Cybersecurity Project

DD2391 HT23

Group 10
Tomi Toma
Qikun Tian
Pol Falguera Guillamón
Pol Fradera Insa

Table of contents

Database leakage and corruption

Introduction

Database leakage or damage refers to a situation where sensitive information in a database is accessed by unauthorized users, or where there are errors or disruptions in the structure or content of the database. Unauthorized remote access is one of the causes of such incidents. Therefore, to prevent remote access from leading to database leakage or damage, a series of security measures must be taken, including configuring access permissions correctly, using strong passwords, implementing firewall rules, and encrypting data transmission.

Threats

Data Leakage: Attackers can obtain sensitive information from the database, such as user data, passwords, and personal profiles, potentially leading to privacy breaches for users.

Data Tampering: Attackers can modify information in the database, potentially compromising data integrity, and even causing confusion or misleading users.

Data Loss: Attackers can delete information from the database, leading to data loss, which can have a severe impact on the normal operation of businesses or services.

Malicious Operations: Attackers can perform destructive operations, such as deleting tables or tampering with records, resulting in structural or content damage to the database.

Denial-of-Service (DoS) Attack: Attackers may overload the database server with a large number of malicious requests, causing it to become unresponsive and unable to provide services normally.

Countermeasure

There are many ways to prevent unauthorized remote access, one way is to set up an Nginx reverse proxy server where requests for the current server on port 80 are forwarded to another backend server. In addition to this, create some firewall rules using iptables to ensure that only explicitly allowed traffic passes through the firewall.

Another better approach is to configure MongoDB's `mongod.cfg` configuration file. By specifying the “bindIp” parameter with the local loopback address, MongoDB will only listen for connections coming from the local machine. Since MongoDB is configured to listen on the local address, it will ignore connection requests from other hosts on the network, effectively blocking any attempts

from remote hosts to connect to the MongoDB server. NodeBB can still access the MongoDB server by connecting to “127.0.0.1”, as they are both running on the same machine, allowing them to continue functioning normally.

Implementation

The installation may differ depending on the system used for the Mongo database but on windows the access restriction can we installed as follows:

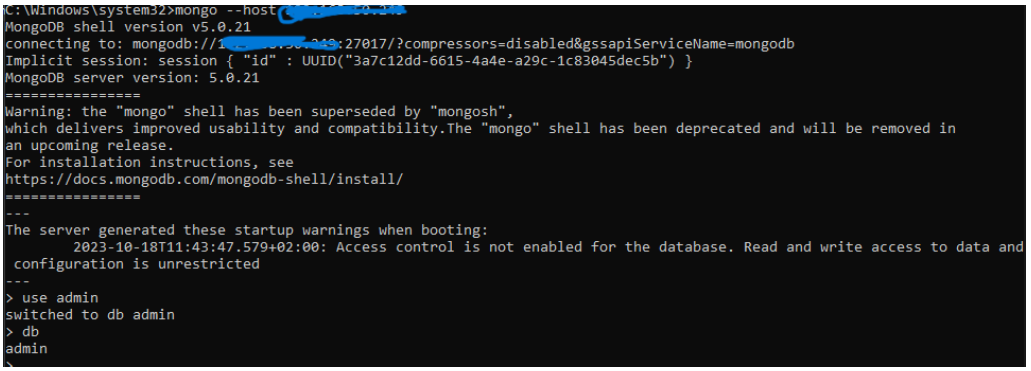
1. Run cmd as administrator and shut down mongodb using net stop mongodb
2. Navigate to the path where mongodb is installed on windows and find the mongod.cfg file
3. Open the file with and edit and add the following at the end:

net:
bindIp: 127.0.0.1

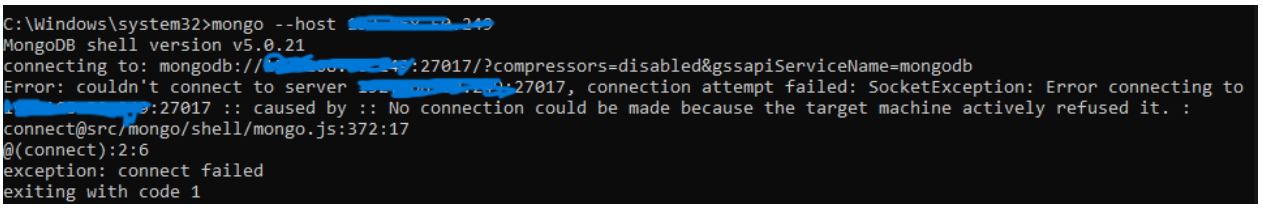
4. Save the file and start the mongodb again from the cmd by using net start mongodb

After this configuration, only localhost connections are accepted, which ensures that only local applications can connect to MongoDB and improves security.

This can be tested by before adding the configuration to connect from another machine using the command mongo –host ip-to-host and before binding the ip to the local machine you are able to connect but after adding the configuration the connection is refused.
These are a example:
Before:



After:



Difficulties

When I first started researching, I saw that the installation instructions stated to install Nginx.I looked up information on the web about what Nginx does and how it is used and I found out that Nginx is a reverse proxy server, so I associated that I could use Nginx to manage all the external traffic, but Nginx was very new to me, and I spent a lot of time on it. Then I learnt on the web that iptables can be used to create some firewall rules to block incoming and outgoing traffic. But iptables is used in linux system, I just started to install MongoDB and nodeBB is a Windows system, so I have to operate on the virtual machine, in the installation process, I encountered a lot of difficulties, because the official installation guide is too outdated, resulting in a lot of problems to follow the installation of its steps, but fortunately all solved these problems. Even though I thought I was on the right track, I still couldn't prevent any remote access to the database that is not mediated by NodeBB software, so maybe I missed some important part of the configuration. I had to ask for help from other members of the group, and eventually I was able to complete the assignment with a few simple lines of code to configure the database with the help of my teammates.

Spam

Introduction

Spam refers to any kind of unsolicited or unwanted communication, often being messages containing irrelevant or inappropriate content, that are sent over the internet. This communication is sent out in bulk and mostly via email, but can also be sent through messaging platforms, social media or even phone calls. Spam can take different forms, including advertisements, scams, phishing attempts, or other unwanted communications, and its primary goal is to promote products, services, or fraudulent schemes, often at the recipient's expense. [3]

A very common way of spamming rose up a few time ago when online forums started to appear: *forum spam.* This consists of posting messages on Internet forums with the different forms mentioned above. [4]

Threats

Spam in a forum can have various negative effects, for example a spammer can fill the forum with a large volume of irrelevant content (posts, links, advertisements) making it difficult for users to find valuable information. Also, a lot of spam posts can frustrate and annoy forum users. Therefore, users may become less active and have a negative perception of the community. [3]

Regarding security threats, spammers might distribute malware or phishing links through their posts, so innocent users who click on malicious links can expose themselves to security risks like viruses, identity theft and financial fraud. Moreover, some spam may involve illegal content, such as copyrighted material or hate speech, that can cause the shutdown of the forum for hosting or allowing illegal content.

Countermeasure

First of all, we noticed we could apply simple countermeasures from the admin account. As an admin, we can define a number of seconds that a user has to wait until making another post after making one. This would avoid users making posts in a bulk. You can also provide specific posting restrictions to new users, such as how much time does it have to pass until they can make their first post or having their first post approved before being published. This would allow us to identify new users whose aims are spamming the forum.
Another stronger countermeasure we have used for mitigating spam occurrences in the NodeBB environment is a provided plugin called *nodebb-spam-be-gone.* This anti-spam plugin uses three

different services to limit spam: ***Akismet***, ***Project Honey Pot***, ***Google reCAPTCHA***, ***StopForumSpam*** and ***hCaptcha***. Each of them require a one-time setup to activate. [5]

Akismet is a spam filtering service run by the blogging platform WordPress. When a user submits a comment on a forum supervised by Akismet, it is automatically being analyzed and marked in case it is considered spam.[5][6]
Project Honey Pot is a distributed system for identifying spammers and the spambots they use to scrape addresses from a website, by extracting useful features from the IP's that access the website.[5][7]
Google reCAPTCHA is a widely used security tool that helps protect websites from automated bots and spam by presenting challenges, such as identifying objects in images or solving puzzles, to users to prove they are human. [5][8]
Stop Forum Spam is a free service that records reports of spam on forums, blogs and wikis, among others. They make these records available to you, so you can search and view them but most importantly, they can be accessed in an automated way for blocking suspected spammers before they can get in the front door. [9]
hCaptcha is a security tool almost identical to *Google reCAPTCHA.* The aim is the same and they both implement similar strategies. [10]

Implementation

To limit the number of posts within a certain time frame, you can access Settings -> Posts -> Posting Restrictions. Here, you can specify the time interval between posts for all users. It is not advisable to set a large number of seconds, as it may hinder the forum's operation.

To restrict the number of posts from new users, navigate to Settings -> Posts -> New User Restrictions. Here, you can set the time interval between posts for new users and establish a Reputation threshold to determine who qualifies as a new user.

To activate the post queue, access Settings -> Posts -> Post Queue. You can enable the post queue and define the required reputation level for users to bypass it.

Regarding the nodebb-plugin-spam-be-gone, you need to install and activate it. To do this, you must be logged in as an admin and navigate to Admin -> Plugins -> Install Plugins. [5]

Moreover, you need to set up keys and configure plugins for all services. To do this, go to Admin -> Plugins -> Spam Be Gone. In each service, you must follow the provided links to obtain the keys and configure them on your own. [5]

Difficulties

The main difficulties we had were in the testing process. First, we found a script on the Internet that was supposed to log in into NodeBB automatically and, once logged in, it would start to post comments in NodeBB in bulk, as a spam attack could look. This script worked until the posting part, where the script just halted and did not post the comments.

Since we spent a lot of time trying to solve it with no positive results, we decided to manually create posts that could be considered spam, so then our anti-spam countermeasures would take place. When doing this, we encountered the problem that NodeBB was not allowing us to post comments. Since the reason remains unknown, we decided to uninstall NodeBB and install it again. The problem then was gone and we have been able to upload some screenshots to show our countermeasure works properly.

Unauthorized access

Introduction

Brute force attacks are a popular form of cybersecurity threat where malicious actors attempt to gain unauthorized access to systems or accounts by systematically trying various combinations of usernames and passwords. These attacks involve automated scripts or bots that repeatedly submit login requests using different credentials until they find the correct combination or gain access. Brute force attacks are a significant concern for online platforms as they can lead to compromised user accounts, data breaches, and potential misuse of the system.

Threats

One of the significant security threats faced by online platforms like NodeBB is brute force attacks, where an attacker uses automated scripts or bots to repeatedly attempt to guess user passwords. This poses a severe security risk as it can lead to unauthorized access to user accounts which in turn can lead to other threats within the website the attacker managed to get unauthorized access.

The impact of a successful bruteforce attack can have multiple serious consequences. If an attacker manages to gain access to a user account they can steal sensitive personal data associated with that account. The attacker can use the unauthorized access to impersonate the person and use that identity to trick other users and could possibly conduct a scam or spread malware because others might not hesitate to open a link sent by someone they know. If an attacker is lucky they might manage to brute force an admin account which will give them unrestricted access to the website and different admin interfaces which can be used to cause more harm and steal more personal data.

Countermeasure

There are multiple and different countermeasures that can be implemented to prevent brute force attacks but we want a solution that prevents brute force attacks without compromising the availability for users. One idea would be to lock a user account after X amounts to failed login attempts but this will lead to worse availability for a user. For example an attacker can use this to lock out a large number of accounts by brute force attacking multiple accounts on the website [1]. Even if the accounts get unlocked the attacker can just perform the same attack again and lock them out which leads to availability for the user.

A better idea is a CAPTCHA which is a Turing test that can tell humans and computers apart. CAPTCHAs are designed to present tests that are easy for humans to solve but difficult for automated scripts or bots to pass. By requiring human interaction to solve the CAPTCHA, the

system can effectively differentiate between genuine users and malicious automated bots. This does not worsen the availability for the users but is still effective to stop bots and scripts from brute forcing into user accounts. Even if a CAPTCHA is relatively simple and a sophisticated bot might randomly guess the answer, it still introduces a delay for each login attempt. This delay significantly extends the time required for a brute force attack to try all possible combinations of usernames and passwords. In this case the Google reCAPTHA was used which is one of the most used CAPTCHAs and is effective when it comes to preventing bots.

Implementation

To integrate the Google reCAPTHA in the nodebb interface the Spam Be Gone plugin was used and this can be implemented using these steps:

1. Login with an account that has admin access and navigate to the admin dashboard
2. Click on Extend -> Plugins and then scroll though and find Spam Be gone and click on activate
3. Click on Dashboards -> overview and then on that interface click on Rebuild & Restart (This install and activates the plugin)
4. Click on Plugins -> Spam Be Gone -> Google reCAPTCHA
5. Now we need to create a API key so we need to navigate to https://www.google.com/recaptcha/admin/create
6. Create a project and register a new site for the reCAPTCHA (**It is important to choose v2 on the reCAPTCAHA type because that is what nodebb supports**)
7. Copy paste the Site key to Re-Captcha Public API Key and Secret key to Re-Captcha Private API Key
8. Click on Enable Re-Capthca and Enable Re-Capthca on login page as well
9. Save the changes and return to the dashboard overview and click Rebuild & Restart

Now the CAPTCHA should be working and asked for every time a user logs in to the website.

Difficulties

The main difficulty with this task was to choose a countermeasure that is effective when it comes to preventing brute force attacks but still provides good availability for users. As mentioned before there are multiple methods to prevent brute force attacks but you need to find a good balance for the site being secure and still provide a smooth experience for the users.

Own contribution

Tomi Toma

In the beginning of the project when we reached out to each other to try to set up a meeting to start with the tasks, I suggested that we use discord to have a channel where we can easily communicate with each other instead of sending emails. I created a discord channel and sent a link to all group members which we then used to discuss when we should meet and book a meeting with the TA and etc.

On our first meeting we installed the nodebb setup and used the system for a bit to see how it worked and then we also discussed the tasks and selected two of the different options that were available but at that meeting we did not distribute the workload yet. In the one hour supervision with the TA I asked questions about the chosen tasks to get a better overview on what we need to do and what was expected of us.

After the meeting with the TA we split up the work and the idea was that we would work on our part with the implementation and write the report on the parts we worked on. My part of the work was to do the work for the Unauthorized access task. I looked into the task and did a bit more research on how to solve the problem and also had in mind the recommendations of the TA. I decided to implement the google reCAPTCHA as a countermeasure and wrote the report for that task.

I also tried helping the other group members to the best of my ability and when I could for example helping with brainstorming ideas for the other tasks or helping with issues with nodebb or mongodb that someone had an issue with on their computer etc.

Qikun Tian

At the start of the group work, I joined the discord channel created by the group and selected a date for a meeting with the TA and a time for our group to discuss the meeting.

In the first group meeting we worked together to install MongoDB, NodeBB and configure the environment. During the installation we had some difficulties and I offered some help to other group members and got help from other group members. Also we chose two other tasks in this meeting.

After that we made the decision to split the group work in discord, each person is responsible for a different task module, and write a report for the task module he/she is responsible for.The part I am responsible for is the mandatory task database leakage and corruption, before I started I did a lot of searching for information, and accidentally found some information about the Before I started, I did a lot of information searching and accidentally found some information about spam and shared it with the group members, which might be able to help them. One of the solutions I initially conceived was to use Nginx reverse proxy server and set some firewall rules via iptables to restrict remote access, so I installed Ubuntu again on a virtual machine and installed NodeBB,MongoDB and Nginx on Ubuntu. When I tried to implement my initial solution, it never worked and I think I may have missed some important part.

In order to solve this problem, I asked for some help from my group members as well as referring to some advice given by the TA in a video conference, and finally got it done by configuring the file mongod.cfg with bindIp,bind to local loopback address in the file.

Pol Fradera Insa

At the beginning of the project, I made an announcement through our Canvas group along with my teammate Pol Falguera, in order to reach out to the rest of the group members. To facilitate communication, we created a Discord channel, and the first thing we did was meet in person to set up both MongoDB and NodeBB and configure them on Windows. I contributed to the decision-making process regarding the problems to be addressed, and we all reached a consensus.

We had a meeting with our TA where we asked about the aspects of the project that generated the most doubts. Once we had those questions resolved, we started working on our respective tasks.

We divided the work according to topics, and Pol Falguera and I were assigned the task of dealing with the spam problem. First, I gathered information about the nature of the problem, and then I researched possible solutions. I discovered that there was a plugin that could address the issue, so I had to understand the functionality of this plugin, test it and document its operation and functionalities in the report. As for the report, I have written the sections on threats and implementation, and I have also collaborated on other parts. I spent a lot of time testing the different countermeasures due to the issues that arose.

I tried to create a script to post many messages to test the countermeasures, but I encountered a series of errors when attempting to post the messages that I couldn't resolve. Therefore, I had to perform the manual test.

As much as I could, I began resolving my teammates' doubts in the same way they resolved mine, and also offering ideas that could be useful for the project.

Pol Falguera Guillamón

At the beginning of the project, me and my team colleague Pol Fradera wrote an announcement through the group's forum to provide the other team members with our contact details. After that, Tomi Toma reached out and we all joined a Discord group to have an efficient communication.

After that, I suggested booking a supervision meeting with the TA as soon as we could to be able to pick a slot that suited us. After some days, the TA canceled the meeting for personal reasons, so I reached him out to see what we could do. The solution was picking another slot with another TA and so we did.

Once we had our meeting arranged, I suggested meeting in person the whole group so we could meet each other and start working on the project. In the first meeting we all focused on installing NodeBB and MongoDB, and configuring both of them. During the meeting, I focused on the installation steps, asking when errors occurred and trying to solve any doubt my team colleagues had. We discussed which problems to target and, since none of us had dealt with these problems before, I suggested picking the ones that could be more understandable and manageable to us.

During the meeting with the TA, beside the general concerns that Tomi Toma asked, I asked the TA some doubts I had about the project format and what they were expecting us to submit, and also asked for some clarifications of possible solutions to the problems we picked. We also discussed with the TA about them.

After the meeting with the TA, things were a bit clearer for all of us. I suggested splitting the work, so then everyone could schedule their time as it suited them, and then meeting back again to put in common all the research and work we had done. In this meeting we put it all together, showing and explaining our work to our team colleagues, and trying to fix together possible problems that had arisen.

I was in charge of the Spam problem, along with my team colleague Pol Fradera Insa. I first tried to write a script on my own, in order to mitigate the spam in NodeBB, but that was very hard since I found little information on the Internet and also it was quite challenging to me to understand NodeBB code. My colleague Pol found a very good plugin that is used to mitigate spam, so I started researching it and figuring out how to activate it and its features. I wrote its introduction, the countermeasures and some difficulties we faced with my colleague Pol.

Along the project, I did my best to solve my colleagues' questions as they did with me and help them in any way they needed, as well as give suggestions I thought would be useful for the project's workflow.

References

[1] Esheridan, OWASP, *Blocking Brute Force Attacks*, available at: https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks

[2] Google, reCAPTHCA. Avalibale at: https://www.google.com/recaptcha/about/

[3] MalwareBytes, "What is Spam? | Definition & Types of Spam," Malwarebytes, 2022, available at: https://www.malwarebytes.com/spam

[4] "Forum spam," Wikipedia, Nov. 06, 2021, available at: https://en.wikipedia.org/wiki/Forum_spam

[5] "Setting up spam protection for NodeBB", Youtube, available at: https://www.youtube.com/watch?v=Pok3V-nTHww

[6] "Akismet: Spam Protection for WordPress" Akismet, available at: https://akismet.com/

[7] "The Web's Largest Community Tracking Online Fraud & Abuse | Project Honey Pot" Project Honey Pot, available at: https://www.projecthoneypot.org/index.php

[8] "reCAPTCHA", available at: https://www.google.com/recaptcha/about/

[9] "Stop Forum Spam", available at: https://www.stopforumspam.com

[10] "What is hCaptcha?", available at: https://www.hcaptcha.com/what-is-hcaptcha-about