Mastodon

Setup OIDC for Guacamole

https://www.youtube.com/embed/WIyAGuMbjmY

Setting up a Domain Name and Reverse Proxy

I use NGinX Proxy Manager as my reverse proxy of choice. Feel free to use any reverse proxy you like, but my expectation is that you'll know how to configure it to match my settings as needed.

Open NGinX Proxy Manager, click on the Proxy Hosts option, then select ‘Add New Host’ from the top.

In the pop-up modal, enter the domain or sub-domain you want to use for your guacamole install.

> NOTE: You need to have ports 80 and 443 forwarded through your firewall to the host running NGinX Proxy Manger.

After typing out your domain name, press Enter or Tab to make sure it turns into a “chip” entry. Next move to the IP Address field, and enter the LAN IP address of your server running Guacamole. If you are running Guacamole on the same server as NGinX Proxy Manager, you can enter “localhost” in this field.

If you are running both Guacamole and NPM in the same docker network (which cannot be the default docker network) then you can also enter the Guacamole container name. You can find this by typing:

```
docker ps
```

in the terminal, and looking for the name entry.

I run Guacamole and NPM on different physical servers, so I'll be using the IP address of my Guacamole host server in the IP field.

Next, enter the port you used in the port mapping of your docker-compose.yml file on the left side. If you didn't change it, this port number will be 8080.

Now enable the options for ‘Block Common Exploits’ and ‘Websockets Support’.

Move to the ‘SSL’ tab, and select “Request a new certificate” from the drop-down menu. Next, enable the options for ‘Force SSL’ and ‘HTTPS/2 Support’. Make sure your email is filled in, and enable the option to accept the LetsEncrypt Terms of Service.

Finally, click ‘Save’ and wait patiently. If all is setup properly, you'll be issued a LetsEncrypt certificate for your domain, and you'll be able to now access your Guacamole instance using the domain name you just setup. You should also be accessing it over SSL encrypted HTTPS. Awesome!

If you don't intend to use an OpenID Connect server for authentication, then you are set, and ready to start creating connections to any machines you want to access remotely.

Setup OIDC (OpenID Connect) for Your Guacamole Install

If you happen to run, or are thinking of starting to run your own authentication system, then being able to login with SSO becomes a huge time-saver, and blissful gift to your mind that's overburdened with tens or hundreds of passwords.

Today, I'll show you how to add OIDC to the Guacamole install we've just done. None-the-less, you should still go through the steps above first, and make a new administrative user account with the Guacamole login that is not the default set of credentials, and remove the default credentials as a solid measure of security.

When you've done that, and you are ready to move forward with OIDC, then read on.

Adding OIDC to the Guacamole Docker Container

First, we need to adjust our Guacamole container so it will include the necessary extension for OIDC to work for us.

After that we'll add some configuration values that are also necessary for Guacamole to recognize the OIDC Authentication server, and for that server to know where to send us to after a successful login.

First, we'll adjust our docker-compose.yml file. Make sure you are in the directory on the server where you put the Guacamole docker-compose.yml file, and open the file in a text editor. I use nano because it comes with most distros built in, but feel free to use VI, VIM, Emacs, or any other text editor you prefer.

`nano docker-compose.yml`

Now, move down below the ‘ports’ entries, and create a new section at the same level called ‘environment:’, then below that indent two spaces, and add the following:

`- EXTENSIONS=auth-sso-openid`

Once complete the full docker-compose.yml should look like the following:

```
version: "3"
services:
  guacamole:
    image: jwetzell/guacamole
    container_name: guacamole
    volumes:
      - ./postgres:/config
    ports:
      - 8080:8080
    environment:
      - EXTENSIONS=auth-sso-openid
    restart: unless-stopped
volumes:
  postgres:
    driver: local
```

Next, we need to adjust our Guacamole properties file. To do this, we'll edit the file located in the path ./postgres/guacamole/guacamole.propertiees.

If you change the volume mapping from what i have above, you'll need to find this file wherever you mapped the volume on your system.

```
nano postgres/guacamole/guacamole.properties
```

> NOTE: You may need to use sudo nano postgres/guacamole/guacamole.properties if you have not set the folder permissions for this folder to be owned by your user.

Inside this file, you'll likely see some settings already configured. Do not change these settings. All we need to do is add a few settings below them. The settings we want to add below them are shown below.

Feel free to copy and paste, but keep in mind you'll need to replace my place-holder values with the real values from your authentication system. I am using Authentik currently, and these values are made very easy to locate in the Provider I created for Guacamole.

An Aside for those using Authentik

If you are using Authentik, and aren't sure how to setup an OIDC provider, fear not, it's pretty straight forward.

Navigate to your Administrator settings area.

1. On the left, expand the Applications section, and choose the Providers option.
2. Create a new Provider.
3. Select the OpenID Connect (OIDC) Provider type, and click ‘Next’.
4. Give the Provider a name that makes sense for this application (e.g. Guacamole-OIDC).
5. Choose the default Authentication flow (unless you've setup a custom one, then feel free to use that).
6. Choose the default Authorization flow (explicit).
7. Leave it as ‘Confidential’.
8. In the Redirect URIs/Origin field, enter the URL of your Guacamole server (e.g. https://guac.mygreatdomain.org)
9. Click ‘Finish’.

1. Now click on the Application option on the left navigation menu.
2. Click ‘Create’.
3. Enter a Name for the application (e.g. Guacamole). Note the slug will auto populate based on what you enter in the Name field.
4. If you use permission groups for your applications, add the groups that can access the Guacamole server.
5. In the Provider drop-down select your freshly created Guacamole-OIDC provider.
6. Click Create.

Now you'll have all the values you need in order to get this running.

Get the Values

Note that for the `openid-group-claims-type` value, I entered ‘admins’. This is the only group I have in Guacamole. For your server you may want to set a lower level group, but this works for me as the only user on Guacamole.

```
openid-authorization-endpoint: https://auth.mygreatdomain.org/application/o/authorize/
openid-jwks-endpoint: https://auth.mygreatdomain.org/application/o/guacamole/jwks/
openid-issuer: https://auth.mygreatdomain.org/application/o/guacamole/
openid-client-id: some-super-Long-5trinG-0f-Ch4rac7ers
openid-redirect-uri: https://guac.mygreatdomain.org
openid-groups-claim-type: admins
extension-priority: openid
```

When you've added all of your values, you should have a file that looks like this:

```
postgresql-hostname: localhost
postgresql-port: 5432
postgresql-database: guacamole_db
postgresql-username: somegreatuserintheether
postgresql-password: an-awesome-strong-long-complex-crazy-but-memorable password

# ldap-hostname: ldap.example.net
# ldap-port: 389
```

```
# ldap-encryption-method: none
# ldap-max-search-results: 1000
# ldap-search-bind-dn:
# ldap-search-bind-password:
# ldap-user-base-dn: ou=people,dc=example,dc=net
# ldap-username-attribute: uid
# ldap-user-search-filter: (objectClass=*)

openid-authorization-endpoint: https://auth.mygreatdomain.org/application/o/authorize/
openid-jwks-endpoint: https://auth.mygreatdomain.org/application/o/guacamole/jwks/
openid-issuer: https://auth.mygreatdomain.org/application/o/guacamole/
openid-client-id: some-super-Long-5trinG-0f-Ch4rac7ers
openid-redirect-uri: https://guac.mygreatdomain.org
openid-groups-claim-type: admins
extension-priority: openid

enable-clipboard-integration: true
```

> NOTE: The LDAP information was already there in my file, so I left it. It was also already commented out, so there is no harm in leaving it.

Once you've added all of the values from your OIDC provider, save your changes with CTRL + O, then press Enter to confirm. Next, press CTRL + X to exit the nano editor.

Now, you need to go back to your docker/guacamole folder, and run the command:

`docker compose down`

After it comes down completely, bring it back up, and watch the log output as it starts.

`docker compose up -d && docker compose logs -f`

> NOTE: You can probably just do docker compose restart instead of the down and back up again, I just do it this wayto make sure it completely restarts.

This time, make sure you aren't seeing any errors in the log as it scrolls. If you have any values set that the properties file and extension can't read, you'll likely get errors and possibly a message

about an exit code.

If you see this, stop the log output with CTRL + C, then bring down the container with:

```
docker compose down
```

Next, check all of your entries in the properties file again. Make sure you didn't add any extra spaces, quotation marks, or other characters, and ensure all the values are correct.

Also check the docker-compose and make sure you've got it all aligned properly.

Then bring your container back up with

```
docker compose up -d && docker compose logs -f
```

When successful, you should see no errors or exit code messages (possibly exit code 0), but not on this container as far as I recall.

Once you feel it's up and running properly, navigate to your FQDN (https://guac.mygreatdomain.org) and you should be presented with your Authentication system login prompt.

> NOTE: On Firefox, because of the caching it does, I initially got into a login loop, but logging into a private window or separate browser will usually get you in with no issues. After closing fFrefox completely, and then logging in, the login loop no longer occurred.

Creating a Guacamole Admin User for OIDC

If you've already brought up OIDC and tried to login, you may notice that you have a new user created vs. logging in as an admin user. This is because Guacamole creates a new user if it can't find a user to match on. Let's set up an admin user to match.

First to

```
docker compose down
```

Then go into the guacamole.properties file, and change the extension-priority value to be like this:

`extension-priority: *, openid`

Now bring Gucamole back up with

`docker compose up -d`

Log into Guacamole with the original admin user you created.

It's important to realize that Guacamole will use the username field for your OIDC user matching. You need to use the email address coming from your OIDC provider for Guacamole to match on.

If you don't create an admin user for your OIDC to match to, you'll have a hard time with Guacamole, so let's create one.

1. Go into Settings in Guacamole
2. Click on the 'Users' tab.
3. Create New User
4. In the username filed, enter the email address of the admin user that will be coming from your OIDC provider.
5. IMPORTANT! Do not enter anything in the password fields. In fact, do not tab or click into the password fields. If you enter this field, Guacamole will see it as "dirtied" and will not let you save the user with a blank password.
6. Go through the rest of the fields, and assign your admin user all permissions, and access to all Connections already created.
7. You can additionally add them to any admin groups you may have already created.
8. Click 'Save'.

One last time, you'll do

`docker compose down`

Wait for Guacamole to stop.

Change the Login to be "openid" only in the configuration file, then restart Guacamole with

`docker compose up -d && docker compose logs -f`

Revision #2
Created 11 November 2023 15:40:24 by Brian McGonagill

Updated 11 November 2023 16:18:31 by Brian McGonagill