Tools Used for the exploid

Then following tool used

Burpsuite , ZAP Proxy, Manual script , Bandit, Multiple XSS , SQL Payload,

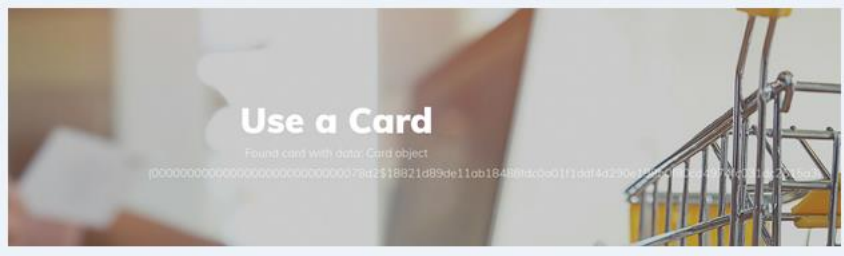
SQL Injection

While reviewing the code base, i found that raw () was used .

card_query = Card.objects.raw('select id from LegacySite_card where data = \'%s\'' % signature)

Teh following query used to obtain the hash vi signature fields and out shows in the screenshot.

"['union select password from LegacySite_user where username = 'admin'--']"}]}



Testing tool.

bandit -r /home/kali/Desktop/AppSecAssignment2.1/GiftcardSite/LegacySite

-t B608,B610,B611 -o test_result.html -f html

Object Relational Mapping (ORM). ORM simply means that a developer does not need to write direct SQL queries, but instead uses the in-built QuerySet APIs. Django then converts the Python query to SQL query and communicates with the database.Query Parameterization and Escaping of parameters ensures that Django applications are protected against SQL injection.

XSS Attack

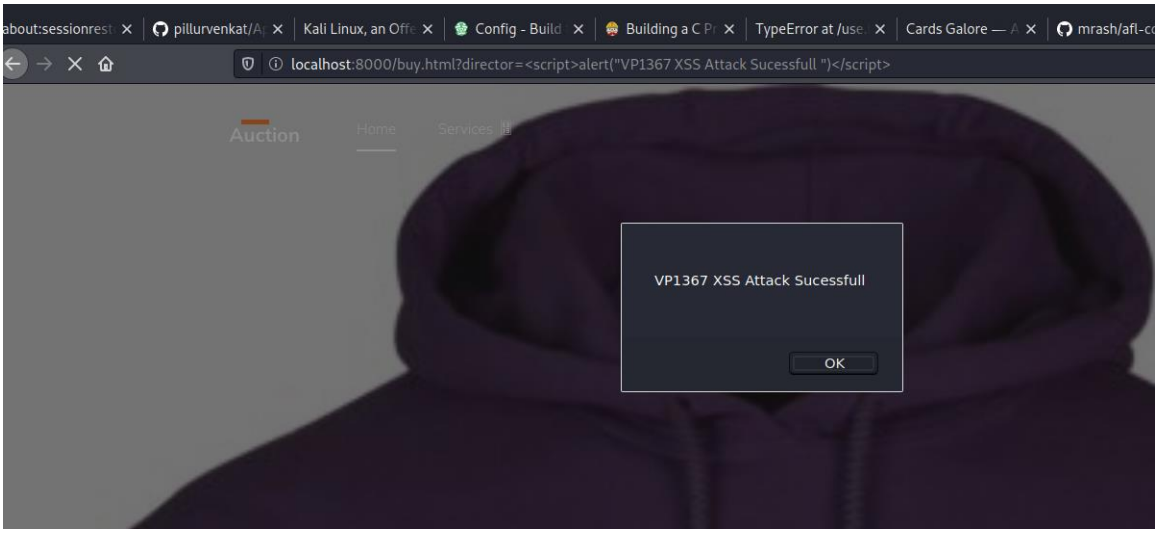
The below screen shot shows the XSS attack was succesful . I was ale to inject client side scripts into the browsers. I have used escape libray to escape the data to secure against XSS.

After fixing the code , the attack was not sucessfull.

espace library imported and used in director path to resolve this issue.

from flask import escape

context['director'] = escape(director)



File path Directory trevesal attack

Issue detail

The **username** parameter is vulnerable to path traversal attacks, enabling read access to arbitrary files on the server.

The payload **../../../../../../../../../../../../../etc/passwd** was submitted in the username parameter. The requested file was returned in the application's response.

Issue background

File path traversal vulnerabilities arise when user-controllable data is used within a filesystem operation in an unsafe manner. Typically, a user-supplied filename is appended to a directory prefix in order to read or write the contents of a file. If vulnerable, an attacker can supply path traversal sequences (using dot-dot-slash characters) to break out of the intended directory and read or write files elsewhere on the filesystem.

This is typically a very serious vulnerability, enabling an attacker to access sensitive files containing configuration data, passwords, database records, log data, source code, and program scripts and binaries.

CSRF Attack

The CSRF attack used along with the sigantaure fieled to use someone card . Actaully i was able to use token to pass this csrf using burbsuite. giftcard view program applied

a) activated the django.middleware.csrf.CsrfViewMiddleware in the settings.py file.

b) added a context dictionary in RequestContext object and pass them through render_to_response() method

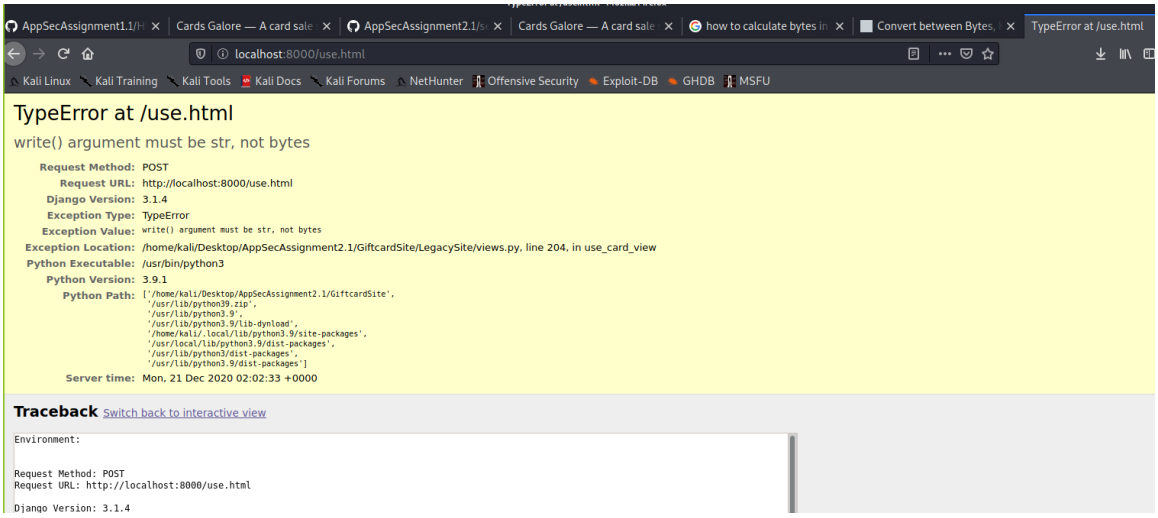
c) added a csrf_token value as an element in the context dictionary object.

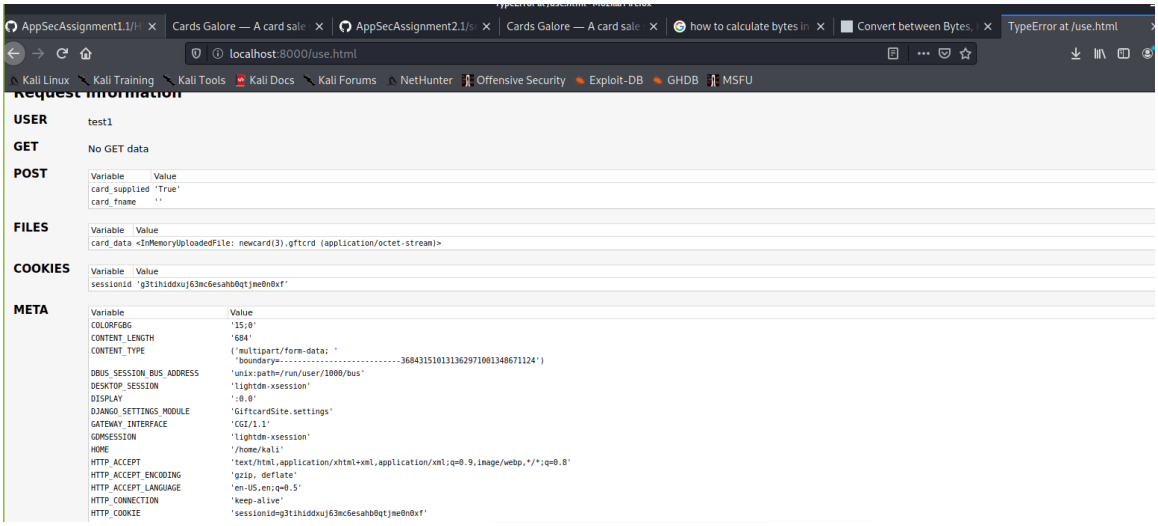
d) Usedthe csrf_token in the HTML template file as a hidden field value that will be sent along with the form post request.

```
context_ditc ={}


conext_dict.update(csrf(request))
```

Error handling - i disovcered impropoer /poor error handling and teh attacker can leverage the information to craft especail attack. Also senstive infomration leak which is consider data leak. ettc. This is one of the critical vulnerabilities





Requirement - Use encryption using django crypto

I have made following changes with models.py

from django.db import models

from django_cryptography.fields import encrypt

from django_cryptography.fields import EncryptedField

I will be using only fields for encryption

Adding Encryption

data = encrypt(models.BinaryField(unique=True))

Key management in settings.py

Updated the key management in setting.py

To use an encrypted field in a Django model, use one of the fields from the

``cryptographic_fields`` module:

INSTALLED_APPS = [

    # Adding

'cryptographic_fields',

Django management command ``generate_encryption_key`` provided

with the ``cryptographic_fields`` library. Used this command to generate a new

encryption key to set as ``settings.FIELD_ENCRYPTION_KEY``.

./manage.py generate_encryption_key

print an encryption key to the terminal, which is configured in the environment settings file.