadmin** (pink header): Includes an 'Enable' button.

Fig. 9. Screen grab of the Webpage UI developed to configure the parameters of jail file

VI. DISCUSSION

From the implementation perspective, developed scripts lacks a Graphical UI to monitor the list of banned IPs and remove them from the ban list as needed. But the implementation to do so, can be scoped and tested in future. Also right now, for the whole system to work, two scripts need to run with root permission in background. May be it can be limited to one script which renders GUI webpage and also handles the code to ban the IP could be implemented.

Looking at how the experiments went, we can see that blocking IP addresses isn't necessarily that complex of an operation to perform. It usually merely boils down to adding an additional firewall rule that prevents that specific IP from entering a connection with the server. However, in reality, brute force attacks usually occur from many different IP addresses. Whereas our system mainly focuses on blocking a certain IP address, what might happen is that when there are a lot of different IP addresses trying to attack the system at the same time (this can be achieved by, for example, spoofing) the system might not be able to be fast enough to keep track of all the different attempts. Log files may scale

indefinitely, and the prevention system may be very late on trying to scan the logs or perhaps even crash due to the large workload that is being put on the system. All in all this means that this system is mostly limited for local use for practice purposes, although it does not mean that it isn't a valuable learning opportunity. But it's mostly limited to that.

VII. RELATED WORK

In the current day, lots of research into the development of these intrusion prevention systems has already been performed. The first steps with the designing of these prevention systems stem from 2004, where it was proposed as a good solution to improving network security. It essentially is an improvement on the then already existing intrusion detection systems, which mainly focusses on just the detection of malicious behavior and not necessarily on also acting upon those threats. The intrusion prevention systems combine the workings of detecting the threats with the techniques used in firewalls, as could be seen in the Fail2Ban explanations in the background section. It essentially offers both detection and protection, which makes the overall defense mechanism a lot better to use [16].

However, that paper was the first real paper that actively contributed to the design of these prevention systems. That paper dates back until 2004, and since then a lot more has happened. Since the first idea of the IPS's was proposed, a wide spread of implementations of these systems has been observed. And when a given system gets implemented more and more, what also arises with time are issues with the security side of the systems. Issues with IPS's include, but are not limited to, issues with the volume of traffic (note that in the current day a lot more traffic goes through the systems compared to years ago), and also efficient logging. Whilst Fail2Ban is dependant on logging files to scan for issues and malicious behavior, this may prove to be a bigger issue than thought due to the high volume of logging that is going on in systems. Whilst these issues definitely are considerable, solutions have already been proposed [12].

Lastly, when looking at more state of the art literature on IPS's, we see that usually the modern strategy of tackling these security issues stem from the same fundamental ideas and approaches as seen in the trends paper on IPS's [12], namely the Anomaly Detection Approach, the Signature Detection Approach and the Hybrid Detection Approach. Whereas the state of the art paper improves on these concepts and come up with

newer ideas using more recent tools to achieve these goals [1].

VIII. CONCLUSION

We have seen a lot today on the emergence of these intrusion prevention systems. There is already a lot of research and a lot of jumps in the development of better and better systems in the current day. Although those trends show that it usually boils down to the same idea: preventing threats rather than repressing threats when they've already taken place. In today's world, the security of our data, and more so our lives, are in the hands of systems that we assume to be safe and we assume to have our best interests at heart. Being able to try and develop such a system helps a lot with understanding how such systems work in practice on a more smaller scale, as it essentially boils down to detecting strange, potentially harming behavior in a system, and then acting upon it in order to stop the attack and potential leaks or system shutdowns from even happening at all. By implementing this system, we have proven to be able to implement a small local system that keeps track of faulty logins within a certain time frames, something that shows characteristics of a brute force login attack, trying to get into a system by trying many different passwords until one actually works. These attacks can be prevented if a certain IP address tries lots and lots of attempts within a very short period of time. When this happens at a rate that can be considered suspicious, our system comes in action and bans that IP, disallowing traffic from that IP address completely.

Additionally, we learned a lot on the background knowledge of the platforms and systems we were supposed to support. It's always good to get to know more about the tools that you use on a very regular basis, which helps getting to understand the security implications that come with trying to develop new ideas and services for the public, which is an important part of the Computer Science study as a whole. In the end, this project taught us a lot about the implications and risks that arise with building the systems, and although in reality most attacks are already far more complicated and well thought out, it doesn't take away that we finish this assignment knowing more than when we started.

REFERENCES

- [1] Mohamed Amine Agalit, Ali Sadiqui, Youness Khamlichi, and El Mostapha Chakir. Hybrid intrusion detection system for wireless networks. In *WITS 2020*, pages 507–513. Springer, 2022.
- [2] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Dynamic group diffie-hellman key exchange under standard assumptions. In *International conference on the theory and applications of cryptographic techniques*, pages 321–336. Springer, 2002.
- [3] Fail2Ban. Fail2ban manual. URL: https://www.fail2ban.org/wiki/index.php/MANUAL_0_8.
- [4] Hackersploit. Using fail2ban for ssh brute-force protection. URL: <https://www.linode.com/docs/guides/how-to-use-fail2ban-for-ssh-brute-force-protection/>.
- [5] Andre Hotzler. Protect joomla login with fail2ban. URL: <https://www.andrehotzler.de/en/blog/technology/63-protect-joomla-login-with-fail2ban.html>.
- [6] Brian Jackson. Joomla vs wordpress. URL: <https://kinsta.com/blog/joomla-vs-wordpress/>.
- [7] Nan Li. Research on diffie-hellman key exchange protocol. In *2010 2nd International Conference on Computer Engineering and Technology*, volume 4, pages V4–634. IEEE, 2010.
- [8] oussemos. fail2ban-dashboard, 2018. URL: <https://github.com/oussemos/fail2ban-dashboard>.
- [9] phpMyAdmin. About. URL: phpmyadmin.net.
- [10] Brook Shelley. How to protect wordpress with fail2ban on ubuntu 14.04. URL: <https://www.digitalocean.com/community/tutorials/how-to-protect-wordpress-with-fail2ban-on-ubuntu-14-04>.
- [11] SSH.com. Ssh home page. URL: <https://www.ssh.com/academy/ssh>.
- [12] Deris Stiawan, Abdul Hanan Abdullah, and Mohd Yazid Idris. The trends of intrusion prevention system network. In *2010 2nd International Conference on Education Technology and Computer*, volume 4, pages V4–217. IEEE, 2010.
- [13] Lenz Weber. using fail2ban to secure the phpmyadmin login. URL: <https://phryneas.de/fail2ban-phpmyadmin>.
- [14] Wikipedia. Secure shell. URL: https://en.wikipedia.org/wiki/Secure_Shell.
- [15] Wordpress. Wordpress security. URL: <https://wordpress.org/about/security/>.
- [16] Xinyou Zhang, Chengzhong Li, and Wenbin Zheng. Intrusion prevention system design. In *The Fourth International Conference on Computer and Information Technology, 2004. CIT '04.*, pages 386–390, 2004. <https://doi.org/10.1109/CIT.2004.1357226> doi:10.1109/CIT.2004.1357226.