

WebSocket 内存马，一种新型内存马技术 - 先知社区

“ 先知社区，先知安全技术社区

1. 前言

WebSocket 是一种全双工通信协议，即客户端可以向服务端发送请求，服务端也可以主动向客户端推送数据。这样的特点，使得它在一些实时性要求比较高的场景效果斐然（比如微信朋友圈实时通知、在线协同编辑等）。主流浏览器以及一些常见服务端通信框架（Tomcat、netty、undertow、webLogic 等）都对 WebSocket 进行了技术支持。

2. 版本

2013 年以前还没出 JSR356 标准，Tomcat 就对 Websocket 做了支持，自定义 API，再后来有了 JSR356，Tomcat 立马紧跟潮流，废弃自定义的 API，实现 JSR356 那一套，这就使得在 Tomcat7.0.47 之后的版本和之前的版本实现方式并不一样，接入方式也改变了。

JSR356 是 java 制定的 websocket 编程规范，属于 Java EE 7 的一部分，所以要实现 websocket 内存马并不需要任何第三方依赖

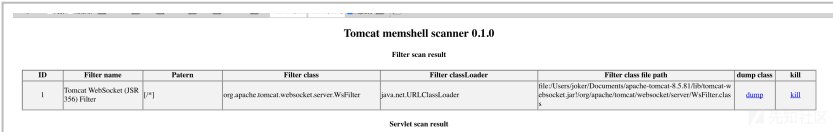
3. 服务端实现方式

(1) 注解方式

```
@ServerEndpoint(value = "/ws/{userId}", encoders =  
    {MessageEncoder.class}, decoders = {MessageDecoder.class},  
    configurator = MyServerConfigurator.class)
```

Tomcat 在启动时会默认通过 WsSci 内的 ServletContainerInitializer 初始化 Listener 和 servlet。然后再扫描 classpath 下带有 @ServerEndpoint 注解的类进行 addEndpoint 加入 websocket 服务

所以即使 Tomcat 没有扫描到 @ServerEndpoint 注解的类，也会进行 Listener 和 servlet 注册，这就是为什么所有 Tomcat 启动都能在 memshell scanner 内看到 WsFilter



ID	Filter name	Pattern	Filter class	Filter class loader	Filter class file path	dump class	kill
1	Tomcat WebSocket (JSR 356) Filter	[*]	org.apache.tomcat.websocket.server.WsFilter	java.net.URLClassLoader	file:/Users/joker/Documents/apache-tomcat-8.5.81/lib/tomcat-websocket.jar/org/apache/tomcat/websocket/server/WsFilter.class	dump	kill

(<https://xzfile.aliyuncs.com/media/upload/picture/20220715175317-f3a68f56-0423-1.png>)

(2) 继承抽象类 Endpoint 方式

继承抽象类 Endpoint 方式比加注解 @ServerEndpoint 方式更麻烦，主要是需要自己实现 MessageHandler 和 ServerApplicationConfig 。 @ServerEndpoint 的话都是使用默认的，原理上差不多，只是注解更自动化，更简洁

可以用代码更方便的控制 ServerEndpointConfig 内的属性

```
ServerEndpointConfig serverEndpointConfig =
    ServerEndpointConfig.Builder.create(WebSocketServerEndpoint3.class,
        "/ws/{userId}").decoders(decoderList).encoders(encoderList)
        .configurator(new MyServerConfigurator()).build();
```

3.websocket 内存马实现方法

之前提到过 Tomcat 在启动时会默认通过 WsSci 内的 ServletContainerInitializer 初始化 Listener 和 servlet。然后再扫描 classpath 下带有 @ServerEndpoint 注解的类进行 addEndpoint 加入 websocket 服务

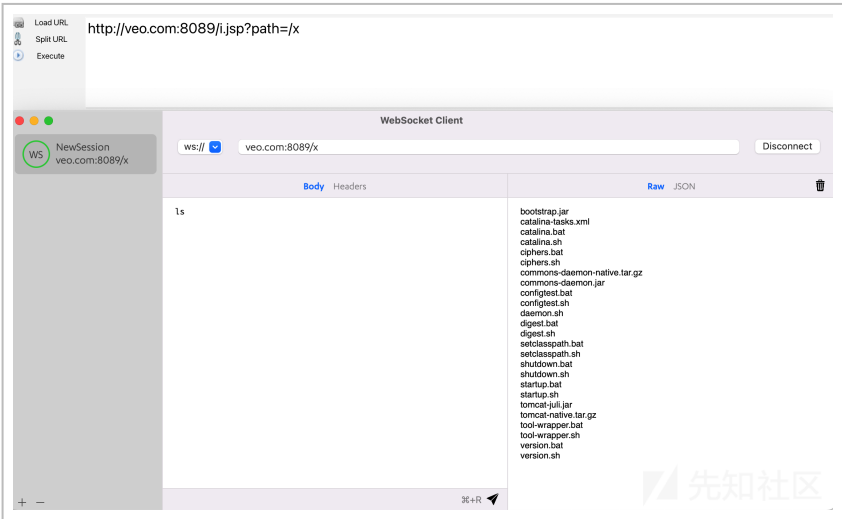
那如果在服务启动后我们再 addEndpoint 加入 websocket 服务行不行呢？答案是肯定的，而且非常简单只需要三步。创建一个

ServerEndpointConfig, 获取 ws ServerContainer, 加入 ServerEndpointConfig, 即可

```
ServerEndpointConfig config =
ServerEndpointConfig.Builder.create(EndpointInject.class,
"/ws").build();
ServerContainer container = (ServerContainer)
req.getServletContext().getAttribute(ServerContainer.class
.getName());
container.addEndpoint(config);
```

4. 效果

首先利用 i.jsp 注入一个 websocket 服务, 路径为 / x, 注入后 利用 ws 连接即可执行命令



(<https://xzfile.aliyuncs.com/media/upload/picture/20220715175328-fa4c458a-0423-1.png>)

且通过 memshell scanner 查询不到任何异常 (因为根本就没注册新的 Listener、servlet 或者 Filter)

Tomcat memshell scanner 0.1.0							
Filter scan result							
ID	Filter name	Pattern	Filter class	Filter classLoader	Filter class file path	dump class	kill
1	Tomcat WebSocket (SRV 506) Filter	/^/	org.apache.tomcat.websocket.server.WsFilter	java.net.URLClassLoader	file:/Users/joker/Documents/apache-tomcat-8.5.81/lib/tomcat-websocket.jar/org/apache/tomcat/websocket/server/WsFilter.class	dump	kill
Servlet scan result							
ID	Servlet name	Pattern	Servlet class	Servlet classLoader	Servlet class file path	dump class	kill
1	jsp	*.jspx	org.apache.jasper.servlet.JspServlet	java.net.URLClassLoader	file:/Users/joker/Documents/apache-tomcat-8.5.81/lib/jasper.jar/org/apache/jasper/servlet/JspServlet.class	dump	kill
2	jsp	*.jsp	org.apache.jasper.servlet.JspServlet	java.net.URLClassLoader	file:/Users/joker/Documents/apache-tomcat-8.5.81/lib/jasper.jar/org/apache/jasper/servlet/JspServlet.class	dump	kill
3	Test	/Test	com.memshell.servletTestPackage.ServletTest	org.apache.catalina.loader.ParallelWebappClassLoader	file:/Users/joker/IdeaProjects/memshell/src/main/webapp/WEB-INF/classes/com/memshell/servletTestPackage/ServletTest.class	dump	kill
4	default	/	org.apache.catalina.servlets.DefaultServlet	java.net.URLClassLoader	file:/Users/joker/Documents/apache-tomcat-8.5.81/lib/catalina.jar/org/apache/catalina/servlets/DefaultServlet.class	dump	kill

(<https://xzfile.aliyuncs.com/media/upload/picture/20220715175342-02baaafe-0424-1.png>)

5. 代理

WebSocket 是一种全双工通信协议，它可以用来做代理，且速度和普通的 TCP 代理一样快，这也是我研究 websocket 内存马的原因。

例如有一台不出网主机，有反序列化漏洞。

以前在这种场景下，可能会考虑上 reGeorg 或者利用端口复用来搭建代理。

现在可以利用反序列化漏洞直接注入 websocket 代理内存马，然后直接连上用上全双工通信协议的代理。

注入完内存马以后，使用 Gost: <https://github.com/go-gost/gost> (<https://github.com/go-gost/gost>) 连接代理

```
./gost -L "socks5://:1080" -F "ws://127.0.0.1:8080?path=/proxy"
```

然后连接本地 1080 端口 socks5 即可使用代理

6. 多功能 shell 实现

想要使用 ws 马首先得支持连接 ws 协议的工具，目前市面的 webshell 管理工具都要从源码上修改才能支持 ws 协议

具体实现过程也并不复杂，相当于只是替换了协议，内容其实可以不变。例如给出的哥斯拉支持样例，基本逻辑并没发生改变，只是协议变了

还有一个问题是 ws 马必须先注入再连接，并不能直接连接 jsp 马。

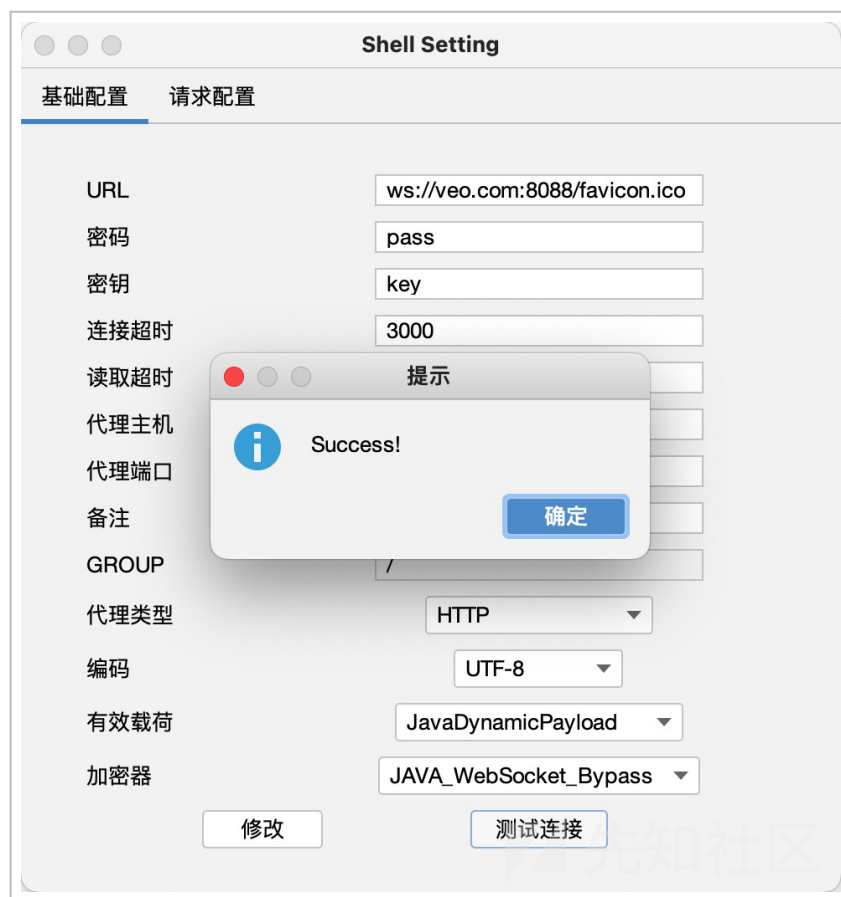
然而例如哥斯拉的 jsp 马本身就是支持远程代码执行，那么 jsp 马其实可以保持不变就用哥斯拉原版，但发送 class 要修改，先发送过主生初始化注册 ws 马的 class，连上 ws 以后再初始化

又通过公钥初始化/注册 ws 用的 class，通过 ws 以/公钥初始化
恶意 class，多一步，第二步连接的时候使用 ws 连接。



(<https://xzfile.aliyuncs.com/media/upload/picture/20220715175553-50a36ac6-0424-1.jpg>)

如果是内存注入的内存马则不需要连接 jsp，直接连接 ws



(<https://xzfile.aliyuncs.com/media/upload/picture/20220715175445-27d117d8-0424-1.jpg>)

版权声明

完整代码：<https://github.com/veo/wsMemShell>
(<https://github.com/veo/wsMemShell>)

本文章著作权归作者所有。转载请注明出处！

<https://github.com/veo> (<https://github.com/veo>)

