ATTACK DEFENSE

by PentesterAcademy

Name	TruffleHog: Locating Sensitive Information
URL	https://www.attackdefense.com/challengedetails?cid=2044
Type	DevSecOps Basics: Sensitive Information Scan

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Challenge Description

TruffleHog is open-source software for searching sensitive information in git repositories. It does this by checking the code in the commit history, branches with regular expression.

A Kali CLI machine (kali-cli) is provided to the user with TruffleHog installed on it. The source code for three sample web applications is provided in the home directory of the root user for scanning.

Objective: Scan the web application source code with TruffleHog and find sensitive information in the code!

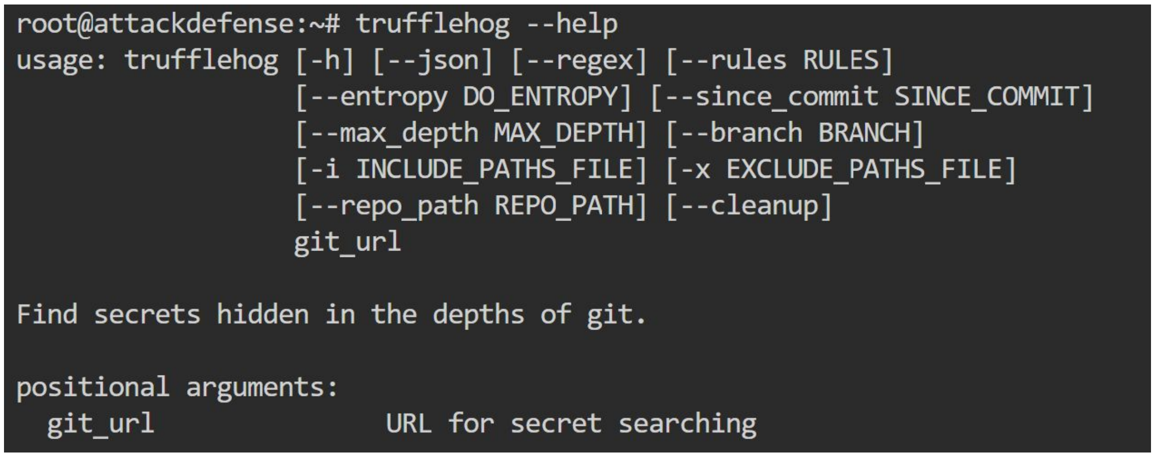
Instructions:

- The source code of web applications is provided at /root/github-repos

Solution

Step 1: Check the available options for TruffleHog

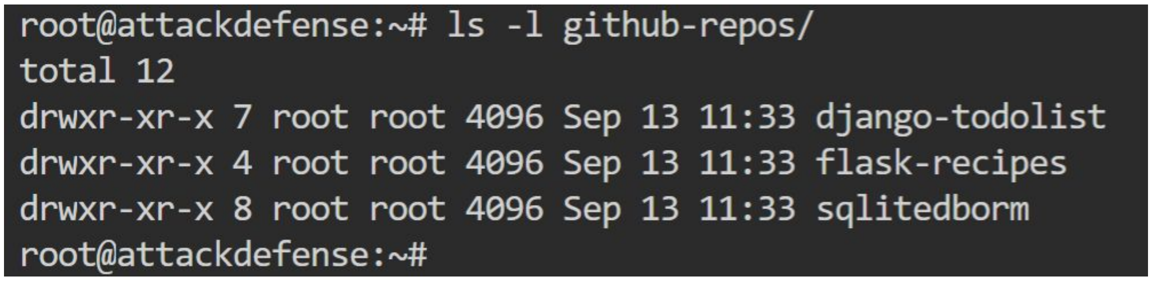
Command: trufflehog --help



```
root@attackdefense:~# trufflehog --help
usage: trufflehog [-h] [--json] [--regex] [--rules RULES]
                  [--entropy DO_ENTROPY] [--since_commit SINCE_COMMIT]
                  [--max_depth MAX_DEPTH] [--branch BRANCH]
                  [-i INCLUDE_PATHS_FILE] [-x EXCLUDE_PATHS_FILE]
                  [--repo_path REPO_PATH] [--cleanup]
                  git_url

Find secrets hidden in the depths of git.

positional arguments:
  git_url               URL for secret searching
```

Step 2: Check the provided web applications.

Command: ls -l github-repos



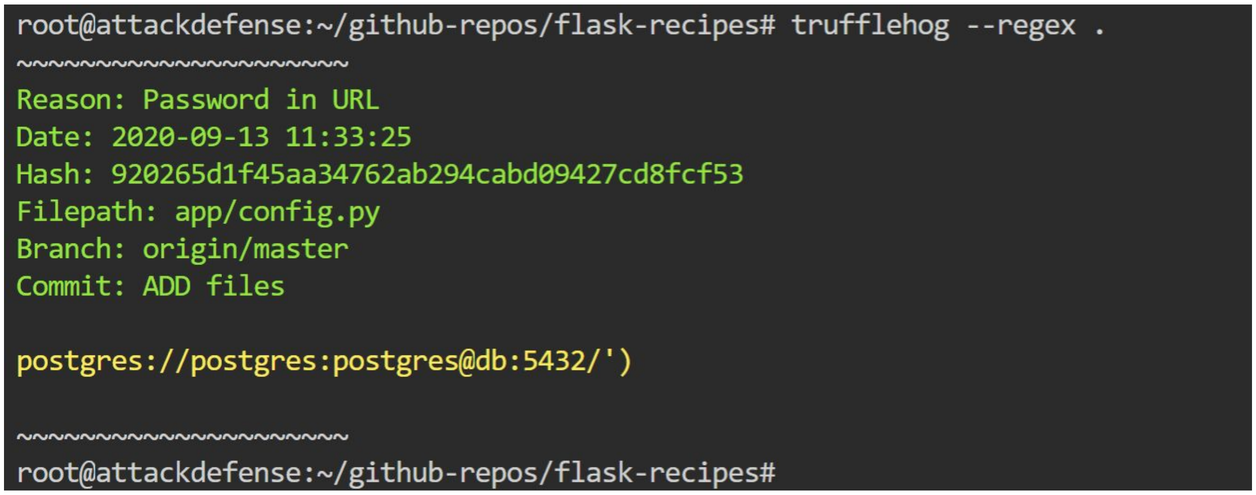
```
root@attackdefense:~# ls -l github-repos/
total 12
drwxr-xr-x 7 root root 4096 Sep 13 11:33 django-todolist
drwxr-xr-x 4 root root 4096 Sep 13 11:33 flask-recipes
drwxr-xr-x 8 root root 4096 Sep 13 11:33 sqlitedborm
root@attackdefense:~#
```

We will take one example at a time and run the tool on that.

Example 1: Flask Recipes

Step a: Change to the flask-recipes directory and run the trufflehog tool.

Commands:
cd github-repos/flask-recipes/
trufflehog --regex .



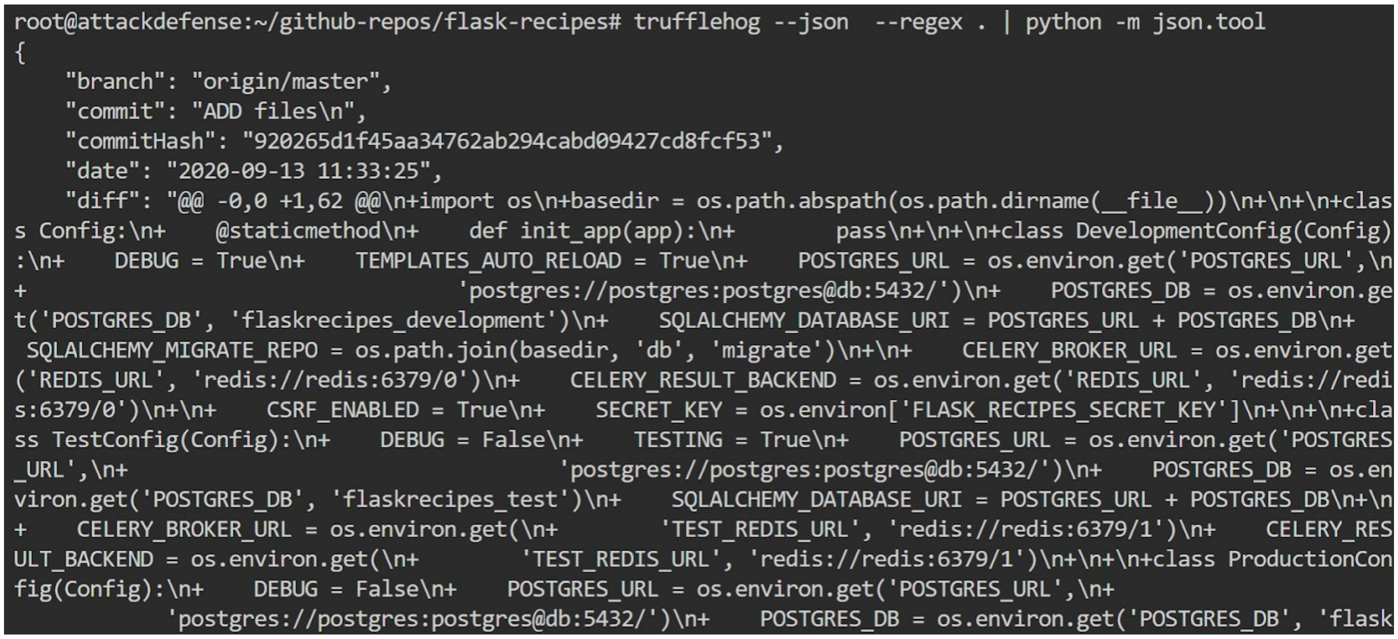
```
root@attackdefense:~/github-repos/flask-recipes# trufflehog --regex .
~~~~~~~~~~~~~~~~~~~~~
Reason: Password in URL
Date: 2020-09-13 11:33:25
Hash: 920265d1f45aa34762ab294cabd09427cd8fcf53
Filepath: app/config.py
Branch: origin/master
Commit: ADD files

postgres://postgres:postgres@db:5432/')

~~~~~~~~~~~~~~~~~~~~~
root@attackdefense:~/github-repos/flask-recipes#
```

Issues Detected

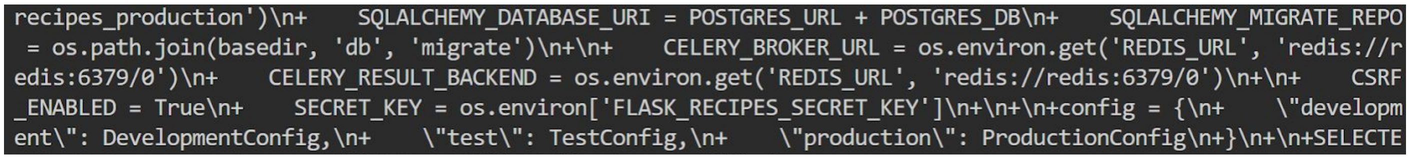
- Found postgresql credentials from the application

Step b: Run the trufflehog tool and dump the output in JSON format.

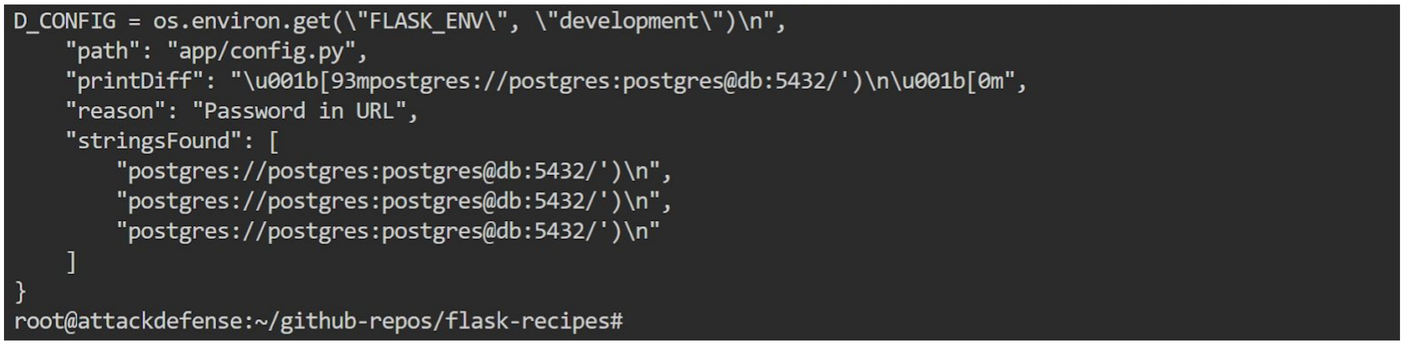
Command: trufflehog --json --regex . | python -m json.tool



```
root@attackdefense:~/github-repos/flask-recipes# trufflehog --json  --regex . | python -m json.tool
{
    "branch": "origin/master",
    "commit": "ADD files\n",
    "commitHash": "920265d1f45aa34762ab294cabd09427cd8fcf53",
    "date": "2020-09-13 11:33:25",
    "diff": "@@ -0,0 +1,62 @@\n+import os\n+basedir = os.path.abspath(os.path.dirname(__file__))\n+\n+\n+clas
s Config:\n+    @staticmethod\n+    def init_app(app):\n+        pass\n+\n+\n+class DevelopmentConfig(Config)
:\n+    DEBUG = True\n+    TEMPLATES_AUTO_RELOAD = True\n+    POSTGRES_URL = os.environ.get('POSTGRES_URL',\n
+                                  'postgres://postgres:postgres@db:5432/')\n+    POSTGRES_DB = os.environ.ge
t('POSTGRES_DB', 'flaskrecipes_development')\n+    SQLALCHEMY_DATABASE_URI = POSTGRES_URL + POSTGRES_DB\n+
 SQLALCHEMY_MIGRATE_REPO = os.path.join(basedir, 'db', 'migrate')\n+\n+    CELERY_BROKER_URL = os.environ.get
('REDIS_URL', 'redis://redis:6379/0')\n+    CELERY_RESULT_BACKEND = os.environ.get('REDIS_URL', 'redis://redi
s:6379/0')\n+\n+    CSRF_ENABLED = True\n+    SECRET_KEY = os.environ['FLASK_RECIPES_SECRET_KEY']\n+\n+\n+cla
ss TestConfig(Config):\n+    DEBUG = False\n+    TESTING = True\n+    POSTGRES_URL = os.environ.get('POSTGRES
_URL',\n+                                          'postgres://postgres:postgres@db:5432/')\n+    POSTGRES_DB = os.en
viron.get('POSTGRES_DB', 'flaskrecipes_test')\n+    SQLALCHEMY_DATABASE_URI = POSTGRES_URL + POSTGRES_DB\n+\n
+    CELERY_BROKER_URL = os.environ.get(\n+        'TEST_REDIS_URL', 'redis://redis:6379/1')\n+    CELERY_RES
ULT_BACKEND = os.environ.get(\n+        'TEST_REDIS_URL', 'redis://redis:6379/1')\n+\n+\n+class ProductionCon
fig(Config):\n+    DEBUG = False\n+    POSTGRES_URL = os.environ.get('POSTGRES_URL',\n+
            'postgres://postgres:postgres@db:5432/')\n+    POSTGRES_DB = os.environ.get('POSTGRES_DB', 'flask
```



```
recipes_production')\n+    SQLALCHEMY_DATABASE_URI = POSTGRES_URL + POSTGRES_DB\n+    SQLALCHEMY_MIGRATE_REPO
 = os.path.join(basedir, 'db', 'migrate')\n+\n+    CELERY_BROKER_URL = os.environ.get('REDIS_URL', 'redis://r
edis:6379/0')\n+    CELERY_RESULT_BACKEND = os.environ.get('REDIS_URL', 'redis://redis:6379/0')\n+\n+    CSRF
_ENABLED = True\n+    SECRET_KEY = os.environ['FLASK_RECIPES_SECRET_KEY']\n+\n+\n+config = {\n+    \"developm
ent\": DevelopmentConfig,\n+    \"test\": TestConfig,\n+    \"production\": ProductionConfig\n+}\n+\n+SELECTE
```

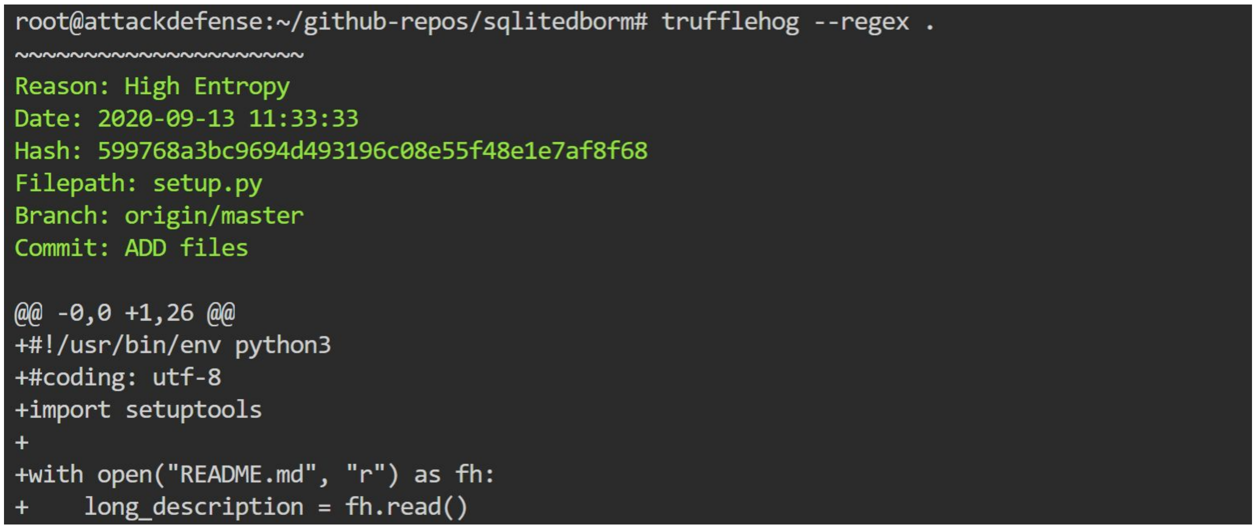


```
D_CONFIG = os.environ.get(\"FLASK_ENV\", \"development\")\n",
    "path": "app/config.py",
    "printDiff": "\u001b[93mpostgres://postgres:postgres@db:5432/')\n\u001b[0m",
    "reason": "Password in URL",
    "stringsFound": [
        "postgres://postgres:postgres@db:5432/')\n",
        "postgres://postgres:postgres@db:5432/')\n",
        "postgres://postgres:postgres@db:5432/')\n"
    ]
}
root@attackdefense:~/github-repos/flask-recipes#
```

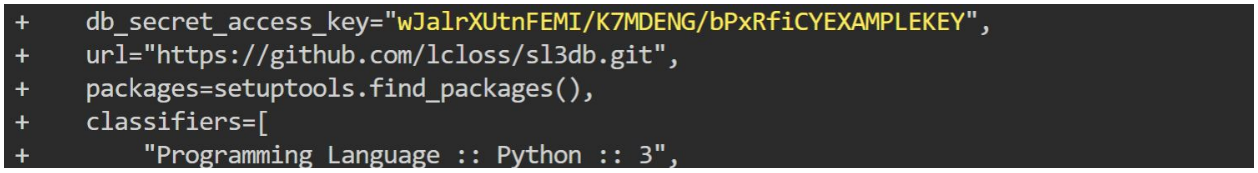
Example 2: sqlitedborm

Step a: Change to the sqlitedborm directory and run the trufflehog tool.

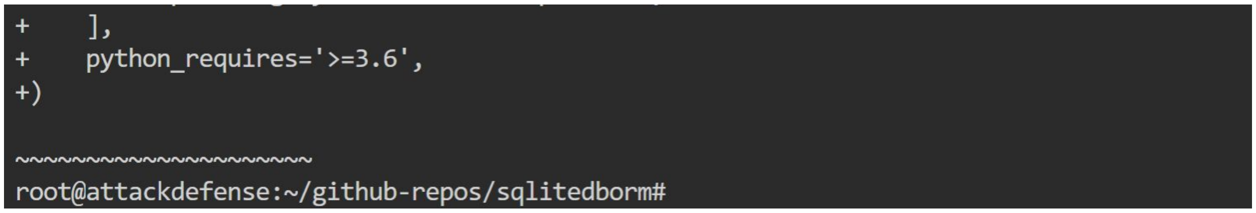
Commands:
cd ~/github-repos/sqlitedborm
trufflehog --regex .



```
root@attackdefense:~/github-repos/sqlitedborm# trufflehog --regex .
~~~~~~~~~~~~~~~~~~~~~
Reason: High Entropy
Date: 2020-09-13 11:33:33
Hash: 599768a3bc9694d493196c08e55f48e1e7af8f68
Filepath: setup.py
Branch: origin/master
Commit: ADD files

@@ -0,0 +1,26 @@
+#!/usr/bin/env python3
+#coding: utf-8
+import setuptools
+
+with open("README.md", "r") as fh:
+    long_description = fh.read()
```



```
+    db_secret_access_key="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
+    url="https://github.com/lcloss/sl3db.git",
+    packages=setuptools.find_packages(),
+    classifiers=[
+        "Programming Language :: Python :: 3",
```



```
+    ],
+    python_requires='>=3.6',
+)

~~~~~~~~~~~~~~~~~~~~~~
root@attackdefense:~/github-repos/sqlitedborm#
```

Issues Detected

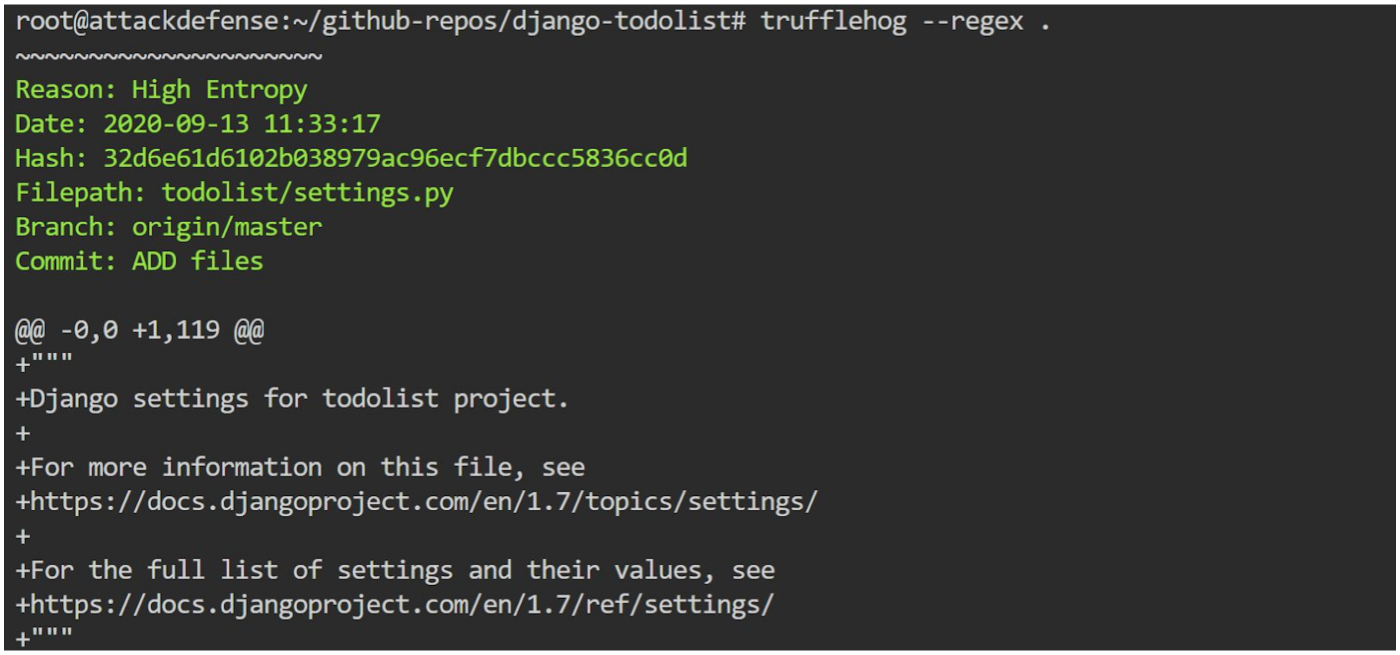
- Found database secret access key

Example 3: django-todolist

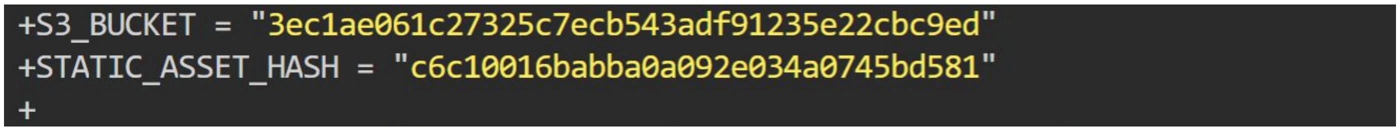
Step a: Change to the django-todolist directory and run the trufflehog tool.

Commands:

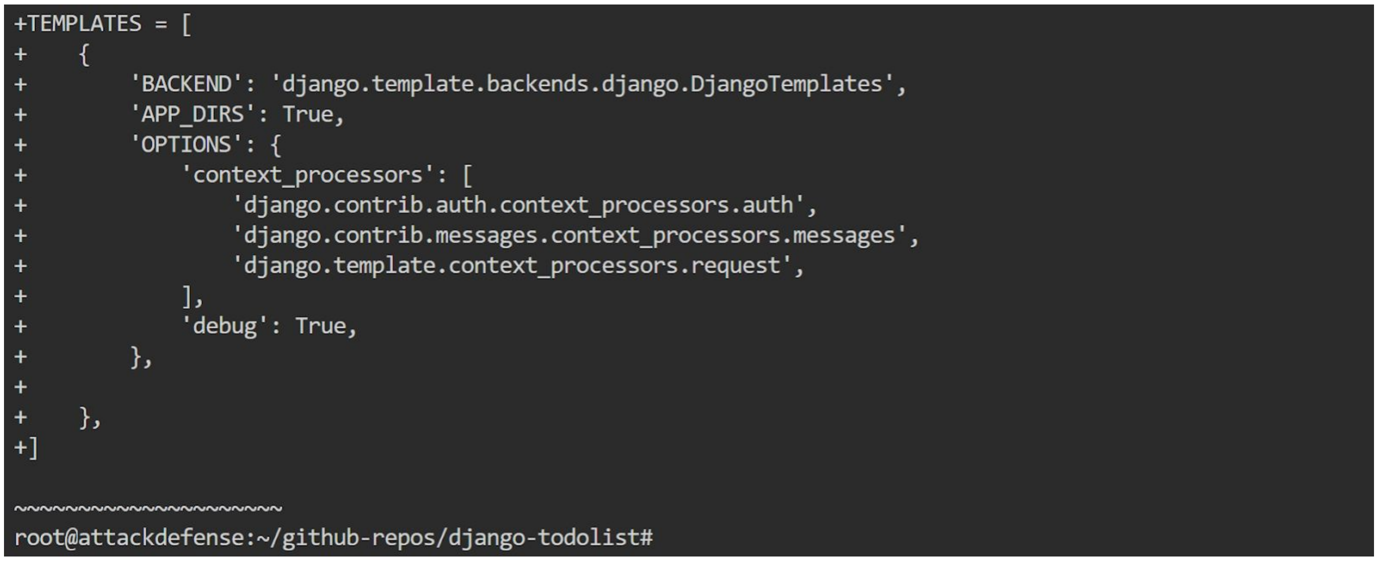
cd ~/github-repos/django-todolist
trufflehog --regex .



```
root@attackdefense:~/github-repos/django-todolist# trufflehog --regex .
~~~~~~~~~~~~~~~~~~~~~~~
Reason: High Entropy
Date: 2020-09-13 11:33:17
Hash: 32d6e61d6102b038979ac96ecf7dbccc5836cc0d
Filepath: todolist/settings.py
Branch: origin/master
Commit: ADD files

@@ -0,0 +1,119 @@
+"""
+Django settings for todolist project.
+
+For more information on this file, see
+https://docs.djangoproject.com/en/1.7/topics/settings/
+
+For the full list of settings and their values, see
+https://docs.djangoproject.com/en/1.7/ref/settings/
+"""
```



```
+S3_BUCKET = "3ec1ae061c27325c7ecb543adf91235e22cbc9ed"
+STATIC_ASSET_HASH = "c6c10016babba0a092e034a0745bd581"
+
```



```
+TEMPLATES = [
+    {
+        'BACKEND': 'django.template.backends.django.DjangoTemplates',
+        'APP_DIRS': True,
+        'OPTIONS': {
+            'context_processors': [
+                'django.contrib.auth.context_processors.auth',
+                'django.contrib.messages.context_processors.messages',
+                'django.template.context_processors.request',
+            ],
+            'debug': True,
+        },
+
+    },
+]

~~~~~~~~~~~~~~~~~~~~~
root@attackdefense:~/github-repos/django-todolist#
```

Issues Detected

- Found S3 Bucket keys in the application

Learnings

Scan web application source code to find sensitive information using the TruffleHog.