School of Electrical Engineering and Computer Science

Division of Theoretical Computer Science

Cybersecurity Project

DD2394/DD2391 HT22

Fall 2022

Group 7: Carina Alemón, Parosh Shaways, Daniel Tottie

Table of Contents

Problems **3**

Spam 3

Abstract 3

Introduction 3

Countermeasure 3

Installation 4

User Authentication 6

Threats (problem + impact) 6

Countermeasures 6

Implementation & Configuration 7

Google SSO Steps 7

2 Factor Authentication Steps 7

Difficulties 8

Firewall 9

Abstract 9

Introduction 9

Countermeasure 9

Difficulties 10

Own Contribution **11**

Parosh Shaways 11

Carina Alemón 12

Daniel Tottie 13

References **15**

Problems

Spam

Abstract

Spam is a problem for many forums. Spam comes in different forms. In this report we will mention spam related to websites in this case nodeBB. If the website is not protected against spam as soon as possible then nodebb itself will have crucial problems with it.
Different types of spam are unsolicited messages over the internet that can contain everything from advertising, attempted fraud, phishing and computer viruses. Another word for spam is junk mail.

Introduction

There is a major problem in today's society, where almost 85 % of all the mail that is sent in the world is estimated to be spam. That is quite a lot.
The most common spam is unwanted mail containing ads but surprisingly a lot of the mail contains messages designed to trick the receiver of money or information.

It is illegal to send spam to anyone who has not ordered it. Luckily we are in a safe environment and are allowed to test and implement this with nodebb.
The problem we are facing here specifically is that people or bots who register could be registering for spam purposes on nodebb. Then the users could use spam for posts.
The reason why spam is used is mostly because it actually works. Luckily there are countermeasures in this case.

Countermeasure

For nodebb the countermeasure is to use a plugin that will filter out most of the spam created in nodebb.
To combat spams in forums like nodebb we are using a service in a single anti-spam plugin known as Spam-be-Gone.

The reason why this plugin was chosen is because spam-be-gone makes use of three separate services to limit spam. Each of these services requires a one-time set-up to activate.
The first service in spam-be-gone is **Akismet**. It is a spam filtering service that is run by the blogging platform WordPress.
Spam comments is a problem for forums and akismet filters these types of spam. What exactly it does is that it goes through the forum posts/comments, and filters out spam using its algorithms.

This algorithm learns from other websites on what to look out for and the actions those other websites have been taken.
An example is that if other websites start reporting specific posts as spam, then Akismet will know this in the future.
Project Honey Pot is another service in spam-be-gone that is used to help identify individuals and bots who are known to be responsible for high volumes of spam postings.This is done by collecting information about the IP addresses used when for example harvesting email addresses in spam or other forum posts frauds. The third and fourth service used are the **Google Re-Captcha & hCaptcha** that can be used to add an additional layer of spam prevention by setting up a challenge that filters humans from bots basically telling human and bots apart. The technique here is to filter out bots, for example in the registration section where the "I'm not a robot" checkbox is required for a legitimate user to proceed, indicating that they are not a bot. hCaptcha requires website visitors to label images.
All these services can be tested by for example sending spam forum messages and logging in/register in the UI.

Installation

Step 1 install & activate nodebb-plugin-spam-be-gone:

On nodebb after logged in as admin, go to: Admin>Plugins>Install Plugins

Here you search for your specific plugin, in this case nodebb-plugin-spam-be-gone.
Install the plugin, then activate it by clicking "activate".
Once activated go to Dashboards>Overview and on your right press "rebuild & restart".
Once it has loaded, update browser and you should now be ready to setup keys for each service you would like to use.

Step 2 Setting up keys & configure plugins for all services:

From start forum page Go to Admin>PLUGINS>Spam Be Gone

Service 1 Akismet: Follow the link akismet.com to get API key. There you need to register first in order to receive the key. Also here you are able to decide the minimum reputation level to classify flagged posts as false positives (HAM). Posts made by users with at least this level reputation will never be flagged as spam.
Also you can allow users with minimum reputation of X to submit posts to Akismet as spam via flagging (leave blank to disable).

Service 2 Project Honeypot:

Follow link projecthoneypot.org and get Honeypot API key. Register to obtain the key.

Service 3 Google reCAPTCHA:
Here you can sign up if you want at to obtain keys at google.com/recatcha, but if you want for the purpose of testing locally then check out reCAPTCHA documentation:
https://developers.google.com/recaptcha/docs/faq#id-like-to-run-automated-tests-with-recaptcha.-what-should-i-do
and then get these testing keys:
Site key: 6LeIxAcTAAAAAJcZVRqyHh71UMIEGNQ_MxjiZKhI
Secret key: 6LeIxAcTAAAAAGG-vFI1TnRWxMZNFuojJ4WifJWe

The reCAPTCHA widget will show a warning message to claim that it's only for testing purpose. Please do not use these keys for your production traffic.

Service 4: StopForumSpam:

Get keys from stopforumspam.com/keys. Simply add your website and generate the key to add to nodebb.

Service 5: Google hCAPTCHA

Go to https://dashboard.hcaptcha.com and get one site key and one secret key. On the website, Add the sites and hostnames on which to serve hCaptcha.

For some services, in order to get the keys and run the service on localhost you need to change your hosts file. For example, You can do that in /etc/hosts where you add or change for example:
127.0.0.1 test.mydomain.com
Instead of
127.0.0.1 localhost

Step 3 Restart:
Save each change to services. Then go back to Admin>Dashboards>Overview and click “restart” in order for it all to take effect.

User Authentication

Threats (problem + impact)

Authentication is the process in which an entity verifies that someone, in this case, the user is who it claims to be, in order to access a system, device, network or computing resource. This is done with “a human-to-machine transfer of credentials during interactions on a network to confirm a user's authenticity.” (TechTarget, 2021). User authentication “keeps unauthorized users from accessing sensitive information.” (Mayaan, G.D., 2022). This is usually done by providing a password, since the user is supposed to be the only one who knows it, however, cyberattackers might be able to access an account if the user authentication it uses is vulnerable and fails to prevent a breach.

There have been quite some cases in which companies have failed to have secure user authentication and thus have faced attacks, such as the Equifax data breach in 2017, where the attackers “exposed credit card data of more than 147 million consumers.” (Mayaan, G.D., 2022). Another example is the Yahoo data theft in 2013 where all of its 3 billion accounts were hacked, making it the largest breach in history. Without countermeasures users and companies are at risk of being attacked and hackers might be able to access private information and manipulate it as they please.

Countermeasures

In this case two countermeasures were chosen, the first one being SSO (Single Sign On) from Google. It allows a user to use their existing credentials to access a suite of applications, SSO also “reduces the number of attack surfaces because users only log in once each day and only use one set of credentials.” (OneLogin, n.d.). And by using the Google SSO, authentication becomes more secure, given that Google leads the technological field and they state that when using their SSO “users aren't prompted to enter a password when they try to access Google services. Instead, they are redirected to an external identity provider (IdP) to authenticate.” (GoogleCloud, n.d.).

The other chosen countermeasure is 2 factor authentication, this in order to add an extra layer of security and reduce the chances of data breach and attacks. There exists many types of 2FA, such as: code via email, sms, physical tokens, and biometric information, in this case a digital token was chosen which is also provided by Google and it’s called “Authenticator”, which is app that has tokens that provide 6 digit codes which change every few seconds, it uses the “Time-based One-time Password Algorithm and HMAC-based One-time Password Algorithm” (Habets, T., 2021.), verifying the authenticity of users and augmenting security. This 2FA technique was chosen since it uses 2 different algorithms, limiting time and code usage, it also prevents phishing via sms or email given that the code is sent to the user via the verified Google

Authenticator app which is set up by scanning a QR code directly from the website with the user's phone.

By implementing the countermeasures, mentioned above two authentication techniques are used, 'something you know' (the password) and 'something you have' (a code sent to the user's phone).

Implementation & Configuration

Google SSO Steps

Step 1: Access NodeBB with an admin profile

This is necessary in order to install the plug-ins needed for the chosen countermeasures to work.

Step 2: Access Control Panel → Extend → Plugins

Access the section 'Plugins' to find the plugins needed, type 'SSO' on the search bar, once you find the 'nodebb-plugin-sso-google' click 'Install', then go to the 'Installed' tab, find the SSO plugin and click 'Activate'.

Step 3: Rebuild and Restart

Go to the dashboard, Dashboard → Overview and click 'Rebuild & Restart' to ensure that the plugin you just activated works.

Step 4: Configure the plugin in the Google Cloud API Manager

Access the Google plugin configuration, Plugins → Social Authentication: Google, follow the instructions and set the plugin in the Google Cloud API Manager. This in order to connect the plugin with the NodeBB site. ****detailed steps can be found in the repository***

Step 5: Make setup in NodeBB

Go back to NodeBB, copy the 'ClientID' and 'Secret' in the Google API Manager and save the settings. Restart NodeBB once again to make the changes and now the plugin is ready to use!

2 Factor Authentication Steps

Step 1: Access NodeBB with an admin profile

This is necessary in order to install the plug-ins needed for the chosen countermeasures to work.

Step 2: Access Control Panel → Extend → Plugins

Access the section ‘Plugins’ to find the plugins needed, type ‘2fa’ on the search bar, once you find the ‘nodebb-plugin-2factor’ click ‘Install’, then go to the ‘Installed’ tab, find the 2FA plugin and click ‘Activate’.

Step 3: Rebuild and Restart

Go to the dashboard, Dashboard → Overview and click ‘Rebuild & Restart’ to ensure that the plugin you just activated works. And now the plugin is set up, all that is left to do is for the user to enable it.

Step 4: Enable Two-Factor Authentication via Authenticator app

Login or register in NodeBB, the ‘Two-Factor Authentication’ manager should open automatically, if it doesn’t go to Profile → ⋮ → Two-Factor Authentication → Authenticator App → Enable. A QR code will pop up, scan it with Google Authenticator app or with the phone’s camera, the token will now work on your phone and will constantly generate codes. Insert the 6 digit code from the app into the website, it will ask for it twice (it might change due to the Time-based One-time Password Algorithm). Now the plugin is enabled and will be used every time a user tries to access the NodeBB site. Making it more secure!

Difficulties

A main difficulty encountered was getting NodeBB to start and continue to work properly, since it would crash on both Safari and Google Chrome, once the Google SSO plugin was activated. The way this was overcomed was by deleting the cookies, given that the system would saturate and be unable to load the login page. Rebuilding and restarting NodeBB from the Control Panel also seemed to work at times.

Another difficulty was doing the setup in Google’s API Manager, since it was a bit confusing, and the instructions could be broad every so often. This was overcomed by complementing the instructions watching a video tutorial and using the trial and error method.

The last difficulty worth mentioning is the need to make different user accounts to test the user authentication plugins. This more than a difficulty was a burden, thus why a limit and purpose was set for the test accounts.

Firewall

Abstract

Firewalls are key components in any production IT setup. They exist everywhere (even in SAAS solutions), and they are in many ways considered one of the strongest forms of protection against attacks (Digicert, n.d). By limiting the traffic flow you can almost completely eliminate attack vectors that could otherwise be exploited. In IPv4, you can use NAT, port forwarding and access policies to achieve this. Studying the area of firewalls and network best practices led to the decision of limiting access to the admin panel, as well as blocking most of the network flow in and out of the virtual machine.

Introduction

Many times when you install a server software on your computer, or if you use a software that uses network communication in any way, it's very common that you expose ports on your machine that were previously not opened. These ports may accept different forms of input or send various forms of output. Every port that is accessible remotely on your machine is considered an attack vector in the eyes of attackers, and through enumeration and further investigations they may very well find ways to gain access to the computer in ways you had not imagined when you installed the software. Keeping track of all the ports opened on your server, and what they are used for, could be non-trivial in a large setup. Thus is it important to limit who or what can access these ports. Limiting overall network traffic is generally considered the safest route, and by using a firewall you can achieve this.

However, not only do we need to limit access to specific ports, but we may also need to limit access to certain features available on these ports, and in the case of a web service we may want to make sure that attackers can not reach the administrative interface at all (Palo Alto, n.d). This way we don't need to worry about fixing every web vulnerability that could exist or arise in the code. Instead we limit access to it completely.

Countermeasure

To make sure that outside clients can not access the administrative interface of the NodeBB service, we create a proxy configuration to forward requests from port 80 to the default port of nodeBB, only if the requests are not aimed towards the /admin subdirectory of the web server. All request sent to that directory will be denied.

We then create fundamental firewall rules in IPtables to block all incoming and outgoing traffic, which of course means we need to create rules for allowing traffic from localhost to localhost. We block outgoing traffic generally, except when it's sourced from port 80, to block any reverse shells from being established, should there be a vulnerability in the web service accessible from outside.

After these fundamental rules are created, we create another rule to allow traffic on port 80, which is where our nginx proxy is running. If a user now needs to access the administrative interface, they can connect from the local machine to the default port of nodeBB, and thus bypass the proxy. In general we can create a port forward to this port to trusted IP addresses if we want to allow some sort of remote access to the admin interface, but in general all external traffic will go through the nginx proxy.

Installation

Step 1: Install Nginx proxy on the server
Step 2: edit the configuration file to forward traffic from port 80 to the default NodeBB port for every directory.
Step 3: create a specific rule in the nginx config file for the /admin directory to deny all traffic
Step 4: run the iptables commands as specified in our github repository to create the necessary rules described in the introduction part.

Difficulties

The biggest difficulty encountered in this process was realizing that iptables manages internal traffic on the server as well as external traffic. After creating the rules and port forwards, the machine's web service would no longer work. This was because, among other things, NodeBB could no longer connect to the local MongoDB database, because this traffic simply allowed. This was revealed through some basic troubleshooting, and after creating a rule allowing traffic from localhost to localhost, everything worked as expected.

Own Contribution

Parosh Shaways

My contribution to this project has in general been making sure the flow of the project has been in a good manner.
I have made sure to understand each aspect of the course with everything from the rules to reaching out to the teacher for assistance regarding issues or problems for example when a team member was dropped out and if our topics as a group were okay.

I made sure to ask as many questions as possible during the 1 hour consultation to take advantage of the assistant helping us, not only for me but for the group in case they missed asking something or they came up with follow up questions, unfortunately nothing technical.
I have taken care of one problem/issue that we as a group together picked. The problem that was taken care of by me was spam for the forum nodebb.
For nodebb I had to investigate first what exactly spam was in detail, specicially spam towards a forum platform like nodebb. Then I learnt about plugins and that they could be used against such things. I installed the plugin and I made sure I used my email account to obtain each key for the plugins.

As much as I could I also helped with anyone wanting help to my ability. For example helping setting up repository and also helping with any technical issue one would have with virtualbox or the VM the group used.
I have pulled the nodebb and installed mongodb as instructed from the websites which took a day, then investigating the website and how it works too was also a day.

As coming closer to the report deadline I planned to make sure the group would meet up at school or in zoom to go over specific things but not too often so one could work for themselves on their issue.
For the issue and problem I was given, I wrote the section for that report. The whole spam section in the report was written by myself, including the installation part, information of spam and the specific nodebb plugin, and this part of the report was available on google docs for the rest to see and go back too if they needed to check. Also made sure the everything regarding spam was pushed to the repository.

This project has been fun but quite stressful due to not understanding fully in the beginning what to do. Thanks to the individual 1-hour consultation I got enough understanding to do what was needed. This consultation should be earlier in the course for the reason to not feel stressed.

Carina Alemón

At the start of this project I tried to reach out to the other group members and agree on a time and place where to meet. I also suggested using a different platform other than Canvas to ease communication and information flow within the group. Once a platform was chosen I created the group chat and shared the link with everyone in the group.

During our group meetings I made sure to read the guidelines for the project, to corroborate that we were following them correctly, as for the 1 hour supervision, I asked questions about the expected outcome of the project, including the report and presentation, to try and clarify details that weren't explicitly stated in the project's 'structure and requirements', such as the need to prepare slides for the project presentation, as well as the time notice for the topic to present and demos that should be ready for it.

Regarding the implementation, I took charge of everything apropos User Authentication. I chose 2 features to implement, one being SSO (Single Sign-On) from Google, as discussed by the group before, and the other one being 2 Factor Authentication, since after our supervision meeting, the TA suggested implementing more than 1 feature. I chose the type of 2FA considering the options available, their benefits and downsides, as described in the "User Authentication" section of this report.

Finally, for documentation, I wrote the "User Authentication" section of this report, as well as "Own Contribution - Carina Alemón" (this section) and contributed to the writing of the "Group Summary". In addition to that, I did the formatting of the report. In respect of the functionality and detailed implementation guide I wrote the instructions and took screenshots to make it more comprehensible and coherent to the reader, this information can be found in our Git-Hub repository.

Daniel Tottie

My contribution consists of a lot of brainstorming from the get go. After reaching out to group members and establishing a time and place for the meeting, I immediately began brainstorming what we could potentially do for the project, after reading through the guidelines and instructions provided on Canvas.

I made an effort to try to think of solutions and inspire a creative thought process within the group, so that we could collaboratively reach the best solution for each problem, and whenever something was unclear I made my best attempt at finding an answer within the course information or the scheduled supervision.

As we were working on different OS, I suggested we do the work on a virtual machine setup, so we can have similar environments during setup of our different problem solving configurations. I suggested we use Ubuntu as our VM OS, as it has very good firewall and proxy tools ready for use, as well as being the recommended OS for NodeBB.

For the technical aspects of the project, my role eventually became to set up the firewall solution for NodeBB. After studying the best practices regarding internet traffic flow and access rights, I decided to use Iptables on our virtual machine to block incoming traffic, except for the web service port. I also decided to block outgoing requests from the machine to prevent reverse shells from being opened through web vulnerabilities.

Another best practice I applied was to block access to the admin subdirectory. To achieve this, I configured an nginx proxy on port 80, which denies all requests to the admin directory. This way, we could create iptables rules to allow traffic on port 80, which only gives access to the user side of nodeBB. To then access the admin panel, you can connect on the default nodeBB port, which is not open externally. In our setup, this is only allowed from the local machine, but you could make a simple port forward and whitelist any IP address.

I also wrote the documentation and instructions for the firewall/proxy part of the project. As well as uploaded the relevant configuration to our github repository.

Group Summary

The group decided in the beginning to first gather more knowledge about how this should be done. A first initial meeting was set up to download the system and for whoever needed help could ask the team member for it. It was a good way of setting the tone for the group as a whole for further workflow.

The group decided on channels to contact each other for help and meetings, amongst other things, the main one being WhatsApp. On the group's first meeting the problems to be targeted were chosen and doubts about the project were discussed, the main one being the expected outcome and how the group would work with one less team member. The group also began installing the requirements for the program, as well as making a shared GitHub repository and attempting to make NodeBB to work. At this time, the group faced its first problem which was the installation of NodeBB, it took a long time and many errors were encountered.

Later, the group experimented with the possibility of working on a virtual machine to synchronize work and ease the setup. Some group members managed to make it work, however, another team member could not get it to work and even its virtual machine stopped working altogether. Due to this and practicality the team decided to divide the labor, given that there were 3 problems chosen and 3 team members, each member chose a problem to allocate. Each member was in charge of the background research, implementation, report section and documentation concerning the problem they chose. In addition to that all group members has to collaborate writing the group summary.

The group scheduled a 1 hour supervision which indeed was helpful. There the group managed to get specific information about certain problems and what countermeasures to use for NodeBB. Initially the group wanted to add the problem of ddos attacks but thanks to the supervision the group understood that it was not for this course in the sense of how much work would actually go into it. The group settled for a new problem which was changed to spam.

As far as time planned, the group followed the time allocation guideline from the courses canvas page. The actual time invested was very close to that time allocation but due to some technical problems it had to go over time a bit. Going over nodebb system took most time since one would have to understand the system before actually getting into it.
Overall this was a fun project to understand how to make a system or website more secure.

References

Godwin, N. (2020). *4 User Authentication Issues Developers and Admins Struggle With (Solved)*. SMSEagle. Available at: https://www.smseagle.eu/2020/01/27/4-user-authentication-issues-developers-and-admins-struggle-with-solved/

Google Authenticator. (n.d.) *Google Authenticator.* Google. Available at: https://googleauthenticator.net

Google Cloud. (n.d.) *Single sign-on.* Google. Available at: https://cloud.google.com/architecture/identity/single-sign-on

Habets, T. (2020). *Google Authenticator OpenSource.* GitHub. Available at: https://github.com/google/google-authenticator

Mayaan, G.D. (2022). *5 User Authentication Methods that Can Prevent the Next Breach.* ID R&D. Available at: https://www.idrnd.ai/5-authentication-methods-that-can-prevent-the-next-breach/

OneLogin. (n.d.). *Why is single sign-on important?*. OneLogin. Available at: https://www.onelogin.com/learn/why-sso-important

Soni, R. (n.d.). *7 Benefits of Single Sign-On (SSO) and Why Your Business Needs It.* loginradius. Available at: https://www.loginradius.com/blog/identity/benefits-single-sign-on-sso/

TechTarget. (2021). *user authentication.* TechTarget. Available at: https://www.techtarget.com/searchsecurity/definition/user-authentication

digicert. (n.d). *Importance of using firewalls.* DigiCert. Available at: https://www.websecurity.digicert.com/security-topics/importance-using-firewall-threat-protection

Palo Alto. (n.d). *Best practices for securing administrative access.* Palo Alto. Available at: https://docs.paloaltonetworks.com/pan-os/9-1/pan-os-admin/getting-started/best-practices-for-securing-administrative-access