ATTACK DEFENSE

by PentesterAcademy

Name	Radamsa: Automated Fuzzing
URL	https://www.attackdefense.com/challengedetails?cid=2058
Type	DevSecOps Basics: Dynamic Code Analysis

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

Radamsa is a general-purpose fuzzer to check if the applications handle malformed or potentially malicious inputs.

A Kali CLI machine (kali-cli) is provided to the user with Radamsa installed on it. The two sample web applications are provided in the home directory of the root user. The user will run these applications and then test these with Radamsa.

Objective: Run Radamsa on provided web applications and find issues!

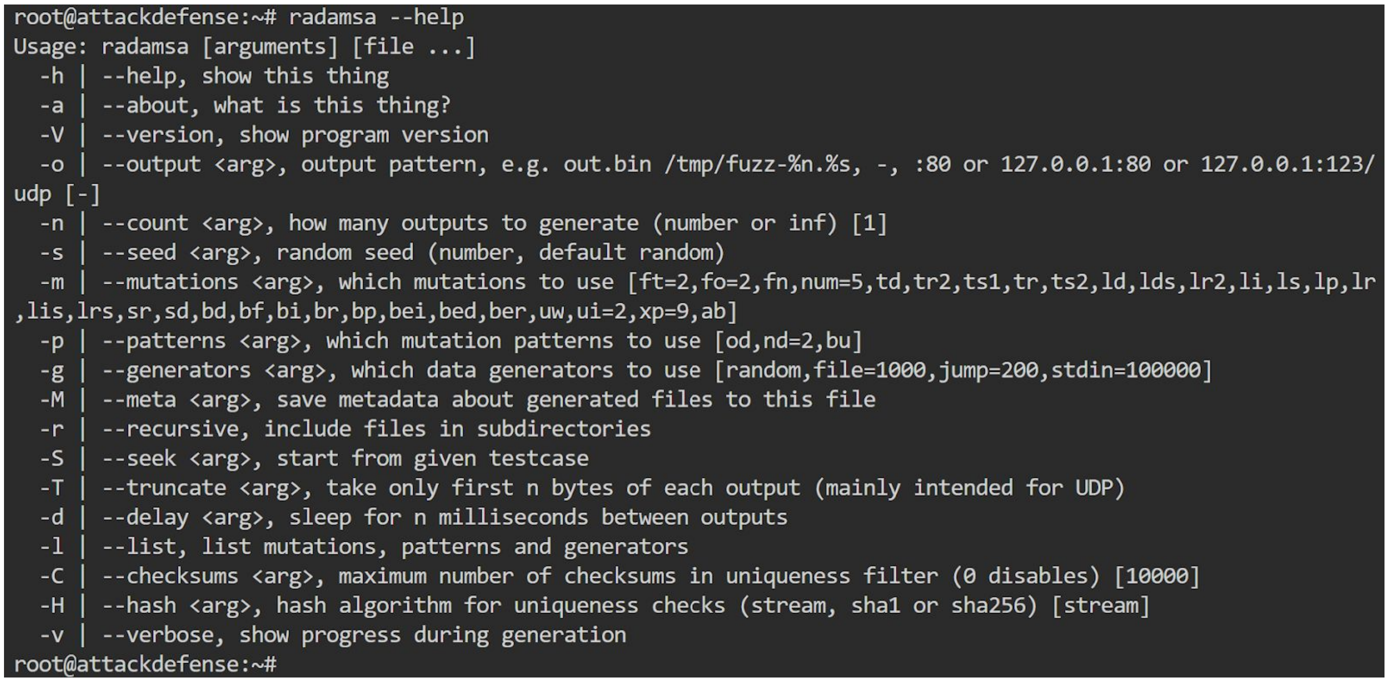
Instructions:

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

Solution

Step 1: Check the available options in Radamsa

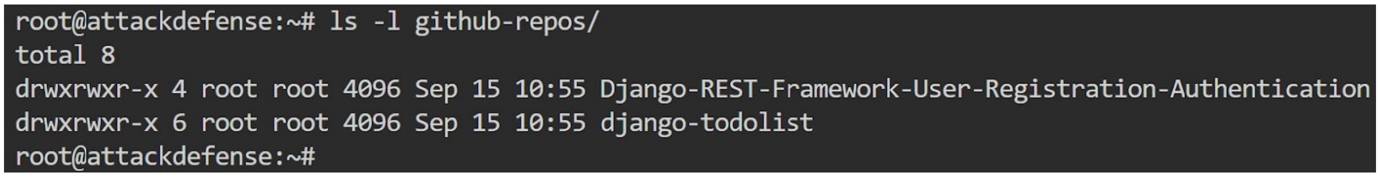
Command: radamsa --help



```
root@attackdefense:~# radamsa --help
Usage: radamsa [arguments] [file ...]
  -h | --help, show this thing
  -a | --about, what is this thing?
  -V | --version, show program version
  -o | --output <arg>, output pattern, e.g. out.bin /tmp/fuzz-%n.%s, -, :80 or 127.0.0.1:80 or 127.0.0.1:123/
udp [-]
  -n | --count <arg>, how many outputs to generate (number or inf) [1]
  -s | --seed <arg>, random seed (number, default random)
  -m | --mutations <arg>, which mutations to use [ft=2,fo=2,fn,num=5,td,tr2,ts1,tr,ts2,ld,lds,lr2,li,ls,lp,lr
,lis,lrs,sr,sd,bd,bf,bi,br,bp,bei,bed,ber,uw,ui=2,xp=9,ab]
  -p | --patterns <arg>, which mutation patterns to use [od,nd=2,bu]
  -g | --generators <arg>, which data generators to use [random,file=1000,jump=200,stdin=100000]
  -M | --meta <arg>, save metadata about generated files to this file
  -r | --recursive, include files in subdirectories
  -S | --seek <arg>, start from given testcase
  -T | --truncate <arg>, take only first n bytes of each output (mainly intended for UDP)
  -d | --delay <arg>, sleep for n milliseconds between outputs
  -l | --list, list mutations, patterns and generators
  -C | --checksums <arg>, maximum number of checksums in uniqueness filter (0 disables) [10000]
  -H | --hash <arg>, hash algorithm for uniqueness checks (stream, sha1 or sha256) [stream]
  -v | --verbose, show progress during generation
root@attackdefense:~#
```

Step 2: Check the source code of provided web applications.

Command: ls -l github-repos



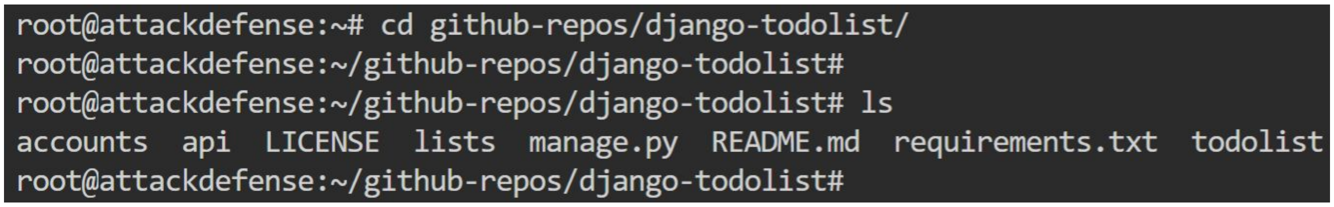
```
root@attackdefense:~# ls -l github-repos/
total 8
drwxrwxr-x 4 root root 4096 Sep 15 10:55 Django-REST-Framework-User-Registration-Authentication
drwxrwxr-x 6 root root 4096 Sep 15 10:55 django-todolist
root@attackdefense:~#
```

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

Example 1: Django Todolist

Step 1: Change to the django-todolist directory and check its contents.

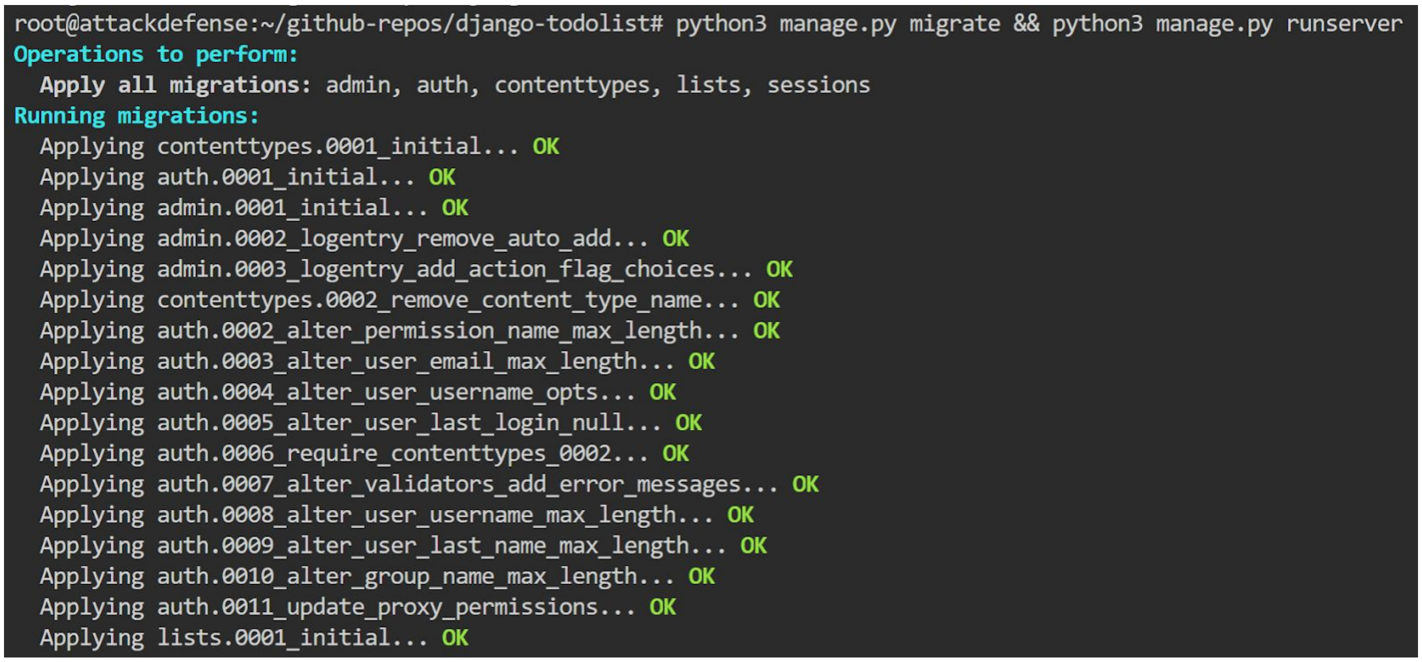
Commands:
cd github-repos/django-todolist
ls

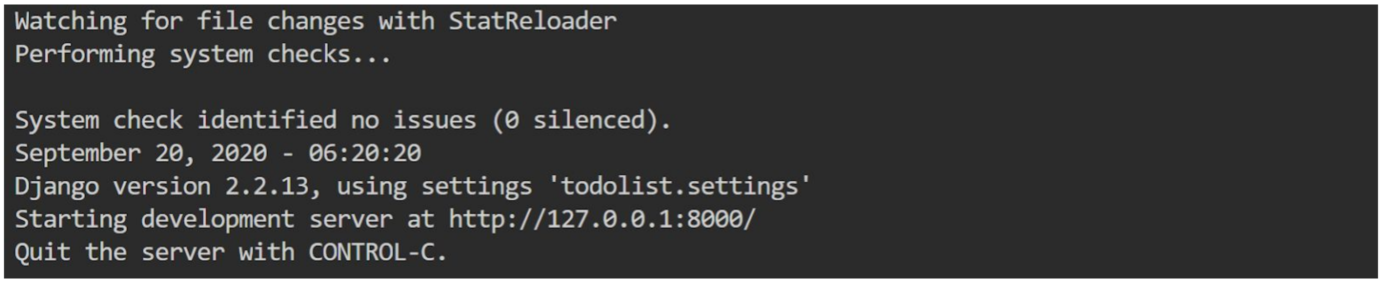

```
root@attackdefense:~# cd github-repos/django-todolist/
root@attackdefense:~/github-repos/django-todolist#
root@attackdefense:~/github-repos/django-todolist# ls
accounts  api  LICENSE  lists  manage.py  README.md  requirements.txt  todolist
root@attackdefense:~/github-repos/django-todolist#
```

Step 2: Start the django server to test the application using the radamsa tool.

Command: python3 manage.py migrate && python3 manage.py runserver

Manage.py migrate - This command will propagate the database schema of the application
Manage.py runserver - This command will start the server at port 8000

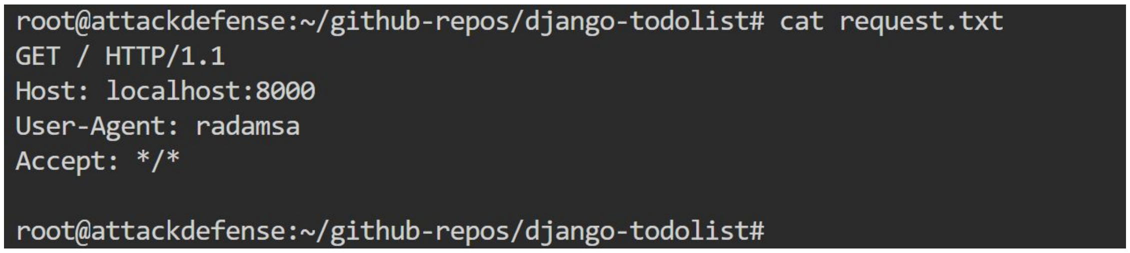

```
root@attackdefense:~/github-repos/django-todolist# python3 manage.py migrate && python3 manage.py runserver
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, lists, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying lists.0001_initial... OK
```


```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
September 20, 2020 - 06:20:20
Django version 2.2.13, using settings 'todolist.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Step 3: Start a new terminal and store a GET request in a file

Request:
GET / HTTP/1.1
Host: localhost:8000
User-Agent: radamsa
Accept: */*

Command: cat request.txt

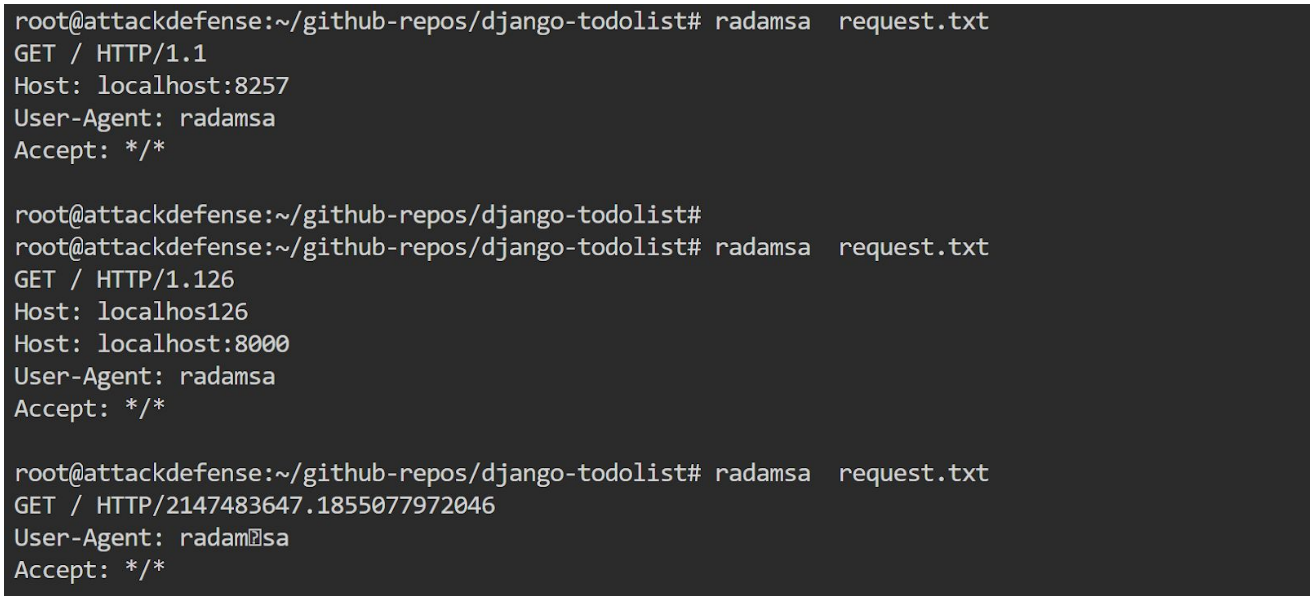


```
root@attackdefense:~/github-repos/django-todolist# cat request.txt
GET / HTTP/1.1
Host: localhost:8000
User-Agent: radamsa
Accept: */*

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

The request will be fuzzed using radamsa tool which then will be passed to the application.

Step 4: Run the radamsa tool and pass the request.txt as an argument

Command: radamsa request.txt

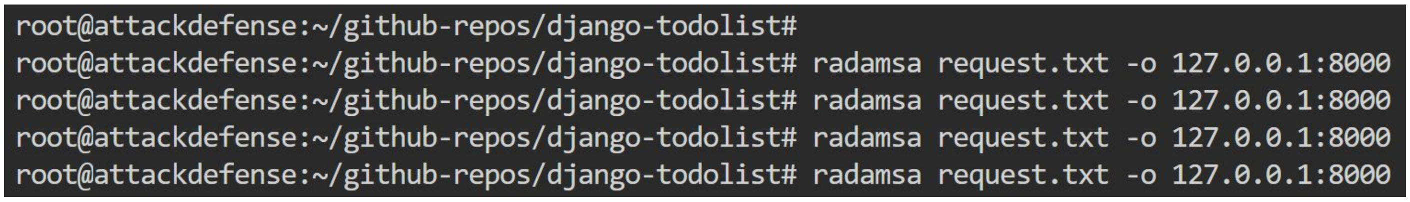


```
root@attackdefense:~/github-repos/django-todolist# radamsa  request.txt
GET / HTTP/1.1
Host: localhost:8257
User-Agent: radamsa
Accept: */*

root@attackdefense:~/github-repos/django-todolist#
root@attackdefense:~/github-repos/django-todolist# radamsa  request.txt
GET / HTTP/1.126
Host: localhos126
Host: localhost:8000
User-Agent: radamsa
Accept: */*

root@attackdefense:~/github-repos/django-todolist# radamsa  request.txt
GET / HTTP/2147483647.1855077972046
User-Agent: radam?sa
Accept: */*
```

The radamsa tool places random strings and data inside the passed file and creates multiple payloads which can be used to fuzz an application.

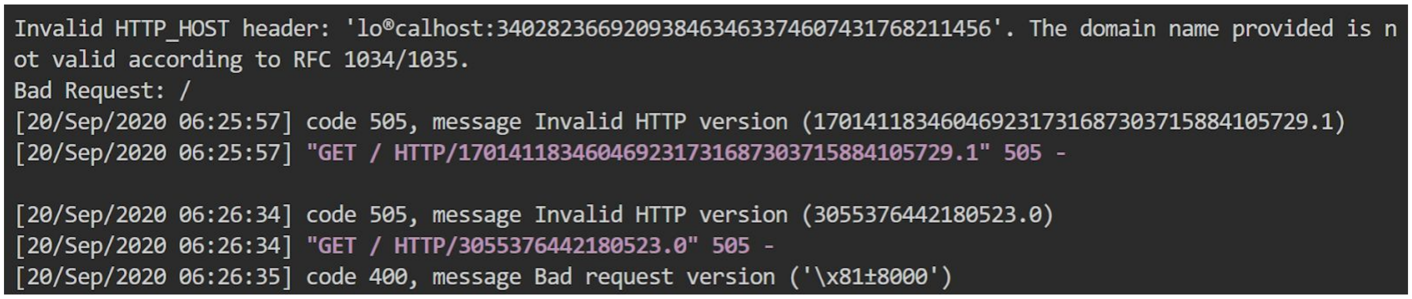
Step 5: Pass the output from radamsa tool to django server running at port 8000

Command: radamsa request.txt -o 127.0.0.1:8000



```
root@attackdefense:~/github-repos/django-todolist#
root@attackdefense:~/github-repos/django-todolist# radamsa request.txt -o 127.0.0.1:8000
root@attackdefense:~/github-repos/django-todolist# radamsa request.txt -o 127.0.0.1:8000
root@attackdefense:~/github-repos/django-todolist# radamsa request.txt -o 127.0.0.1:8000
root@attackdefense:~/github-repos/django-todolist# radamsa request.txt -o 127.0.0.1:8000
```

Execute the command multiple times and check the django server's terminal



```
Invalid HTTP_HOST header: 'lo®calhost:340282366920938463463374607431768211456'. The domain name provided is n
ot valid according to RFC 1034/1035.
Bad Request: /
[20/Sep/2020 06:25:57] code 505, message Invalid HTTP version (170141183460469231731687303715884105729.1)
[20/Sep/2020 06:25:57] "GET / HTTP/170141183460469231731687303715884105729.1" 505 -

[20/Sep/2020 06:26:34] code 505, message Invalid HTTP version (3055376442180523.0)
[20/Sep/2020 06:26:34] "GET / HTTP/3055376442180523.0" 505 -
[20/Sep/2020 06:26:35] code 400, message Bad request version ('\x81±8000')
```

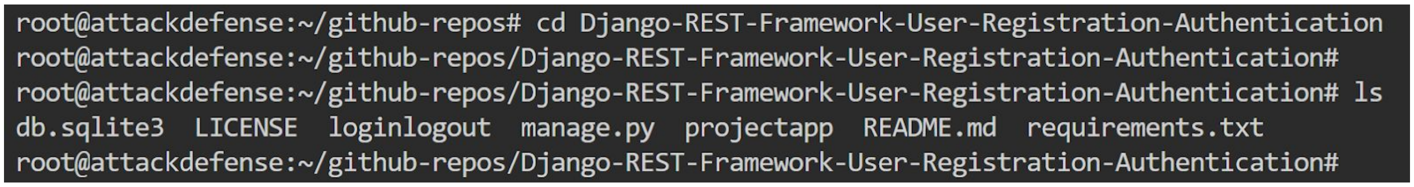
The server received malformed requests from Radamsa and still worked as expected i.e. giving right error codes.

Note: Close the server listening before proceeding to the second application.

Example 2: Django REST Framework User Registration Authentication

Step 1: Change to the Django-REST-Framework-User-Registration-Authentication directory and check its contents.

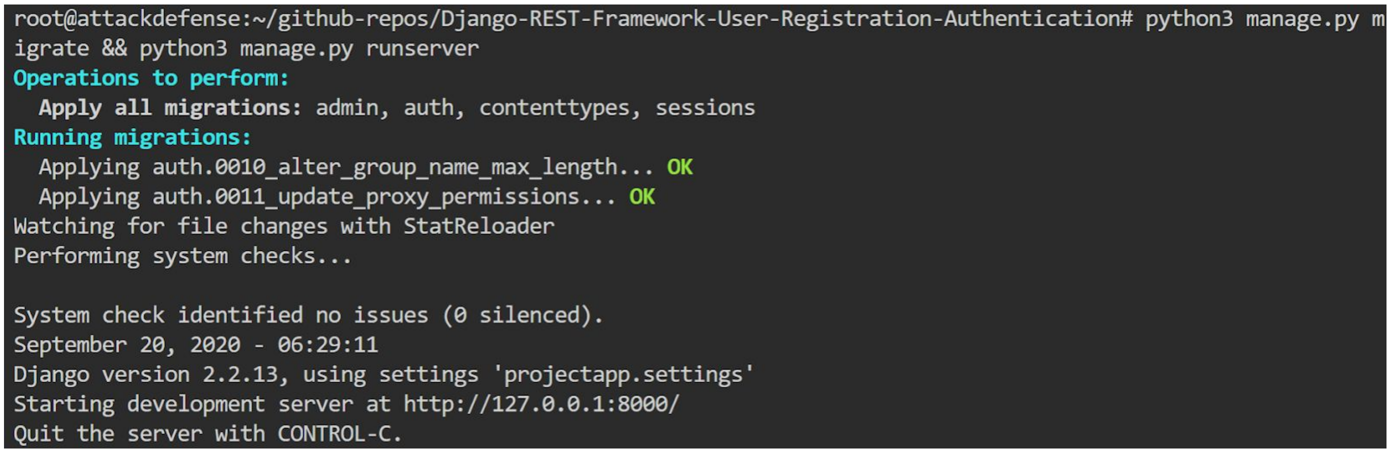
Commands:
cd Django-REST-Framework-User-Registration-Authentication
ls



```
root@attackdefense:~/github-repos# cd Django-REST-Framework-User-Registration-Authentication
root@attackdefense:~/github-repos/Django-REST-Framework-User-Registration-Authentication#
root@attackdefense:~/github-repos/Django-REST-Framework-User-Registration-Authentication# ls
db.sqlite3  LICENSE  loginlogout  manage.py  projectapp  README.md  requirements.txt
root@attackdefense:~/github-repos/Django-REST-Framework-User-Registration-Authentication#
```

Step 2: Start the django server.

Command: python3 manage.py migrate && python3 manage.py runserver



```
root@attackdefense:~/github-repos/Django-REST-Framework-User-Registration-Authentication# python3 manage.py m
igrate && python3 manage.py runserver
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
September 20, 2020 - 06:29:11
Django version 2.2.13, using settings 'projectapp.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Step 3: Start a new terminal and store the POST request in a file

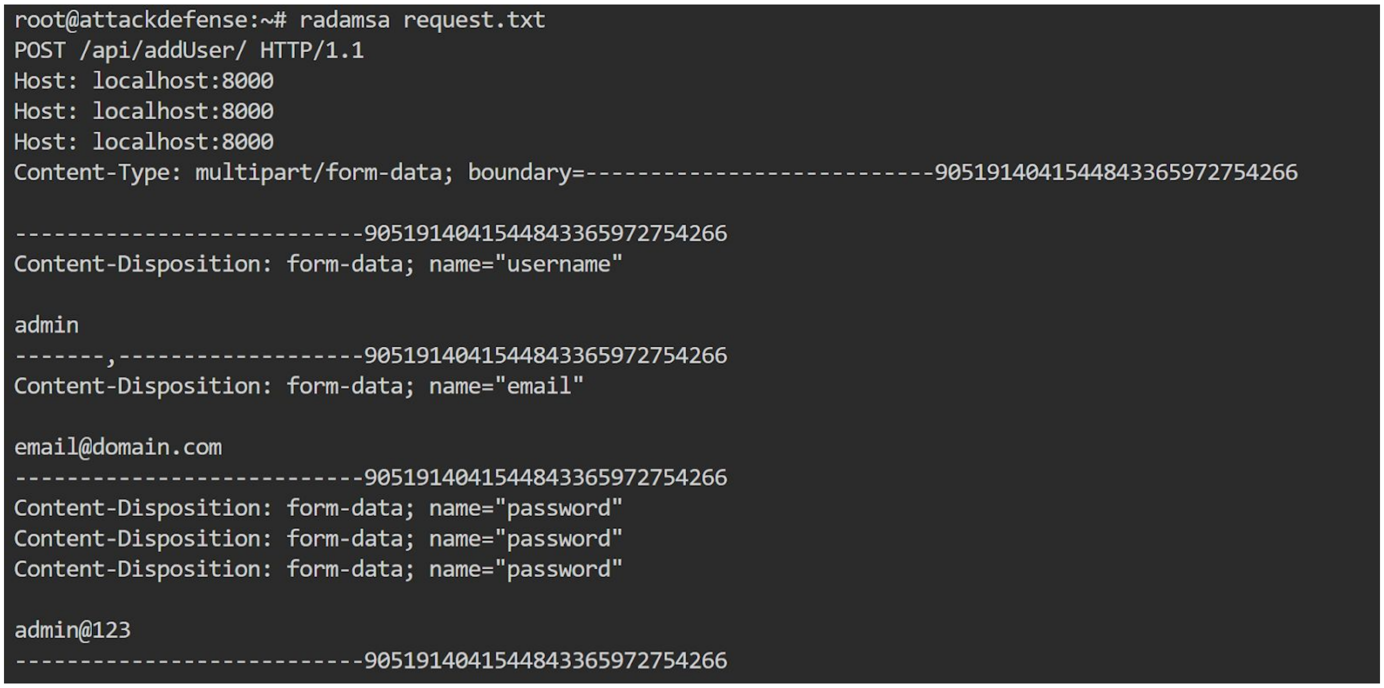
Request:

```
POST /api/addUser/ HTTP/1.1
Host: localhost:8000
Content-Type: multipart/form-data; boundary=---------------------------9051914041544843365972754266

-----------------------------9051914041544843365972754266
Content-Disposition: form-data; name="username"

admin
-----------------------------9051914041544843365972754266
Content-Disposition: form-data; name="email"

email@domain.com
-----------------------------9051914041544843365972754266
Content-Disposition: form-data; name="password"

admin@123
```

--------------------------9051914041544843365972754266

Step 4: Run the radamsa tool and pass the request.txt as an argument

Command: radamsa request.txt

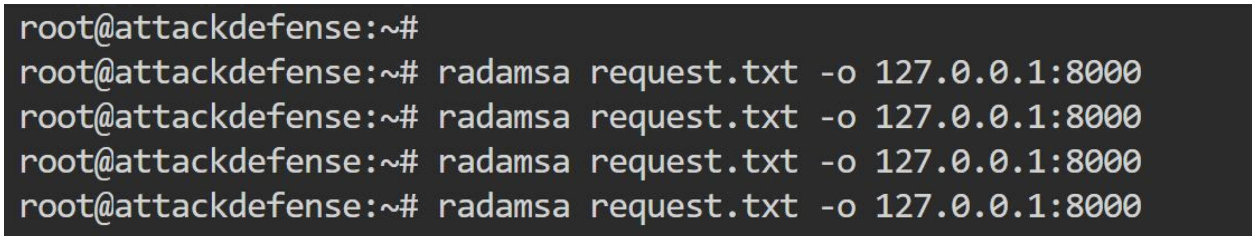


```
root@attackdefense:~# radamsa request.txt
POST /api/addUser/ HTTP/1.1
Host: localhost:8000
Host: localhost:8000
Host: localhost:8000
Content-Type: multipart/form-data; boundary=--------------------------9051914041544843365972754266

--------------------------9051914041544843365972754266
Content-Disposition: form-data; name="username"

admin
-------,------------------9051914041544843365972754266
Content-Disposition: form-data; name="email"

email@domain.com
--------------------------9051914041544843365972754266
Content-Disposition: form-data; name="password"
Content-Disposition: form-data; name="password"
Content-Disposition: form-data; name="password"

admin@123
--------------------------9051914041544843365972754266
```

The radamsa tool will place random data in the marked fields (e.g. username, email, password) to create different payloads.

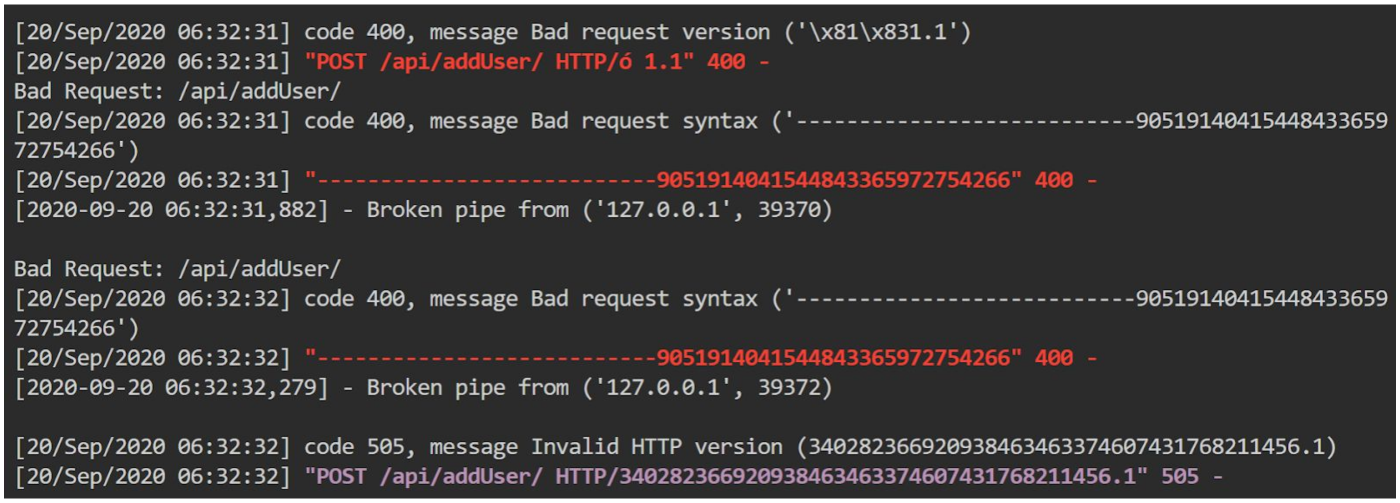
Step 12: Pass the output from radamsa tool to Django server running at port 8000

Command: radamsa request.txt -o 127.0.0.1:8000



```
root@attackdefense:~#
root@attackdefense:~# radamsa request.txt -o 127.0.0.1:8000
root@attackdefense:~# radamsa request.txt -o 127.0.0.1:8000
root@attackdefense:~# radamsa request.txt -o 127.0.0.1:8000
root@attackdefense:~# radamsa request.txt -o 127.0.0.1:8000
```

Execute the command multiple times and check the Django server's terminal



```
[20/Sep/2020 06:32:31] code 400, message Bad request version ('\x81\x831.1')
[20/Sep/2020 06:32:31] "POST /api/addUser/ HTTP/ó 1.1" 400 -
Bad Request: /api/addUser/
[20/Sep/2020 06:32:31] code 400, message Bad request syntax ('--------------------------905191404154484336597
2754266')
[20/Sep/2020 06:32:31] "--------------------------9051914041544843365972754266" 400 -
[2020-09-20 06:32:31,882] - Broken pipe from ('127.0.0.1', 39370)

Bad Request: /api/addUser/
[20/Sep/2020 06:32:32] code 400, message Bad request syntax ('--------------------------905191404154484336597
2754266')
[20/Sep/2020 06:32:32] "--------------------------9051914041544843365972754266" 400 -
[2020-09-20 06:32:32,279] - Broken pipe from ('127.0.0.1', 39372)

[20/Sep/2020 06:32:32] code 505, message Invalid HTTP version (340282366920938463463374607431768211456.1)
[20/Sep/2020 06:32:32] "POST /api/addUser/ HTTP/340282366920938463463374607431768211456.1" 505 -
```

The server received requests from the Radamsa tool and gave corresponding responses.

This tool can be customized and added to the DevSecOps pipeline to make sure that the input fields and request handling is done correctly.

Learnings

Performing fuzzing using the Radamsa utility.