第七期

业务安全实战攻防

平安SRC线上沙龙系列主题活动

- 时间：2024.11.22 14:00~17:00

主办方

平安安全应急响应中心 PINGAN Security Response Center

合作伙伴

补天 漏洞响应平台

百度安全应急响应中心 Baidu Security Response Center

度小满安全应急响应中心 Du Xiaoman Security Response Center

无糖信息

DAY 1

TIMELINESEC

北山学院

Arthas+AI，新视角看代码审计

Kuiplatain

Timeline Sec 成员

目录

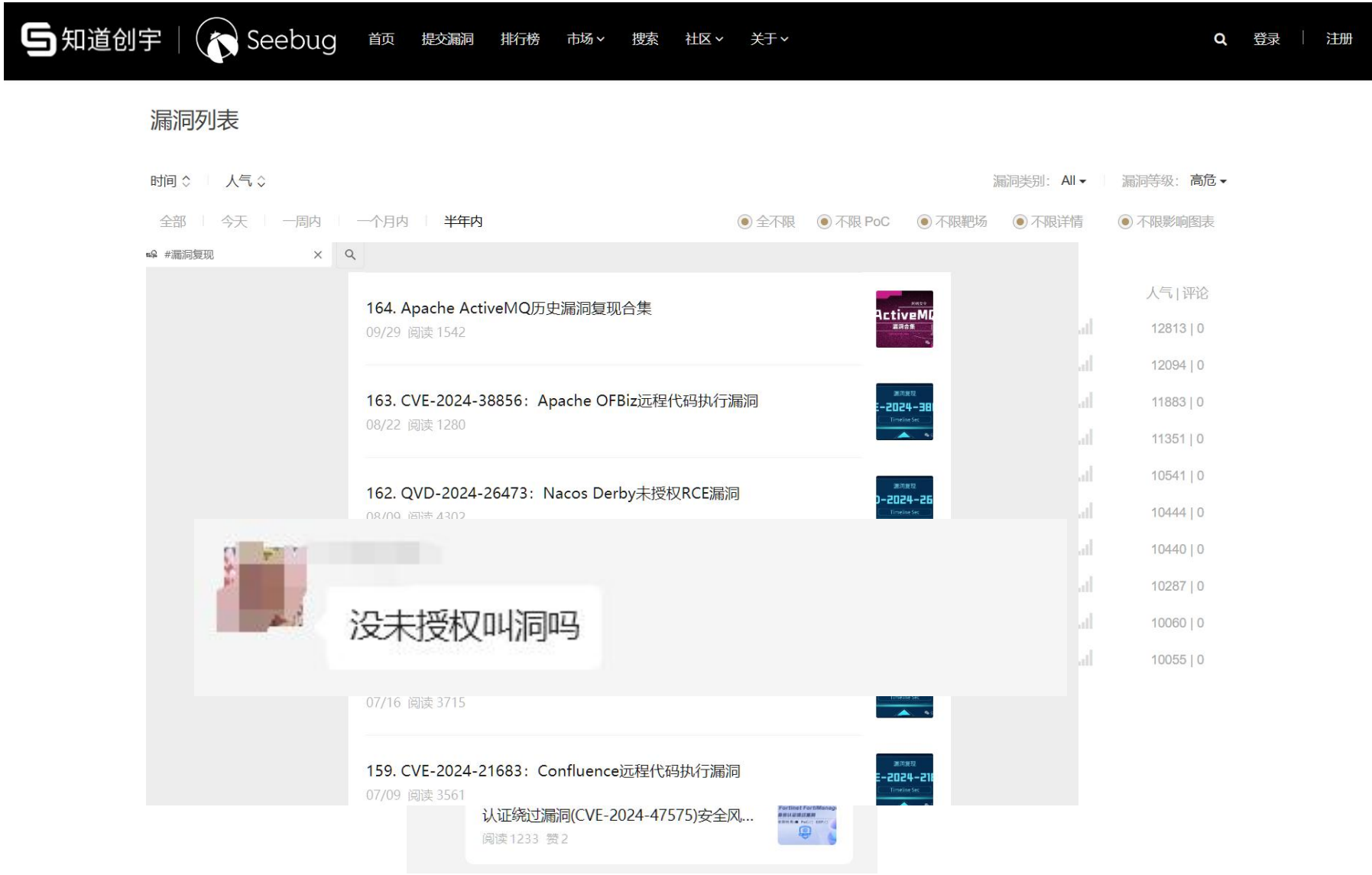
PART ONE

代码审计现状

目前挖掘未授权RCE的套路

近半年的人气高危漏洞：

1. 未授权RCE
2. 信息泄露
3. 权限绕过



开放更多权限给用户 => 暴漏更多功能点给用户 => 开放绕过权限就接手服务器

怎么绕过权限验证?

1. bypass auth filter
2. 白名单路由
3. 第三方依赖组件

解决方案？

PINGANSECURITYRESPONSECENTER

So，为了解决上面的问题开源工具，我们有哪些选择：

1. codeql
2. Tai-e
3. TABBY
4. foritify

用SQL语句实现逻辑检测?

Arthas的外道用法

如何将Arthas与代码审计有效结合，减少时间成本

在开始之前，先让我们回忆下内存马的分类：

1. filter
2. controller
3. servlet
4. listener
5. agent（非常的amamzing）

// etc

1. 功能丰富：涵盖了线程、内存、GC、类加载、类结构、方法调用等多个方面的诊断和监控功能。

2. 操作简便：支持命令行和WEB界面两种方式，操作直观易懂。

3. 轻量级：可以在不停止应用的情况下附着到目标JVM上，进行在线诊断。

4. 对生产环境友好：调试过程对生产环境的影响较小，不会导致应用程序崩溃或停机。

5. 动态调试：支持动态追踪方法调用和修改代码，提高调试效率。

arthas的有趣用法

PINGANSECURITYRESPONSECENTER

1. jad – 反编译指定已加载类的源码 ×

   随看随调√

2. ognl – 执行 ognl 表达式。 ×

   直接调试线上函数√

获取全部servlet

vmtool --action getInstances --className org.springframework.web.servlet.DispatcherServlet --express 'instances[0].getHandlerMappings().{? #this instanceof org.springframework.web.servlet.handler.AbstractHandlerMethodMapping}.{#this.mappingRegistry.registry.entrySet().{"Result: ll"+#this.value.mapping+": "+#this.value.handlerMethod.method+"ll"}}.toString'

T2Front

e-chart绘制api路径图+filter

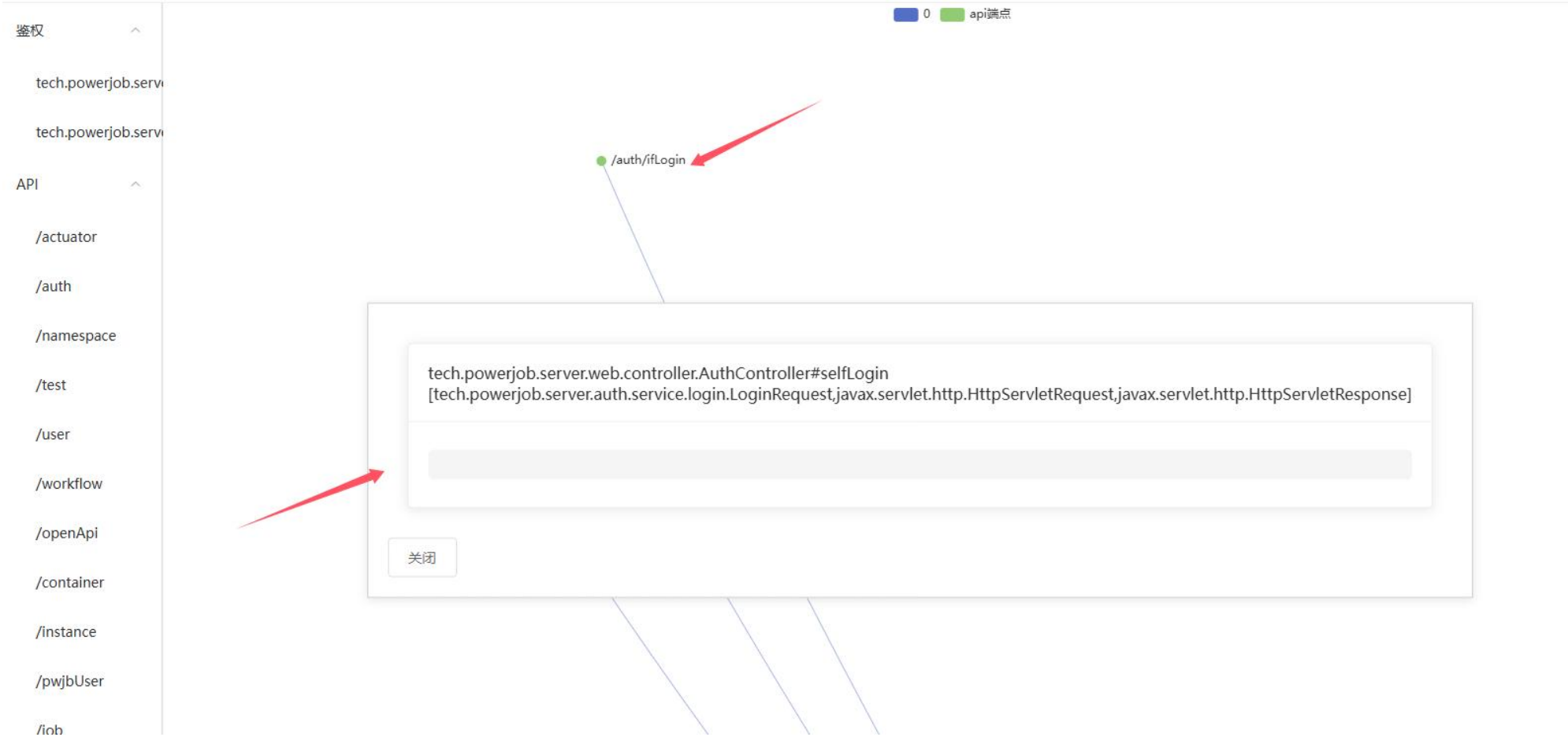
arthas反编译代码+AI分析代码给出参考

鉴权分析

PINGANSECURITYRESPONSECENTER

自动关联filter与全量api接口，使用AI将依赖版本和代码关联分析预测是否存在问题（这部分数据有点问题，后面我会更新下

点击API接口自动反编译对应的函数以及AI注释（这部分数据有点问题，后面我会更新下）



依赖分析

PINGANSECURITYRESPONSECENTER

搜集了部分依赖特定版本下的漏洞，可以手动查看也会用来作为AI关联分析的一部分（这部分数据有点问题，后面我会更新下）

jar包	版本	漏洞
spring-boot-starter-web	2.7.2	asd
spring-boot-starter	2.7.2	asd
spring-boot	2.7.2	asd
spring-boot-autoconfigure	2.7.2	asd
spring-boot-starter-logging	2.7.2	asd
logback-classic	1.2.11	asd
logback-core	1.2.11	asd
log4j-to-slf4j	2.17.2	asd
jul-to-slf4j	1.7.36	asd
jakarta.annotation-api	1.3.5	asd
snakeyaml	1.30	asd
spring-boot-starter-json	2.7.2	asd
jackson-databind	2.13.3	asd

一些更深入的feature

关于为什么这个小工具会拖这么久

当然，上面那么简单的功能为什么会让我浪费那么多时间去思考怎么写呢?

答案就是，污点传播+AI

实际情况

PINGANSECURITYRESPONSECENTER

chatgpt出来那年就有人去测试能不能做污点传播，当年的效果是，不行。

几年后的今天呢，我依旧可以告诉各位，不行。

个人的理解 PINGANSECURITYRESPONSECENTER

- 虽然AI的确是万能函数，但transformer架构的模型本质是在做预测
- 需求要分析的环节越多就越容易出错
- 类似于他能处理的问题复杂度有一个上限。

结合我个人的一些测试结果发现：

- **将AI拿来处理涉及语义理解的部分能有效减少开发难度**，如下的两种产品。

RASP

RASP防御的核心就是在Web应用程序
执行关键的Java API之前插入防御逻辑，
从而控制原类方法执行的业务逻辑。

https://www.javasec.org/java-rasp/

ISAT

IAST通过插桩技术（Instrumented）收集、监控Web应用程序运行时的函数执行、数据传输，并与服务端（server）进行实时交互，高效、准确地识别安全缺陷及漏洞。

基于开发语言自身的插桩技术，在软件运行过程中采用污点传播技术跟踪用户输入数据（污点）执行流程，来检查安全漏洞。

那跟IAST有什么区别？ --某位不愿透漏姓名的jabaer如是说道

更激进

arthas有ognl调用函数，反编译函数，修改函数和查看调用位置变量的能力。

那么怎么搞定污点传播呢?

众所周知，污点传播很麻烦的点在于中间路径分析，这部分如果每种bypass策略都编写规则会变得无比冗余。

因此我的思路如下：

先粗找污点传播链，然后用AI删除函数中与污点传播无关的部分。

目前有两个思路

1. 给arthas加功能，可以忽略requests参数，直接进入函数内操作；

2. 构造http请求，用IAST打流量的方式触发。

然后AI删去无关紧要的部分，直接打进去，然后在断点位置分析是否符合预期，bypass的策略交给AI去做。

feature 1

直接删除原始函数中与传播链无关的参数

当然这部分会非常复杂，对于有些涉及到对象初始化，全局变量之类的函数调用链很大概率会误报。

feature 2

将其他filter置空，然后在目标dofilter位置下断点，由算法+AI得出可能的pass路径进行fuzz测试

Thanks

非常感谢七安师傅在前端方面的指导和longofo师傅在一些技术细节的指点

THANKS