GSOC ‘18 PROJECT PROPOSAL

ORGANIZATION: OpenWISP

OpenWISP 2 RADIUS

BASIC INFORMATION

Name and contact information:

- **Name:** Rohith A. S. R. K.
- **Email:** rohith.asrk@gmail.com
- **IRC/Gitter/Github:** rohithasrk
- **Phone number:** +91 9100149957
- **Country/ Region:** Roorkee, India

University and Current Enrollment:

- **University:** Indian Institute of Technology, Roorkee
- **Field of Study:** Electronics and Communication Engineering (Batch of 2019)

ABOUT ME

I am a third year undergraduate student studying Electronics and Communication Engineering at IIT Roorkee. I’m passionate about web, networking and software development. I got introduced to software development in my freshman year and since then I’ve been trying out new technologies and contributing a variety of projects. I, being a part of the Information Managment Group develop and manage the official intranet portal of the college which serves around 8000 students. It is highly scalable and written in latest technologies like Django, django-rest-framework, a whole lot of other Python packages, Angular, and React.

I’ve successfully completed Google Summer of Code 2017 with OpenWISP and have been an active part of the community since then. I maintain django-netjsongraph, openwisp-utils, openwisp-network-topology, and netdiff. I’ve been a very active Google Code In 2017 - 18 mentor for OpenWISP. I have all the resources that are essential for the project. My motivation for GSoC this time is working on challenging projects that would enhance my skills and also be of great use to the network community. I strongly believe in this project that I have chosen to create huge impact over a large group of people. Apart from that, I’d also be helping the other GSoC students which would improve the grounds for being a mentor in the coming years.

PRE-GSOC INVOLVEMENTS

Here is a list of PRs/ discussions made so far in django-freeradius.

- **#110 (open)** - A working implementation of enforcing session/ bandwidth limits on users.
- **#109 (open)** - Batch add users feature.

- **#108 (issue)** - Add more attributes to the RadCheck model to impose more restrictions.
- **#106 (open)** - Altered migration files to add support to freeradius3.
- **#105 (merged)** - Improved docs for configuring freeradius dev environment.
- **#104 (open)** - Removed repetition of id declaration.
- **#97 (issue)** - Import shared code from openwisp-utils
- **#96 (issue)** - Add openwisp-admin theme to tests/ django-freeradius
- **#94 (issue)** - Add support to django > 2.0 to requirements-test.txt

Other major contributions to OpenWISP can be checked out at GSoC 2017 with OpenWISP. Statistically, they could be summarized as

- 43 commits (4288++, 3039- -) in netjson/django-netjsongraph
- 44 commits (2633++, 1217- -) in openwisp/openwisp-network-topology
- 8 commits (1396++, 32- -) in openwisp/openwisp-utils
- 10 commits (230++, 35- -) in ninuxorg/netdiff
- 3 commits (114++, 5- -) in openwisp/ansible-openwisp2
- 1 commit (49++, 1- -) in gregmuellegger/django-sortedm2m
- 1 commit (15++, 0- -) in openwisp/openwisp-users
- 2 commits (6++, 1- -) in openwisp/django-netjsonconfig

PROJECT DETAILS

Introduction

One of the features missing in the OpenWISP modules is AAA (Authorization, Authentication and Accounting). **Remote Authentication Dial-In User Service** (**RADIUS**) is a networking protocol that provides centralized Authentication, Authorization, and Accounting management for users who connect and use a network service. FreeRADIUS is an open source and a very popularly used RADIUS service provider.

django-freeradius is a modular python package developed for managing the FreeRADIUS databases from a unified django admin interface and providing APIs for Authorization, Accounting and Logs. **openwisp-radius** would be a wrapper of django-freeradius which adds multi tenancy on top of it. This project would involve adding all the additional core features to django-freeradius and then pulling them in openwisp-radius.

Features

- A module of OpenWISP 2 which could be used by organizations for managing their FreeRADIUS services from a easy to use django's admin interface.
- Easy management of RADIUS users and groups.
- Enforce session limits and daily bandwidth limits (which every ISP wants to do)
- Secure API with many restrictions on the API using techniques like token authentication, ip specifications etc.

Possible mentors - Federico Capoano

MEASURABLE OUTCOMES

- Enforce session limits using radius groups in django-freeradius.
- Import users from csv via a management command and also the admin interface.
- Add optional ways to restrict API usage for security reasons.
- Integrate django-freeradius in the rest of the openwisp2 ecosystem by creating a new wrapper named openwisp-radius and adding multi tenancy on top of it.
- Add a way to easily define additional fields that user may need by using a Profile model.
- Integrate openwisp-radius in ansible-openwisp2 as an optional module.
- Keep the coverage of django-freeradius 100% and documentation up to date.
- Achieve test coverage of openwisp-radius of 100% and documentation inline with other OpenWISP modules.

ACHIEVING PROJECT GOALS

The entire project can be broadly classified into four phases through completion:

Phase 1: Deliverable before the first Mid-Term evaluation.

- **1.1 Add a way to enforce session limits using radius groups.**
- **1.2 Add a way to import users from CSV, first via a management command, then via the django admin.**
- **1.3 Add optional ways to restrict API usage for security reasons.**
- **1.4 Maintain the coverage of django-freeradius at 100%**
- **1.5 Release django-freeradius 0.1**

Phase 2: Deliverable before the second mid-term evaluation.

- **2.1 Initialize openwisp-radius and prepare a setup.py script that will be used to install the python package of the reusable django app.**
- **2.2 Pull in django-freeradius and add multi tenancy on top of it.**
- **2.3 Add a way to define additional fields that user may need by using a Profile model.**
- **2.4 Release openwisp-radius 0.1 and fix issues upstream.**

Phase 3: Deliverable before the final evaluation.

- **3.1 Integrate openwisp-radius in ansible-openwisp2 as an optional module. It may need the implementation to be done in a sub role that becomes a dependency of ansible-openwisp2.**
- **3.2 Provide documentation using python-sphinx.**
- **3.3 Release ansible-openwisp2 and fix issues if any.**

Wishlist: Can be implemented if all the above tasks are completed successfully and the programme is yet to end, also desired to be taken up post GSoC.

- **Add additional features in django-freeradius and pull them in openwisp-radius. Depending on the feature-requests and suggestions of the openwisp2 users.**

DETAILED EXPLANATION

Phase 1

1.1 **Add a way to enforce session limits using radius groups**

Enforcing session limits could is vital to any ISP, big or small. This feature shall be added to django-freeradius by using the concepts of Radius groups.

- Create a new model by the name RadiusLimit which inherits itself from `AbstractRadiusLimit`.
- Have fields like groupname, daily_session_limit, daily_bandwidth_limit, max_all_time_limit in `RadiusLimit` which would make it easy to enforce from the admin interface.
- Override the save method of `RadiusLimit`, so as to create `RadiusGroupCheck` instances and enforce session/ bandwidth limits.

For enforcing session limits, we'll have to configure our FreeRADIUS server to use module **rlm_sqlcounter** and use attributes like

- **'Max-All-Session-Time'** - the all time maximum session limits.
- **'Max-Daily-Session'** - daily session time limits.

More details about the module configuration could be seen from here.

For enforcing limits on the bandwidth usage, we have to configure our FreeRADIUS server to use the module **rlm_sqlcounter** for calculating the max data usage as shown here. By defining the custom modules we would be able to use attributes like

- **'Max-Daily-Octets'** - max bytes usage daily
- **'Max-Weekly-Octets'** - max bytes usage per week

And many more.

An alternative way of doing it is replicating the logic of sqlcounter using django so that the user need not do any extra configuration in the FreeRADIUS files.

A basic working prototype can be seen at **openwisp/django-freeradius#110**

1.2 Add a way to import users from CSV, first via a management command, then via the django admin.

- Given a csv file with the usernames and passwords, we should be able to import users and provide authorization till a predefined expiration date.
- Additional features include retrieving all the users whenever possible and also the ability to delete all of them.
- To achieve this a new model by the name `'RadiusBatch'` could be defined which inherits itself from the `'AbstractRadiusBatch'`.
- The model shall have many-to-many relations with the default user model (for rest authorization) and the `'RadiusCheck'` model (for freeradius authorization).
- Overriding the delete function could help us deleting all the users/ radcheck elements once the batch is deleted.

```
class AbstractRadiusBatch(TimeStampedEditableModel):
    users = models.ManyToManyField(get_user_model())
    radcheck = models.ManyToManyField(swapper.get_model_name(
                              ‘django_freeradius', 'RadiusCheck'))
    expiration_date = models.DateTimeField(
                                 verbose_name=_('expiration time'),
                                null=True, blank=True)
```

An implementation of this feature can be seen at **openwisp/django-freeradius#109**.

1.3 Add optional ways to restrict API usage for security reasons.
It is necessary for making the authorization APIs more secure so that they don’t get exploited. So adding different authentication techniques like token based authentication, restricting the IP ranges should help. There are different types of token authentication that could be implemented, like **django-rest-framework-jwt**, **django-rest-knox** or the standard **token-authentication** inbuilt in the rest framework. In general, this would involve generating a token, associating it with a user and implementing in an API. When integrating into OpenWISP2, different organizations could have different tokens. Configure tokens with FreeRADIUS files as well.

1.4 Maintain the coverage of django-freeradius at 100%
This would involve writing unit-tests for all the added features and make sure all of them pass. Test driven development could be a key to achieving this with ease.

1.5 Release django-freeradius 0.1
Make sure all the features work and make a public announcement about this.

Phase 2

2.1 Initialize openwisp-radius and prepare a setup.py script that will be used to install the python package of the reusable django app.
- This would involve creating a repository named openwisp-radius and preparing all the necessary scripts needed for a PyPi package.
- This would be similar to the work done in the initial commit of openwisp-network-topology.

2.2 Pull in django-freeradius and add multi tenancy on top of it.
- Multi tenancy is having a single instance which runs of server and serves multiple tenants. openwisp-users is a package which is used to implement multi-tenancy in all the openwisp-modules.
- The AUTH_USER_MODEL would now be openwisp_users.User and an organization field would be added to each and every model of openwisp-radius while pulling in django-freeradius.
- There exists a OrgMixin in openwisp_users which upon inheritance adds up a `organization` field to all the models.
- For implementing multi tenancy in the admin interface, the reusable mixins from openwisp-utils shall be used. These are **MultitenantAdminMixin, MultitenantAdminMixin, MultitenantRelatedOrgFilter**.

2.3 Add a way to define additional fields that user may need by using a Profile model.

- A model by the name `‘UserProfile’` can be defined in openwisp-users which will contain all the additional information like social security number, badge ID, public keys, country identities etc.
- This shall be made reusable so that it can be extended to all the openwisp modules.
- Default fields shall be finalized after a thorough research. All other fields can be added based on the users’ requirements.

2.4 Release openwisp-radius 0.1 and fix issues upstream.

- Wrap up the initial version and release it fixing any issues found on testing.

Phase 3

3.1 Integrate openwisp-radius in ansible-openwisp2 as an optional module.

- It may need the implementation to be done in a sub role that becomes a dependency of ansible-openwisp2.
- We’ll need to make the openwisp-radius module and openwisp-controller both optional modules in ansible-openwisp2.
- This can be implemented by defining role variables for each of openwisp-controller and openwisp-radius and ask users to configure for whatever they need.
- This might need some modification in the present working of ansible-openwisp2.

3.2 Provide documentation using python-sphinx

- Documentation is really vital, especially for this project as it involves a lot of configuration in the FreeRADIUS server.
- Sphinx is a great tool and renders beautiful documentation with a variety of output formats.

3.3 Release ansible-openwisp2 and fix issues if any.

- Make a public announcement about the new addition of the openwisp-radius module.
- Fix issues that arise in the production and ensure the project works well.

PROPOSED TIMELINE

2 Apr - 8 Apr	Merge pending PRs, #104, #106, and #109. Finish the beginners guide of FreeRADIUS.
8 Apr - 15 Apr	
16 Apr - May 6	Inactivity due to university examinations and academic work.
May 7 - May 13	Develop more on #110 and get it merged. Document the added features and the FreeRADIUS configuration section of django-radius documentation.
May 14 - May 20	
May 21 - May 27	Implement the token authentication feature deciding the most appropriate token system to be used.
May 28 - Jun 3	

Jun 4 - Jun 10	Document all the added features and release django-freeradius 0.1.
Jun 11 - Jun 17	Fix if any issues arise after the 0.1 release.
Phase 1 evaluation	
Jun 18 - Jun 24	Initialize openwisp-radius, prepare a setup.py script and make a prospective PyPI package.
Jun 25 - Jul 1	Pull in django-freeradius into openwisp-radius and add multi tenancy on top of it.
Jul 2 - Jul 8	Add a Profile model for easy addition of extra information.
Jul 9 - Jul 15	Release openwisp-radius 0.1 and fix issues upstream
Phase 2 evaluation	
Jul 16 - Jul 22	Integrate openwisp-radius as an optional module with rest of openwisp modules. Modify ansible-openwisp2 to optionally include openwisp-controller.
Jul 23 - Jul 29	
Jul 30 - Aug 5	Release ansible-openwisp2 and fix issues/ add features requested by the users. Conclude GSoC by making final additions/ fixes.
Aug 6 - Aug 12	
Phase 3 evaluation - Conclusion	

Availability

My vacations start from 7 May and end on 15 July. The official GSoC period is from 14 May to 14 August. I can easily devote 40-50 hours a week till my college reopens and 30-40 hours per week after that. I'm free on weekends and mostly free on Wednesdays. I intend to complete most of the work before my college reopens.
Other than this project, I have no commitments/ vacations planned for the summer. I shall keep my status posted to all the community members on a weekly basis and maintain transparency in the project.

AFTER GSoC

> Are you interested in working with OpenWISP after the GSoC ends?

Yes. It has almost been an year I've been contributing to OpenWISP. It has been a great ride. I've learnt a lot of skills over this time and believe that I can now independently put it to the real world use. I aim to research more about wireless networking and feel like OpenWISP has a lot to give. Apart from that, the best part of working with OpenWISP is the goals of the projects and the mentorship of it's developers. With the community growing continuously, I feel responsible for all the projects I'm a part of. I've been one of the most active members on the chatroom and have been contributing in almost all possible ways.

> If we get new business opportunities to build new features, would you be interested in occasional freelance paid work?

Yes, if I feel it would be beneficial to both the community and myself. There is always learning involved in working on projects with deadlines.

REFERENCES

1. OpenWISP GSoC '18 Ideas page
2. Documentation of django-freeradius
3. FreeRADIUS Beginner's guide

****