HEIMDALL: A DIGITAL THREAT MANAGEMENT PLATFORM

Technical Manual

by

Killian O'Connor

For IT Carlow

10/4/2020

Table of Contents

1. Major Technologies Used

1.1 Git

This is a version control system utilised to track changes made to the code over time. Git, coupled with a remote code repository, greatly aided in my ability to work on the same code both from my home PC and laptop.

1.2 GitLab

Gitlab provided a private remote code repository as well as foundational DevOps tools for my pipeline. The inbuilt job scheduling and continuous integration pipelines allowed me to test the major build of Heimdall as they were published to the master branch.

1.3 Docker

A technology used to design and create containerised applications. Docker was by far the most cutting edge technology touched on in this project. The technology allows for applications be deployed preconfigured and "as is" on systems regardless of their environment. This greatly eased the prospect of developing on both my home Windows machine, and my Ubuntu laptop. I had no previous experience with Docker, however with this project completed, I'm sure that I will utilise it in future projects.

1.4 Docker-Compose

This is a tool for defining and running multi-container Docker application. In general containers run in on an individual basis, however this tool allowed the development on communications and processes between containers. While this technology works well, it is very static in its configurations and in future I would look towards other solutions such as Kubernetes.

1.5 Django

The core framework of the project. It is a python based web framework, of which I had no previous experience.

1.6 Nginx

A web server and reverse proxy that was utilised in the production version of Heimdall. A container configured with Nginx served to deliver content from the remote server.

1.7 PostgreSQL

A relational database management system, while similar in syntax and usage to MySQL, of which I have some practical knowledge. This was used for the backend database of the project.

1.8 Python

An extremely popular programming language, this was the main language utilised in the project. While I initially had some experience with using python, it was all hobbyist knowledge. I had to significantly increase my understanding of the language and its idiosyncrasies to make progress in the project.

1.9 BeautifulSoup

This is a python package used for parsing the HTML pages the web scrapers encountered.

1.10 Bash Scripting

Bash scripts were needed for the remote deployment of the Heimdall application to Digital Ocean cloud instances.

1.11 LaTeX

This is a typesetting system which is utilised to specify the format of a documents contents. This Technology was planned to be utilised in dynamic report generation. The example LaTeX templates can be found in the “report_templates” folder.

1.12 Digital Ocean:

Digital Ocean is a cloud computing company whose infrastructure was used to host the remote server instances of Heimdall. While the solution initially seems cheap, the long term cost became too great to continue using their services.

1.13 Balsamiq Wireframes

This is a desktop application utilised to design UI wireframes.

1.14 Lucid Charts

This web application was utilised to design and create the diagrams needed for the project.

2. Coding Standards

Coding standards were maintained by the use of Flake8. This program automatically checked all python code with compliance to the PEP8 standards [1].

A changelog was maintained to track the changes made to the project over time. The changelog recording was made in line with Semantic Versioning standards [2].

3. Version Control

Git was utilised for version control on the project. GitLab was used for a private remote code repository as well as a project Kanban board. A local and remote "Dev" branch was used for minor changes to code, and then major, stable versions were merged to the master branch. The GitLab repo can be found at:

`https://gitlab.com/heimdall-fyp/heimdall-console`

4. Testing

The GitLab CI/CD pipeline was utilised to perform build tests on each commit. Should the project pass the "Build" stage correctly it was then moved onto the "Deploy" stage where the Docker containers and relevant files were deployed to remote Digital Ocean servers.

Basic manual testing was utilised to confirm the functionality of various aspects of the project. The OSINT Scanner module was tested against a live site, mappa.ie, with the permission of the owner.

5. Code Snippets

Due to the large amount of file associated with the project, only files deemed crucially important will be detailed in this document.

5.1 Setup_env.sh

```
#!/bin/sh

cat .env.dev >> .env
<<COMMENT
echo DEBUG=0 >> .env
```

```
echo SQL_ENGINE=django.db.backends.postgresql >> .env
echo DATABASE=postgres >> .env
echo DJANGO_ALLOWED_HOSTS=localhost, 127.0.0.1, [::1], 178.62.9.125 >> .env
echo SECRET_KEY=$SECRET_KEY >> .env
echo SQL_DATABASE=$SQL_DATABASE >> .env
echo SQL_USER=$SQL_USER >> .env
echo SQL_PASSWORD=$SQL_PASSWORD >> .env
echo SQL_HOST=$SQL_HOST >> .env
echo SQL_PORT=$SQL_PORT >> .env
COMMENT

echo WEB_IMAGE=$IMAGE:web  >> .env
echo NGINX_IMAGE=$IMAGE:nginx  >> .env
echo CI_REGISTRY_USER=$CI_REGISTRY_USER   >> .env
echo CI_JOB_TOKEN=$CI_JOB_TOKEN  >> .env
echo CI_REGISTRY=$CI_REGISTRY  >> .env
echo IMAGE=$CI_REGISTRY/$CI_PROJECT_NAMESPACE/$CI_PROJECT_NAME >> .env

echo WEB_IMAGE=$IMAGE:web  >> .env.prod
echo NGINX_IMAGE=$IMAGE:nginx  >> .env.prod
echo CI_REGISTRY_USER=$CI_REGISTRY_USER   >> .env.prod
echo CI_JOB_TOKEN=$CI_JOB_TOKEN  >> .env.prod
echo CI_REGISTRY=$CI_REGISTRY  >> .env.prod
echo IMAGE=$CI_REGISTRY/$CI_PROJECT_NAMESPACE/$CI_PROJECT_NAME >> .env.prod
```

5.2 deploy.sh

```
#!/bin/sh

ssh -o StrictHostKeyChecking=no root@$DIGITAL_OCEAN_IP_ADDRESS << 'ENDSSH'
  cd /heimdall
  export $(cat .env.prod | xargs)
  docker login -u $CI_REGISTRY_USER -p $CI_JOB_TOKEN $CI_REGISTRY
  docker pull $IMAGE:web
  docker pull $IMAGE:nginx
  docker-compose -f docker-compose.prod.yml up -d
ENDSSH
```

5.3 .gitlab-ci.yml

```
image:
  name: docker/compose:1.25.3
  entrypoint: [""]

services:
  - docker:dind

stages:
  - build
  - deploy

variables:
  DOCKER_HOST: tcp://docker:2375
```

```
  DOCKER_DRIVER: overlay2

before_script:
  - export IMAGE=$CI_REGISTRY/$CI_PROJECT_NAMESPACE/$CI_PROJECT_NAME
  - export WEB_IMAGE=$IMAGE:web
  - export NGINX_IMAGE=$IMAGE:nginx
  - apk add --no-cache openssh-client bash
  - chmod +x ./setup_env.sh
  - bash ./setup_env.sh
  - cat .env
  - docker login -u $CI_REGISTRY_USER -p $CI_JOB_TOKEN $CI_REGISTRY

build:
  stage: build
  script:
    - docker pull $IMAGE:web || true
    - docker pull $IMAGE:nginx || true
    - docker-compose -f docker-compose.ci.yml build
    - docker push $IMAGE:web
    - docker push $IMAGE:nginx

deploy:
  stage: deploy
  script:
    - mkdir -p ~/.ssh
    - echo "$PRIVATE_KEY" | tr -d '\r' > ~/.ssh/id_rsa
    - chmod 700 ~/.ssh/id_rsa
    - eval "$(ssh-agent -s)"
    - ssh-add ~/.ssh/id_rsa
    - ssh-keyscan -H 'gitlab.com' >> ~/.ssh/known_hosts
    - chmod +x ./deploy.sh
    - scp  -o StrictHostKeyChecking=no -r ./.env.prod ./docker-
compose.prod.yml root@$DIGITAL_OCEAN_IP_ADDRESS:/heimdall
    - bash ./deploy.sh
  only:
      - master
```

5.4 docker-compose.ci.yml

```
version: "3.7"

services:
  web:
    build:
      context: ./app
      dockerfile: Dockerfile.prod
      cache_from:
        - "${WEB_IMAGE}"
    image: "${WEB_IMAGE}"
    command: gunicorn heimdall_console.wsgi:application --bind 0.0.0.0:8000
    volumes:
      - static_volume:/usr/src/app/staticfiles
      - media_volume:/usr/src/app/mediafiles
    expose:
```

```
      - 8000
    env_file: .env
  nginx:
    build:
      context: ./nginx
      cache_from:
        - "${NGINX_IMAGE}"
    image: "${NGINX_IMAGE}"
    volumes:
      - static_volume:/usr/src/app/staticfiles
      - media_volume:/usr/src/app/mediafiles
    ports:
      - 1337:80
    depends_on:
      - web

volumes:
  static_volume:
  media_volume:
```

5.5 docker-compose.prod.yml

```
version: '3.7'

services:
  web:
    image: "${WEB_IMAGE}"
    command: gunicorn heimdall_console.wsgi:application --bind 0.0.0.0:8000
    volumes:
      - static_volume:/home/app/web/staticfiles
      - media_volume:/home/app/web/mediafiles
    expose:
      - 8000
    env_file:
      - ./.env.prod
  nginx:
    image: "${NGINX_IMAGE}"
    volumes:
      - static_volume:/home/app/web/staticfiles
      - media_volume:/home/app/web/mediafiles
    ports:
       - 1337:80
    depends_on:
      - web

volumes:
  static_volume:
  media_volume:
```

5.6 docker-compose.yml

```
version: '3.7'
```

```
services:
  web:
    container_name: web
    build: ./app
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - ./app/:/usr/src/app/
    ports:
      - 8000:8000
    env_file:
      - ./.env.dev
    depends_on:
      - db
  db:
    container_name: db
    image: postgres:12.0-alpine
    volumes:
      - postgres_data:/var/lib/postgresql/data/
    environment:
      - POSTGRES_USER=heimdall_console
      - POSTGRES_PASSWORD=heimdall_console
      - POSTGRES_DB=heimdall_console_dev
    ports:
      - 5432:5432
volumes:
  postgres_data:
```

5.7 scripts/config.py

```
import psycopg2
from configparser import ConfigParser
import time

osintTargets = ['https://mappa.ie/index.php',
'https://crummy.com/software/BeautifulSoup/bs4/doc/index.html']
darkwebTargets = []


# Dead in batabase config file and return connection parameters
def config(filename='database.ini', section='postgresql'):
    # Create a parser
    parser = ConfigParser()
    # Read config file
    parser.read(filename)

    # Get section, defaults to Postgresql
    db = {}
    if parser.has_section(section):
        params = parser.items(section)
        for param in params:
            db[param[0]] = param[1]
    else:
```

```
        raise Exception('section {0} not found in the {1}
file'.format(section, filename))
    return db


# Connct to the Heimdall Postgreqsl database
def connectionTest():
    conn = None
    try:
        params = config()

        print('Connecting to Postgresql database...')
        conn = psycopg2.connect(**params)
        cur = conn.cusor()
        print('Postgresql db version:')
        cur.execute('SELECT version()')
        db_version = cur.fetchone()
        print(db_version)
        cur.close()
    except(Exception, psycopg2.DatabaseError) as error:
        print(error)
    finally:
        if conn is not None:
            conn.close()
            print('Database connection closed')


def get_user_keys(userId):
    '''
    Query the database for all user keys associated with the user ID
    '''
    sql = "SELECT key FROM module_usage_userKey WHERE \"userId_id\" = %s"

    try:
        params = config()

        print('Attempting to retrieve user keys...')
        conn = psycopg2.connect(**params)
        cur = conn.cursor()
        cur.execute(sql, (userId,))
        userKeys = cur.fetchall()
        print("--Returned Keys: " + str(len(userKeys)))
        cur.close()
        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
        userKeys = []

    finally:
        if conn is not None:
            conn.close()
        return userKeys


def generate_alert(moduleId, userId, alertLink):
    '''
```

```
    Generate an alert in the database based on the discovery of a user key.
    200 defines an all good.
    500 specifies an error.
    '''
    active = True
    summary = "short"
    description = "long"
    otherLinks = "empty"
    conn = None
    sqlParameters = [userId, moduleId, summary, description, alertLink,
otherLinks, active]

    sql = "INSERT INTO module_usage_alert (\"userId_id\", \"moduleId_id\",
\"timeDiscovered\", summary, description, \"alertLink\", \"otherLinks\",
\"active\") VALUES(%s, %s, NOW(), %s, %s, %s, %s, %s) RETURNING \"alertId\""

    try:
        params = config()

        print('Attempting to generate alert...')
        conn = psycopg2.connect(**params)
        cur = conn.cursor()
        cur.execute(sql, sqlParameters)
        generated_alertId = cur.fetchone()[0]
        print("--Alert Generated: " + str(generated_alertId))
        cur.close()
        conn.commit()
        statusCode = 200
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
        statusCode = 500

    finally:
        if conn is not None:
            conn.close()
        return statusCode
```

5.8 scripts/osint_scraper.py

```
import re
import requests
from bs4 import BeautifulSoup
from config import osintTargets, get_user_keys, generate_alert


def collect_links(soup):
    '''
    Process through the page and return valid links
    '''
    print("No links present!")
    # return links


if __name__ == '__main__':
```

```
    moduleId = 1
    userId = 1
    userKeys = get_user_keys(userId)
    urls = osintTargets

    if userKeys:
        print('User keys found')

        for url in urls:
            print("-------------------------")
            print("Checking: " + url)
            r = requests.get(url)
            print("Respose: " + str(r.status_code))
            if r.status_code == 200:
                soup = BeautifulSoup(r.text, 'html.parser')

                # Discover and record all links for future scraping
                '''
                anchors = soup.find_all('a', href=True)
                if anchors:
                    print("Anchors discovered")
                else:
                    print("No anchors found")

                for link in anchors:
                    print(link.get('href'))
                    links.append(link.get('href'))
                links = list(dict.fromkeys(links))
                links.sort()
                # Filter out usage of #
                # filteredLinks = [i for i in links if not regex.match(i)]
                # print(filteredLinks)
                print(links)
                '''

                # Search page for user keys
                for key in userKeys:
                    presentKeys = soup.find_all(string=re.compile(key[0]))
                    if presentKeys:
                        print("-User key: " + key[0] + " present on page!!")
                        if generate_alert(moduleId, userId, url) == 200:
                            print("--Alert Generated Successfully")
                        else:
                            print("--Alert failed")
                    else:
                        print("-User key: " + key[0] + " not found")
            else:
                print("Page is not responding. Skipping.")
    else:
        print('No user keys found!')
```

5.9 nginx/Dockerfile

```
FROM nginx:1.17.5-alpine
```

```
RUN rm /etc/nginx/conf.d/default.conf
COPY nginx.conf /etc/nginx/conf.d
```

5.10 nginx/nginx.conf

```
upstream heimdall_console {
    server web:8000;
}

server {

    listen 80;

    location / {
        proxy_pass http://heimdall_console;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $host;
        proxy_redirect off;
    }

    location /staticfiles/ {
        alias /home/app/web/staticfiles/;
    }

    location /mediafiles/ {
        alias /home/app/web/mediafiles/;
    }

}
```

5.11 app/entrypoint.sh

```
#!/bin/sh

if [ "$DATABASE" = "postgres" ]
then
    echo "Waiting for postgres..."

    while ! nc -z $SQL_HOST $SQL_PORT; do
      sleep 0.1
    done

    echo "PostgreSQL started"
fi

python manage.py flush --no-input
python manage.py migrate
python manage.py collectstatic --no-input --clear

exec "$@"
```

5.12 app/entrypoint.prod.sh

```
#!/bin/sh

if [ "$DATABASE" = "postgres" ]
then
    echo "Waiting for postgres..."

    while ! nc -z $SQL_HOST $SQL_PORT; do
      sleep 0.1
    done

    echo "PostgreSQL started"
fi

exec "$@"
```

5.13 app/Dockerfile

```
# pull official base image
FROM python:3.8.0-alpine

# set work directory
WORKDIR /usr/src/app

# set environment variables
ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

# install psycopg2 dependencies
RUN apk update \
    && apk add postgresql-dev gcc python3-dev musl-dev

# install dependencies
RUN pip install --upgrade pip
COPY ./requirements.txt /usr/src/app/requirements.txt
RUN pip install -r requirements.txt

# copy entrypoint.sh
COPY ./entrypoint.sh /usr/src/app/entrypoint.sh

# copy project
COPY . /usr/src/app/

# run entrypoint.sh
ENTRYPOINT ["/usr/src/app/entrypoint.sh"]
```

5.14 app/Dockerfile.prod

```
############
# BUILDER #
############

# pull official base image
FROM python:3.8.0-alpine as builder

# set work directory
WORKDIR /usr/src/app

# set environment variables
ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

# install psycopg2 dependencies
RUN apk update \
    && apk add postgresql-dev gcc python3-dev musl-dev

# lint
RUN pip install --upgrade pip
RUN pip install flake8
COPY . /usr/src/app/
RUN flake8 --ignore=E501,F401 .

# install dependencies
COPY ./requirements.txt .
RUN pip wheel --no-cache-dir --no-deps --wheel-dir /usr/src/app/wheels -r
requirements.txt


#########
# FINAL #
#########

# pull official base image
FROM python:3.8.0-alpine

# create directory for the app user
RUN mkdir -p /home/app

# create the app user
RUN addgroup -S app && adduser -S app -G app

# create the appropriate directories
ENV HOME=/home/app
ENV APP_HOME=/home/app/web
RUN mkdir $APP_HOME
RUN mkdir $APP_HOME/staticfiles
RUN mkdir $APP_HOME/mediafiles
WORKDIR $APP_HOME

# install dependencies
RUN apk update && apk add libpq
COPY --from=builder /usr/src/app/wheels /wheels
```

```
COPY --from=builder /usr/src/app/requirements.txt .
RUN pip install --upgrade pip
RUN pip install --no-cache /wheels/*

# copy entrypoint-prod.sh
COPY ./entrypoint.prod.sh $APP_HOME

# copy project
COPY . $APP_HOME

# chown all the files to the app user
RUN chown -R app:app $APP_HOME

# change to the app user
USER app

# run entrypoint.prod.sh
ENTRYPOINT ["/home/app/web/entrypoint.prod.sh"]
```

5.15 app/module_usage/admin.py

```
from django.contrib import admin
from .models import Module, User, UserKey, TargetRange, Alert

admin.site.register(User)
admin.site.register(UserKey)
admin.site.register(Alert)
admin.site.register(TargetRange)
admin.site.register(Module)
```

5.16 app/module_usage/forms.py

```
from django import forms
from .models import Module, UserKey, TargetRange, Alert
from datetime import datetime


class ModuleForm(forms.ModelForm):
    class Meta:
        model = Module
        fields = [
            'title',
            'description',
            'resourceReference',
            'active',
        ]


class UserKeyForm(forms.ModelForm):
    class Meta:
        model = UserKey
        fields = [
```

```
            'key',
            'userId',
        ]


class TargetRangeForm(forms.ModelForm):
    class Meta:
        model = TargetRange
        fields = [
            'userId',
            'range',
            'dateAdded',
            'lastAlertDate',
            'active',
        ]


class AlertForm(forms.Form):
    dateFormat = "%Y-%m-%d %H:%M"
    now = datetime.now()
    timeClosed = forms.DateTimeField(
        widget=forms.DateTimeInput(),
        required=True,
        initial=now.strftime(dateFormat)
    )
    remediationComment = forms.CharField(
        widget=forms.Textarea,
        required=True
    )
    active = forms.BooleanField(
        required=False,
    )
```

5.17 app/module_usage/models.py

```
from django.db import models
from django.utils import timezone


class Module(models.Model):
    moduleId = models.AutoField(primary_key=True)
    title = models.CharField(max_length=256, default="No title")
    description = models.TextField(default=None)
    resourceReference = models.TextField(default=None)
    active = models.BooleanField(default=False)

    def __str__(self):
        return self.title


# Move to top level ASAP
class User(models.Model):
    userId = models.AutoField(primary_key=True)
    username = models.TextField(default=None)
```

```
    passwordHash = models.TextField(default=None)
    lastLogin = models.DateTimeField(default=None)

    def __str__(self):
        return str(self.userId) + ", " + self.username


class UserModuleUsage(models.Model):
    usageId = models.AutoField(primary_key=True)
    userId = models.ForeignKey('User', default=0, on_delete=models.CASCADE)
    moduleId = models.ForeignKey('Module', on_delete=models.CASCADE)


class UserKey(models.Model):
    keyId = models.AutoField(primary_key=True)
    userId = models.ForeignKey('User', on_delete=models.CASCADE)
    key = models.TextField(default=None)

    def __str__(self):
        return str(self.keyId)


class TargetRange(models.Model):
    rangeId = models.AutoField(primary_key=True)
    userId = models.ForeignKey('User', on_delete=models.CASCADE)
    range = models.TextField()
    dateAdded = models.DateTimeField(default=None)
    lastAlertDate = models.DateTimeField(default=None)
    active = models.BooleanField(default=False)

    def __str__(self):
        return str(self.rangeId)


class Alert(models.Model):
    alertId = models.AutoField(primary_key=True)
    userId = models.ForeignKey('User', on_delete=models.CASCADE)
    moduleId = models.ForeignKey('Module', on_delete=models.CASCADE,
default=0)
    timeDiscovered = models.DateTimeField(default=None)
    timeClosed = models.DateTimeField(blank=True, null=True, default=None)
    summary = models.TextField(default=None)
    description = models.TextField(default=None)
    alertLink = models.TextField(default=None)
    otherLinks = models.TextField(blank=True, null=True, default=None)
    remediationComment = models.TextField(blank=True, null=True,
default=None)
    active = models.BooleanField(default=True)

    def __str__(self):
        return "AlertID: " + str(self.alertId) + ", for UserID " +
str(self.userId)
```

5.18 app/module_usage/urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('userKeys/', views.userKeys, name='userKeys'),
    path('userKeys/create', views.userKey_create, name='userKey_create'),
    path('userKeys/update/<int:userKeyId>', views.userKey_update,
name='userKey_update'),
    path('userKeys/delete/<int:userKeyId>', views.userKey_delete,
name='userKey_delete'),
    path('userAlerts/', views.userAlerts, name='userAlerts'),
    path('userAlerts/<int:alertId>', views.alertDetails,
name='alertDetails'),
    path('dwScanner/', views.dwScanner, name='dwScanner'),
    path('osintScanner/', views.osintScanner, name='osintScanner'),
    path('cveScanner/', views.cveScanner, name='cveScanner'),
    path('cveScanner/range/', views.cveRange, name='cveRange'),
    path('awsScanner/', views.awsScanner, name='awsScanner'),
    path('awsScanner/range/', views.awsRange, name='awsRange'),
]
```

5.19 app/module_usage/views.py

```
from django.shortcuts import render, get_object_or_404
from django.http import HttpResponseRedirect
from module_usage.models import Module, User, UserKey, TargetRange, Alert
from .forms import UserKeyForm, AlertForm


# --- Generic pages ---
# Base page
def index(request):
    user = request.user
    context = {
        'content': "Module overview page here",
        'page_title': 'Modules',
        'user': user,
    }
    return render(request, 'module_usage/index.html', context)


# View & edit user keys
def userKeys(request):
    userKey_list = UserKey.objects.all()
    context = {
        'page_title': 'User Keys',
        'userKey_list': userKey_list,
    }
    return render(request, 'module_usage/userKeys.html', context)
```

```
# Add user key
def userKey_create(request):
    form = UserKeyForm(request.POST or None)
    if form.is_valid():
        form.save()
        return HttpResponseRedirect('/module/userKeys/')
    context = {
        'form': form,
        'form_type': "Create",
        'page_title': 'Add Key',
    }
    return render(request, 'module_usage/userKeyCreate.html', context)


# Update user key
def userKey_update(request, userKeyId):
    key = UserKey.objects.get(keyId=userKeyId)
    form = UserKeyForm(request.POST or None, instance=key)
    if form.is_valid():
        form.save()
        return HttpResponseRedirect('/module/userKeys/')
    context = {
        'form': form,
        'form_type': "Update",
        'page_title': "Update Key",
    }
    return render(request, 'module_usage/userKeyCreate.html', context)


# Delete user key
def userKey_delete(request, userKeyId):
    key = UserKey.objects.get(keyId=userKeyId)
    key.delete()
    return HttpResponseRedirect('/module/userKeys/')

    form = UserKeyForm(request.POST or None, instance=key)
    if form.is_valid():
        form.save()
        return HttpResponseRedirect('/module/userKeys/')
    context = {
        'form': form,
        'form_type': "Delete",
        'content': "User will be able to update keys here",
        'page_title': "Delete Key",
    }
    return render(request, 'module_usage/userKeyCreate.html', context)


# User alerts
def userAlerts(request):
    alertCount = Alert.objects.count()
    alertList = Alert.objects.filter(active=True).order_by('-
timeDiscovered', 'alertId')
    context = {
        'content': "User will view alerts here",
        'page_title': 'User Alerts',
```

```
        'alert_list': alertList,
        'alert_count': alertCount,
    }
    return render(request, 'module_usage/userAlerts.html', context)


# Provide details on an alert
def alertDetails(request, alertId):
    alertItem = Alert.objects.get(pk=alertId)
    form = AlertForm(request.POST or None)
    if form.is_valid():
        Alert.objects.filter(pk=alertId).update(
            remediationComment=form.cleaned_data['remediationComment'],
            timeClosed=form.cleaned_data['timeClosed'],
            active=form.cleaned_data['active']
        )
        return HttpResponseRedirect('/module/userAlerts/')
    context = {
        'content': "Provide the full details for an alert",
        'page_title': 'Alert Details',
        'alert_item': alertItem,
        'form': form,
    }
    return render(request, 'module_usage/viewAlert.html', context)


# --- AWS S3 Monitor pages ---
# Home page
def awsScanner(request):
    context = {
        'content': "Under Construction (AWS S3 Monitor)",
        'page_title': 'S3 Monitor',
    }
    return render(request, 'module_usage/aws/index.html', context)


# User defined target range
def awsRange(request):
    context = {
        'content': "Under Construction (AWS Range)",
        'page_title': 'S3 Monitor: Target Range',
    }
    return render(request, 'module_usage/aws/targetRange.html', context)


# --- CVE Scanner pages ---
# Home page
def cveScanner(request):
    context = {
        'content': "Under Construction (CVE Scanner)",
        'page_title': 'CVE Scanner',
    }
    return render(request, 'module_usage/cve/index.html', context)


# User defined targt range
```

```
def cveRange(request):
    context = {
        'content': "Under Construction (CVE Range)",
        'page_title': 'CVE Scanner: Target Range',
    }
    return render(request, 'module_usage/cve/targetRange.html', context)


# --- Dark Web Scanner pages ---
# Home page
def dwScanner(request):
    context = {
        'content': "Under Construction (Dark Web Scanner)",
        'page_title': 'Dark Web Monitor',
    }
    return render(request, 'module_usage/dw/index.html', context)


# --- OSINT Scanner pages ---
# Home page
def osintScanner(request):
    context = {
        'content': "Under Construction (OSINT Scanner)",
        'page_title': 'OSINT Monitor',
    }
    return render(request, 'module_usage/osint/index.html', context)
```

5.20 app/module_usage/templates/module_usage/base_generic.html

Home | Alerts | Modules | Reports | Keys | Logout

{% if error_message %}

{{ error_message }}

{% endif %} {% block content %} {% endblock content %}

5.21 app/module_usage/templates/module_usage/index.html

```
{% extends "module_usage/base_generic.html" %}

{% block title %}{{page_title}}{% endblock title %}
```

```
{% block content %}
    <br>
    <h1>Available Modules</h1>
    <br>
    <ul>
        <li><a href="osintScanner/">OSINT Threat Monitoring</a></li>
        <li><a href="dwScanner/">Dark Web Threat Monitoring</a></li>
        <li><a href="cveScanner/">CVE Scanner</a></li>
        <li><a href="awsScanner/">AWS S3 Monitoring</a></li>
    </ul>
    <br>
    <h2>current user: {{user}}</h2>
    {{content}}
{% endblock content %}
```

5.22 app/module_usage/templates/module_usage/main-app.html

```
{% extends "module_usage/base_generic.html" %}

{% block title %}{{page_title}}{% endblock title %}

{% block content %}
    <br>
    <h1>Welcome to Heimdall</h1>
    <h2>current user: {{user}}</h2>
    <br>
    {{content}}
{% endblock content %}
```

5.23 app/module_usage/templates/module_usage/userAlerts.html

{% extends "module_usage/base_generic.html" %} {% block title %}{{page_title}}{% endblock title %} {% block content %}

View Alerts

Alert ID	Time Discovered	Details	Action Alert
			{% for Alert in alert_list %}
{{Alert.alertId}}	**{{Alert.timeDiscovered}}**	**{{Alert.summary}}**	**Action**
			{% endfor %}

{{content}} {% endblock content %}

5.24 app/module_usage/templates/module_usage/userKeyCreate.html

```
{% extends "module_usage/base_generic.html" %}

{% block title %}{{page_title}}{% endblock title %}

{% block content %}
    <br>
    <h1>{{form_type}} Key</h1>

    <form method='POST'>
        {% csrf_token %}
        {{ form.as_p }}
        <input type="submit" value='{{form_type}} key' />
    </form>

    {{content}}
{% endblock content %}
```

5.25 app/module_usage/templates/module_usage/userKeys.html

{% extends "module_usage/base_generic.html" %} {% block title %}{{page_title}}{% endblock title %} {% block content %}

Enter & View User Keys

Create new key

User Key	Edit Key	Delete Key
{% for UserKey in userKey_list %}		
{{UserKey.key}}	Edit Key	Delete Key
{% endfor %}		

{{content}} {% endblock content %}

5.26 app/module_usage/templates/module_usage/viewAlert.html

```
{% extends "module_usage/base_generic.html" %}

{% block title %}{{page_title}}{% endblock title %}
```

```
{% block content %}
    <br>
    <p><b>Details for AlertID:</b> {{ alert_item.alertId }}</p>
    <p><b>Discovered:</b> {{ alert_item.timeDiscovered }}</p>
    <p><b>Main Link:</b> {{ alert_item.alertLink }}</p>
    <p><b>Related Links:</b> {{ alert_item.otherLinks }}</p>
    <p><b>Alert Details:</b> {{ alert_item.description }}</p>
    <br>
    <br>
    <b>Remediation Comments:</b>
    <form method='POST'>
        {% csrf_token %}
        {{ form.as_p }}
        <input type="submit" value='Close Alert' />
    </form>
    {{content}}
{% endblock content %}
```

5.27 app/module_usage/templates/module_usage/osint/index.html

```
{% extends "module_usage/base_generic.html" %}

{% block title %}{{page_title}}{% endblock title %}

{% block content %}
    <br>
    <h1>OSNIT Threat Intelligence</h1>
    {{content}}
{% endblock content %}
```

5.28 report_templates/analyst_report_template.tex

```
\title{Analyst Report}
\author{
        Generated by Heimdall
}
\date{\today}

\documentclass[12pt]{article}

\usepackage{hyperref}

\begin{document}
\maketitle

\begin{center}
\begin{tabular}{l r}
Requesting user: & John Hancock \\ % Name of the user requesting the report
\end{tabular}
\end{center}
```

```
% Alerts will be populated in chronological order
\section*{Alert: 0001} % Code for the alert
\begin{tabular}{l r}
Date Discovered: & January 1, 2020 \\ % Date the alert was discovered
Time Discovered: & 19:20 (GMT) \\ % Time the alert was discovered
Module Used: & OSINT Scanner \\ % Module used to generate alert
Closed Date: & January 2, 2020 \\ %Date alert was closed if it was
remediated
\end{tabular}

\paragraph{Alert Location:}
\url{http://evilmarket.com/info/item=123}

\paragraph{Related Resources:}
\begin{tabular}{l r}
\href{http://heimdall.com/intel/site=evilmarket}{Heimdall Intelligence
report on site} \\
\url{http://news.com/article/evil-market-is-evil/} \\
\end{tabular}

\paragraph{Summary:} % Short summary of alert
User key: [affected\_key] was discovered on site [Alert\_Location]. This
site is a know marketplace for stolen information.

\paragraph{Description:} % Long form description of alert with relevent
material
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat
non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

\paragraph{Remediation Comments:} % If alert was closed, include comments
The affected individuals account credentials were reset and their device
imaged.
\bibliographystyle{abbrv}
\bibliography{main}

\end{document}
```

5.29 report_templates/executive_report_template.tex

```
\title{Executive Report}
\author{
        Generated by Heimdall \\
        Requesting user: John Hancock
}
\date{\today}

\documentclass[12pt]{report}

\usepackage{hyperref} % Links and hrefs
%The mathplotlib python class will be used to dynamically plot graphs
```

```
\begin{document}
\maketitle

\chapter*{Alerting Trends}

Will be done dynamically by the PyLaTeX mathplotlib class.

\chapter*{Discoveries by Module}

Will be done dynamically by the PyLaTeX mathplotlib class.

\chapter*{Summary of Alerts} % Alerts will be populated in chronological
order

\section*{Alert: 0001} % Code for the alert
\begin{tabular}{l r}
Time Discovered: & 19:20 (GMT) \\ % Time the alert was discovered
Module Used: & OSINT Scanner \\ % Module used to generate alert
Closed Date: & January 2, 2020 \\ %Date alert was closed if it was
remediated
\end{tabular}

\paragraph{Summary:} % Short summary of alert
User key: [affected\_key] was discovered on site [Alert\_Location]. This
site is a know marketplace for stolen information.

\paragraph{Remediation Comments:} % If alert was closed, include comments
The affected individuals account credentials were reset and their device
imaged.

\bibliographystyle{abbrv}
\bibliography{main}

\end{document}
```

References

[1] Python Software Foundation, “PEP 8 -- Style Guide for Python Code,” 1 8 2013. [Online]. Available: https://www.python.org/dev/peps/pep-0008/. [Accessed 10 4 2020].

[2] T. Preston-Werner, “Semantic Version 2.0.0,” [Online]. Available: https://semver.org/. [Accessed 10 4 2020].