H2 Plaintext password

H2 has a web-based admin console that can be started via the CLI. One of the arguments is *-webAdminPassword*, which allows the user to specify the password in plaintext for the web admin console. Consequently, a malicious local user or an attacker that has obtained local access through some means would be able to get the password for the H2 web admin console by looking at the running processes.

VERSIONS AFFECTED: From 1.4.198 to the latest (from the time of this report, version 2.1.214 is also vulnerable)

SCORE: 8.4 CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CWE-200: Exposure of Sensitive Information to an Unauthorized Actor

Proof of Concept

Consider the following docker file and h2.server.properties file used by docker.

- Dockerfile:

```
FROM openjdk:11

ENV DOWNLOAD https://github.com/h2database/h2database/releases/download/version-2.1.214/h2-2022-06-13.zip
ENV DATA_DIR /opt/h2-data

RUN groupadd --gid 1000 guest \
    && useradd --uid 1000 --gid guest --shell /bin/bash --create-home guest

RUN mkdir -p ${DATA_DIR}
RUN curl -L ${DOWNLOAD} -o h2.zip
RUN unzip h2.zip -d /opt/
RUN rm h2.zip

COPY h2.server.properties /root/.h2.server.properties
WORKDIR /opt/h2-data
EXPOSE 8090 1521

CMD java  -cp /opt/h2/bin/h2*.jar org.h2.tools.Server ${H2_OPTIONS}
```

- h2.server.properties file:

```
webAdminPassword=123456
webSSL=false
webAllowOthers=true
webPort=8090
tcp=true
tcpAllowOthers=true
tcpPort=1521
0=Generic H2 (Embedded)|org.h2.Driver|jdbc\:h2\:test|sa
1=Generic H2 (Server) Old Information Schema|org.h2.Driver|jdbc\:h2\:tcp\://localhost\:1521/test;OLD_INFORMATION_SCHEMA\=TRUE|sa
2=Generic H2 (Server) MariaDB Mode|org.h2.Driver|jdbc\:h2\:tcp\://localhost\:1521/test;MODE\=MariaDB;|sa
3=Generic H2 (Server)|org.h2.Driver|jdbc\:h2\:tcp\://localhost\:1521/test|sa
```

With these two files, we run the docker file with something similar to this:

```
docker build --no-cache -t h2poc .
docker run -d -p 1521:1521 -p 8090:8090 -e H2_OPTIONS="-webAdminPassword 1234" --name=h2poc h2poc
```

Once the docker instance is running, we can execute the following command in the instance running as a guest user:

```
docker exec --user guest h2poc ps -aux
```

This will reveal the process list and the arguments used to run them, including h2 with the password in plaintext in some form like this:

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   2420   568 ?        Ss   19:30   0:00 /bin/sh -c java  -cp /opt/h2/bin/h2*.jar org.h2.tools.Server ${H2_OPTIONS}
root         7  3.7  3.2 3883748 65532 ?       Sl   19:30   0:01 java -cp /opt/h2/bin/h2-2.1.214.jar org.h2.tools.Server -webAdminPassword 1234
guest       62  0.0  0.1   8592  3280 ?        Rs   19:31   0:00 ps -aux
```

As an aside, it is interesting to note that the CLI argument overwrites what is specified in the potentially better-secured configuration file.

Impact

According to a developer of H2: "Users with ADMIN privileges may do anything with your JVM and system by design".

This means that any user on a system running H2 that has been started with the `webAdminPassword` argument can get the administrator password and escalate their privileges to perform any action on the system that the owner of the H2 process would be able to.

Occurrences

https://github.com/h2database/h2database/blob/96832bf5a97cdc0adc1f2066ed61c54990d66ab5/h2/src/main/org/h2/server/web/WebServer.java#L346-L347

Maintainer answer

This is not a vulnerability of H2 Console, this is an example of how not to use it. I think there is nothing to do with it on H2 side. Passwords should never be passed on the command line and every qualified DBA or system administrator is expected to know that.

`webAdminPassword` is actually not related to privileges of users or users with ADMIN privileges. This is a password for server management tools of H2 Console (backup, restore, etc. without logon to a particular database itself) and it is rarely appropriate to use these tools, especially in production, H2 Console is a tool for developers. And the most of developers don't know that these management tools ever exist. But if somebody knows them and needs them for a some special reason, password can be specified in `.h2.server.properties`. Documentation doesn't suggests any other ways. H2 rewrites password is this configuration file with its salted hash if configuration is saved from settings of H2 Console, so it is possible to use this secure format instead of plaintext form. But people can always find new insecure places for sensitive settings and it isn't possible to imagine all possible ways to do that.

The mentioned `WebServer` class doesn't know how exactly password was specified. For example, application can read some password from its own secure configuration and pass it safely directly to this class. Unfortunately, I don't see a way to detect insecure configurations automatically.

Suggested fix provided

https://github.com/h2database/h2database/compare/HEAD...sniffernandez:no-plaintext-password

We agree that passwords should never be passed on the command line and that H2 be expected to account for creative password storage solutions.

What we're trying to report is related specifically to these lines here. H2 is specifically accepting a `-webAdminPassword` argument on the command line to use as the web console's password on execution. In addition, because the line is after where H2 reads the more secure config file, the command line argument will then overwrite whatever is written in the config file. This seems to be an undesired behavior in H2 with security implications.

Considering documentation doesn't suggest any other way than using the file config, a fix could be removing the `-webAdminPassword` as a valid CLI argument. Another thing to improve the security is to read the CLI arguments and after that set the values provided by the `loadProperties ()` function.

Timeline

- Aug 3rd, 2022 - First report
- Aug 7th, 2022 - Follow up
- Aug 14th, 2022 - Follow up
- Aug 24th, 2022 - Follow up
- Sep 4th, 2022 - Maintainer answer
- Sep 20th, 2022 - Fix suggestion
- Oct 12th, 2022 - Follow up
- Nov 2nd, 2022 - 90 days from the first report
- Nov 23th, 2022 - CVE-2022-45868 was assigned