GSoC ‘17 Project Proposal

Organization: OpenWISP

Django-Radius: Web interface for FreeRADIUS

Basic Information

Name and contact information:

- **Name:** Rohith A. S. R. K.
- **Email:** rohith.asrk@gmail.com
- **IRC Nickname:** rohithasrk
- **Phone number:** +91 9100149957
- **Github:** rohithasrk
- **Skype:** live:5dae8a13c338f257
- **Video Chat:** https://appear.in/rohithasrk

University and Current Enrollment:

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

Meeting with mentors:

- Reachable anytime easily through email or IRC. A well planned session if required:
- Mondays, Wednesday, Fridays between 2:00 pm - 12:00 am IST and any day from 6:00 pm - 12:00 am IST.

Coding Skills

I am very comfortable with all the technologies that are to be used in this project. I can write (in order of proficiency):

- Python: Programming in python, Django web framework.
- JavaScript: JQuery, AngularJS, Vanilla JS.
- Java/ C++ (Can revise if needed).

Development Environment

- Ubuntu 14.04 LTS, Python 2.7 installed.
- Highly proficient in using Vim, Emacs and Sublime text.

- Good familiarity with virtualenv/ vagrant.

Version Control

- Very strong concepts of Git. Used Github and Gitlab a lot.
- Also comfortable with Mercurial and SVN.

About Me:

I am a 19 year old, sophomore currently enrolled in Electronics and Communication engineering at IIT Roorkee. I have developed a passion for programming and web development in my freshmen year and from then, most of my time goes into reading and writing software. I have been contributing to open source regularly since about six months now - looking over to repositories of the products I use/ come across, trying to give my part of contribution back to the organization whose product has been an asset to me.

I have the experience of working closely with a team as I am an active member of **Information management Group** at IIT Roorkee, a bunch of passionate enthusiasts who manage the **institute main website**, internet and intranet activities of the university.

The biggest project that I've worked on is LecTut. LecTut is an intranet study portal of IIT Roorkee where professors and students can upload and share the course material with everyone. My work involved managing the platform and working on the backend infrastructure that runs all the services. This included developing APIs using Django and Django REST Framework, implementing elasticsearch using haystack and coding the frontend in AngularJS. A lot of challenges were solved in implementing this. My role was to understand the needs of the system, design the infrastructure from scratch, implement it and make a tool to automate the batch addition process every year.

This is the first time I might contribute to a big open source project.

Pre-GSoC Involvements

Here is a list of issues/ PRs I've worked on so far in **OpenWISP** or **related** projects:

- **#32 (merged):** Made the hint text bolder
- **#35** (**merged**): Fixed a grammar error in models.py
- **#101(mergeable):** Improved string representation of sortedm2m relationships
- **#100(mergeable):** Added swp files to gitignore
- **#102:** Opened up an issue and suggested a way to fix PEP8 style errors

- **#36:** Opened up an issue to suggest setup virtual environments using Vagrant.
- **#18:** Fix all style errors using automated tools

Other than OpenWISP, I've contributed in other organisations too. These are the list of contributions in them.

Zulip: An open source chat application (Django, JavaScript)
- **#3153 (merged):** This PR involved migrating the use to send_mail function from EmailMultiAlternatives
- **#3416 (merged):** This PR involved setting up better file naming conventions in Zulip and make necessary changes.

InfosecIITR: The Information security organisation of the university
- **#3 & #8 (both merged):** Added security tools for web pentesting.

Previous experiences with OpenWRT:

We are given LAN cables in each and every room of hostel in IIT Roorkee. Accessing the internet requires **802.1X** authentication from the client. I tried setting up a router in my room and I couldn't access the internet because the software in my router didn't support **802.1X auth.** I've then set up OpenWRT in my router and it helped me solve the problem. This got me interested and I started exploring the field of networking.
> Do you have a router at home on which you can flash OpenWRT in order to test OpenWISP?
Yes, I have a router which is OpenWRT compatible on which I plan to do my testing.

Motivation

My motivation for GSoC this year is getting started with the open source community. GSoC is a great program for introducing organizations with prospective contributors and when I saw this project, I thought this is something that aligns with my area of interest.
Networking has always been my field of interest and I'm naturally inclined towards contributing to the project. I started to interact with the community and got more and more support from the prospective mentors. And that is when I gained confidence about the project. The thought of me being able to contribute to something big and in turn learn a lot motivates me to work with OpenWISP for the SoC. I strongly believe that with my skills and the mentorship, I'll be able to contribute well to the organisation.

Project Details

Django Radius: A web interface for FreeRADIUS

Introduction

Remote Authentication Dial-In User Service (**RADIUS**) is a networking protocol that provides centralized Authentication, Authorization, and Accounting (AAA or Triple A) management for users who connect and use a network service. FreeRADIUS includes a RADIUS server, a BSD licensed client library, a PAM library and an Apache module.

This is one of the most important features missing in OpenWISP 2. This project aims to create a web interface to manage a freeradius database, with additional RESTful APIs that will be used to retrieve session data from freeradius and to allow new users to register via third party apps.

Features

- Manage freeradius database tables via django-admin
- Facilitates adding users in batches and store all the batch data in the database
- Generate usernames and passwords and renders a PDF which contains the list of all usernames and passwords for individual users or a batch.
- This could be used in events and conferences where access to network is necessary and the credentials shall expire in some given amount of time.
- RESTful APIs through which authorized users or superadmins in general can retrieve radius sessions. This feature can be customized by users to limit or extend administrative privileges.
- Customizable RADIUS attributes in admin panel which dynamically create tables in the database.
- An extensible signup API.

Possible mentor - Lacopo Spalletti and Federico Capoano

Measurable Outcomes

The main measurable outcomes of the project are as follows:

- Implement a reusable django app which allows to manage the main freeradius database tables (session/accounting, check, group, nas, reply) via the django-admin.

- Model references to `User` must be implemented using the swappable user model mechanism of django and default to `django.contrib.auth.model.User.`
- Implement a “Batch add users” feature, each batch operation and it’s details shall be saved to database.
- Implement a RESTful API through which authorized users will be able to retrieve radius sessions, API implemented using the Django REST framework.
- Achieve a test coverage higher than 80%.
- Provide a documentation using python-sphinx, the documentation must be included in docs/ directory in repository.

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 Prepare a setup.py script that will be used to install the python package of the reusable django app.**
- **1.2 Writing tests for all the functions written till now. I plan to learn about test driven development and follow.**
- **1.3 Creating abstract models in django to manage the main freeradius database tables.**
- **1.4 Create a `RadiusUser` model and use swappable user model mechanism to use a one to one relationship with `User`.**

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

- **2.1 Implement ‘batch add users’ feature saving details to database.**
- **2.2 Implement a RESTful API through which authorized users will be able to retrieve radius sessions.**
- **2.3 Extensible SignUp API**

Phase 3: Deliverable before the final evaluation.

- **3.1 Custom RADIUS attributes options being made available in the admin panel.**
- **3.2 Documentation using python-sphinx.**

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.

- **4.1 Improve UI/ UX of the app**
- **4.2 Improve test coverage upto maximum.**

Phase 1

1.1 Prepare a setup.py script that will be used to install the python package of the reusable django app.

- This would ease the setup work for both the user and a developer.
- There are some specific set of rules that are to be followed in order to accomplish this and is given in the django docs which I tend to follow.
- Can be found **here.**

1.2 Write texts to maximise the test coverage.

- Writing tests is an essential part of building an application. It not only allows easy debugging while development, but also in production or whenever any new feature is introduced.
- Will be using the django.test module, i.e., the unittest framework of python and write tests, following proper guidelines. I think the best way of writing tests is before writing a particular view/ function and will implement this way.

1.3 Create abstract models to manage the freeradius database tables.

- This is the core feature of the project, being able to manage freeradius database tables.
- The first thing we need to do is configure the database in freeradius and also from django, i.e., defining models.
- Therefore, we'll need to create models by names AbstractSession, Check, Group, Nas. This shall be defined later after a very detailed discussion.
- FreeRADIUS will basically just have to read/ write in these tables.
- The way in which freeradius does this is configurable and we may find a good default configuration to provide with our app.
- These shall be documented properly giving in detailed explanation of why and how the things are done to avoid confusion.

1.4 Create abstract user models using the swappable user model mechanism

- A `RadiusUser` model shall be defined and shall be linked to `User` with one to one mapping using the swappable user model mechanism. It can be referred from **here**.
- We use this mechanism so as to provide the user a higher level of abstraction and user custom defined user models as per convenience.
- The code structure would be similar to **django-netjsonconfig**.
- Dynamic creation of these models shall be discussed in the next section.
- The basic code is shown below.

- The model of the RadiusUser model can be defined as follows

```
from django.conf import settings
from django.db import models

class RadiusUser(models.Model):
    user = models.OnetoOneField(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
        primary_key=True
    )

    # ... Remaining fields
```

Phase 2

2.1 Implement 'Batch add users' feature and save details in the database

- In order to accomplish this, we define a new model by the name **`Batch`** which inherits from **`AbstractBatch.`**
- This model stores all the important batch details like ***datetime_created,*** having many to many links with **`User.`**
- We then define another model name **`UserDetails`** which will have a foreign key to **`User`** and store usernames and passwords in clear text format.
- Whenever a Batch instance is being created, there will be a PDF file automatically being generated. This can be achieved by using packages like **reportlab** and can be referred from **here.** This would be a list of all the usernames and passwords which could be used in a conference for example.
- We also define expire functions for each batch where an authorized user can specify a particular time duration for the usernames and passwords to be authentic.
- This can be simply done by adding an **`expire`** and an **`expire_time`** attributes to the **`Batch`** model.
- Reactivation of accounts can implemented by simply changing the expire attribute to False and regeneration of username and password and reprinting the PDFs based on user's preference.
- There would be a API made for the above feature.

2.2 Implement a RESTFul API through which authorized users will be able to retrieve radius sessions.

- This will be implemented by using the Django REST framework which will ease the API design.

- There would be a class based view defined which returns all the sessions active by filtering from the Session model of FreeRADIUS database.

2.3 Extensible SignUp API

- An extra feature in the project would be addition of a SignUP feature where a user can use the urls and post data accordingly.
- Very minimum user details will be asked for, like name, phone number, email, etc. These can be finalised after a thorough discussion.
- Therefore, there will be urls, views defined and the data goes as instances of the above defined classes after proper validation.
- This could be used when some organisation for example doesn't want to create all the usernames and passwords on its own and wishes to provide signUp services.
- This can be extended to provide wide variety of signup features like double authentication, email verification, etc.

Phase 3

3.1 Custom RADIUS attributes option in the admin panel

- RADIUS has a vast list of attributes. Most of which depend on the user's requirements. Therefore, by default we store only the typically used attributes of RADIUS. Giving users options to specify more attributes is one of the most desirable features.
- We will have to dynamically store the data/ allocate space/ create tables in order to achieve it.
- Using packages like **django-eav** shall help in achieving it. The model needs to be registered by eav first and then we can dynamically add attributes.
- Therefore, we can store the list of attributes in a list/ tuple and then give options and take inputs.
- This is one the most important features that we are going implement in this project.

3.2 Provide documentation using python-sphinx

- Sphinx is a great tool and renders beautiful documentation with a variety of output formats.
- I believe in documentation of the code simultaneously after writing the code. Never had much experience with python-sphinx though. Can easily go through it when required.

Proposed Timeline

I have my end semester examinations between 23 April - 1 May. I will be inactive between 10 April to 2 May for exam preparations, which is way off the timeline.

I believe I have enough fuel to get started on my goals - as a result to my involvement with the organization for almost two months now. So I will be setting grounds early to avoid stopgaps during the actual GSoC period. There are a couple of weeks before the actual timeline where I would make sure that I have done all the homework. All dates mentioned here are weeks - starting from Monday and ending on Sunday.

3 April - 9 April (Homework week)

- Setup a RADIUS server and research more about the functioning and the database management of freeradius.
- Learn writing efficient tests.
- Work on unmerged pull requests.

10 April - 7 May (Inactive period)

- End semester exams will be near so, I'll be bit inactive, though always available on email or any other means of communication. At times, I'll try solving an issue or play around with OpenWISP 2 and networks for recreation.
- Between 2 May - 8 May, I'll be packing up from the University campus and moving home, so mostly inactive these days.

8 May - 14 May (Community bonding - I)

- Interact with the members of the community and discuss about the project with the mentor.
- This would be the time I'd utilize properly and learn the concepts which I ain't so familiar with so that I don't lag once the coding period begins. This would involve:
    - Researching more about the functioning of RADIUS server
    - Go through FreeRADIUS documentation and implement test applications with FreeRADIUS configured.
- The next 10 days would be the same too. This period is basically a learning period where I plan to break the ice and get a broader and yet a clear picture of the project.

15 May - 21 May (Community bonding - II)

- Discuss about the workflow with the mentor and make mock ups/ wireframes of the design going to be implemented.

- Decide goals and non-goals of the project and get a clear idea of the features to implement.
- Fix issues in the repositories and add new features to get a quick head start in developing the application.

22 May - 28 May (Community bonding III)

- Finalizing all the technology stack going to be used and the backend design.
- Learn more about making reusable apps and mastering them by making a few test applications.
- Set up initial directory structure to get a kick start to development from the coming week.

29 May - 4 June (Week 1)

- Start work with full enthusiasm.
- Initialise the project repository.
- This involves adding a setup.py file that will be used to install the python package of the reusable django app.
- Create the abstract models that were decided and write tests correspondingly.

5 June - 11 June (Week 2)

- Document all the features written till now in a README file.
- Work on abstract models to manage freeradius databases and create User models accordingly.

12 June - 18 June (Week 3)

- Write tests for the features developed in the previous week.
- Make sure that the models defined are fully functional and in configuration with the freeradius services.
- Obtain a good test coverage.

19 June - 25 June (Week 4)

- This is a buffer week for preparing the code to be presentable for mid-term evaluation.
- My target in this week will be to put a firm pencils down on work done so far, updating documentation, writing tests and code cleanup.

Milestone reached: Phase 1 completed.

26 June - 2 July (Week 5)

- Models and views for batch add users feature.

- Code APIs for the the above features and write tests accordingly
- Implementing the PDF rendering
- Testing the above features

3 July - 9 July (Week 6)

- Document the newly added feature and check the test results.
- APIs for being able to retrieve the list of active sessions.
- Writing tests for the same.

10 July - 16 July (Week 7)

- Work on the custom RADIUS attributes feature in the admin panel
- Clean the code and document some remaining old features implemented.

Milestone reached: Phase 2 completed

My institute re-opens on 18 July, till now I was totally free throughout the day so could easily code for 8 - 10 hours a day. I have kept enough buffers to clear backlogs if any and I am confident that I can work upon the timeline I stated so far. I can work for about 7-8 hours a day from now.

17 July - 23 July (Week 8)

- Continue the work on custom RADIUS features from the admin panel.
- Write tests for the same and document features accordingly.

24 July - 30 July (Week 9)

- It is not necessary that everything goes as planned out and there might be unavoidable delays due to a nasty bug hidden from eyesight.
- Finalize the content of deliverables after Mid-Term Evaluation and update documentation.
- Document the code using python-sphinx and render documentation according to the organization’s preferences.

31 July - 5 Aug (Week 10)

- Document using python-sphinx if something was missed.
- Provide all the instructions on how to setup the freeradius database compatible with the app.
- Debugging and improving the test coverage will be the top priority.
- Improving UI/UX being the next.

Milestone reached: Phase 3 completed

6 Aug - 12 Aug (Week 11)

- **Tentative wrap up - cosmetic refinements.**
- Update documentation - write unit tests wherever missed out and do a general code cleanup. Make sure there is nothing left undone and everything is tidy.

13 Aug - 19 Aug (Week 12)

- Prepare content for the end term evaluation, including deliverables from the wish list.

20 Aug - 29 Aug (Conclusion)

- Buffer week and conclusion of GSoC.

Availability

My vacations start from 3 May and end on 15 July. The official GSoC period is from 5 May to 21 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. Also, I don't plan on doing any internships this 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. I'm interested as I always was. I've been contributing to OpenWISP for over a month now and I've really loved the experience. I got familiar with the community and I believe I've learnt a lot interacting with the prospective mentors. I feel this kind of mentorship is necessary. I'll be an active member in the community and keep contributing. My motivation would always be that I'd be able to contribute to something big and widely in use application. This gives me a lot of satisfaction.

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

Well, I'm always ready for some work that'd teach me something new and cool. I'd be really happy to associate with the community for any business opportunities. I shall take time out of my tough schedule at the college and willingly work for the upliftment of the community.

References

1. **OpenWISP GSoC '17 Ideas page**
2. **FreeRADIUS Documentation**
3. **FreeRADIUS wiki - configuring databases**
4. **Documentation of unittest framework**
5. **Mesh networking**
6. **Github repositories of django-netjsonconfig, django-x509, OWUMS**

****