QRadar Community Edition 7.3.1.6 Arbitrary Object Instantiation

Authored by Yorick Koster, Securify B.V.

Posted Apr 21, 2020

QRadar Community Edition version 7.3.1.6 is vulnerable to instantiation of arbitrary objects based on user-supplied input. An authenticated attacker can abuse this to perform various types of attacks including server-side request forgery and (potentially) arbitrary execution of code.

tags | exploit, arbitrary, file inclusion
advisories | CVE-2020-4272
SHA-256 | 79acda4a95f3ff77796484c45f9a5e4263e1e7678990f7cefeb06fe52b21e965


```
-----------------------------------------------------------------------------
Arbitrary class instantiation & local file inclusion vulnerability in
QRadar Forensics web application
-----------------------------------------------------------------------------
Yorick Koster, September 2019

-----------------------------------------------------------------------------
Abstract
-----------------------------------------------------------------------------
It was found that the QRadar Forensics web application is vulnerable to
instantiation of arbitrary objects based on user-supplied input. An
authenticated attacker can abuse this to perform various types of
attacks including Server-Side Request Forgery and (potentially)
arbitrary execution of code.

In addition, the same input is also used to include PHP files, which can
be used to include arbitrary local files. By abusing the case upload
functionality, it is possible for an authenticated user to upload a PHP
file to a known location on the system. By exploiting the local file
inclusion vulnerability it is possible to run arbitrary PHP code. This
code will be executed with the privileges of the Apache system user
(generally the nobody user).

-----------------------------------------------------------------------------
See also
-----------------------------------------------------------------------------
CVE-2020-4272 [2]
6189645 [3] - IBM QRadar SIEM is vulnerable to instantiation of
arbitrary objects (CVE-2020-4272)

-----------------------------------------------------------------------------
Tested versions
-----------------------------------------------------------------------------
This issue was successfully verified on QRadar Community Edition [4]
version 7.3.1.6 (7.3.1 Build 20180723171558).

-----------------------------------------------------------------------------
Fix
-----------------------------------------------------------------------------
IBM has released the following versions of QRader in which this issue
has been resolved:

- QRadar / QRM / QVM / QNI 7.4.0 GA [5] (SFS)
- QRadar / QRM / QVM / QRIF / QNI 7.3.3 Patch 3 [6] (SFS)
- QRadar / QRM / QVM / QRIF / QNI 7.3.2 Patch 7 [7] (SFS)
- QRadar Incident Forensics 7.4.0 [8] (ISO)
- QRadar Incident Forensics 7.4.0 [9] (SFS)

-----------------------------------------------------------------------------
Introduction
-----------------------------------------------------------------------------
QRadar [10] is IBM's enterprise SIEM [11] solution. A free version of
QRadar is available that is known as QRadar Community Edition [4]. This
version is limited to 50 events per second and 5,000 network flows a
minute, supports apps, but is based on a smaller footprint for
non-enterprise use.

The QRadar web application contains functionality to render various
graphs. The graph that needs to be rendered is based on user-supplied
request parameters. The correct graph and dataset classes are
dynamically loaded based on these parameters. No validation is performed
on the user-supplied parameters, allowing authenticated users to
instantiate arbitrary classes, which can be exploited to perform various
attacks including Server-Side Request Forgery and (potentially)
arbitrary execution of code via specially crafted Phar files [12].

In case a dataset class is provided that has not been declared (loaded)
yet. The code tries to include the correct PHP file in which the class
is defined. The file name of the include file is also based on the same
request parameter. Consequently, the web application is vulnerable to
local file inclusion.

If an attacker manages to place an arbitrary PHP file on the local
system, it is possible to abuse this issue to run arbitrary PHP code. It
was found that the case upload functionality allows uploading of PHP
files to a known location, thus allowing for the execution of arbitrary
PHP code. This code will be executed with the privileges of the Apache
system user (generally the nobody user).


-----------------------------------------------------------------------------
Details
-----------------------------------------------------------------------------
These issues are present in the graphs.php file. This PHP file accepts a
number of request parameters, including chart, dataset, and
output_image.

/opt/ibm/forensics/html/graphs.php:
$chart        = ( isset($_REQUEST['chart']) ?
htmlspecialchars($_REQUEST['chart']) : null );
$dataClass    = ( isset($_REQUEST['dataset']) ?
htmlspecialchars($_REQUEST['dataset']) : null );
$output_image = ( isset($_REQUEST['output_image']) ?
$_REQUEST['output_image'] : null );

If the output_image parameter is set to true, the PHP code will directly
try to instantiate an object with the name provided in the chart
parameter. One argument is passed to the constructor for which its value
is obtain from a request parameter with the same name as the selected
class name. If the class is successfully loaded, the drawChart() method
is called - regardless of whether this method actually exists.

/opt/ibm/forensics/html/graphs.php:
// Present the data
$cparams = $_REQUEST[$chart];
$cs = new $chart($cparams);
if($cs)
  $cs->drawChart();

No validation is performed on the user-supplied input, allowing for
authenticated attackers to instantiate practically any object in scope
of the page. In addition, the first argument that is passed to the
constructor is also controlled by the attacker.

What an attacker might do depends on the class that is instantiated and
the code that is executed by the constructor. A possible attack scenario
would be to perform a Server-Side Request Forgery attack by
instantiating a class that calls a method supporting one of the built-in
PHP wrappers [13].
```

Several classes exists in the Forensics code base, like the
DistribConfigHelper class. There are also built-in PHP classes that are
in scope and also allow for Server-Side Request Forgery, like the
SplFileObject [14] class. For example:

https://<ip>/forensics/graphs.php?
chart=DistribConfigHelper&DistribConfigHelper=https://127.0.0.1/&output_image=1
https://<ip>/forensics/graphs.php?chart=SplFileObject&SplFileObject=https://127.0.0.1/&output_image=1
https://<ip>/forensics/graphs.php?
chart=SplFileObject&SplFileObject=php://filter/read=string.toupper/resource=https://127.0.0.1/&output_image=1

Using the same PHP wrappers it is also possible to load arbitrary Phar
[15] files from the local machine. A known attack [12] (by Sam Thomas
[16]) exists where an attacker can trigger PHP objects to be
deserialized when a Phar file is loaded. Although code execution through
deserialization is possible in the Forensics application, exploiting
this issue is not that trivial. In particular, the attack can only be
executed from an object with a __wakeup() or __destruct() PHP magic
method [17]. The classes in scope of the vulnerable page don't appear to
have suitable magic methods that could be used to execute an exploit
(POP) chain.

Besides finding a suitable magic method, exploiting the Phar wrapper
also requires that the attacker can place a Phar file on the target
systems as Phar files can't be loaded from remote locations. It was
found that the case upload functionality allows uploading of files to a
known location. However, since the graph page also contains a local file
inclusion vulnerability, it makes more sense to target that
vulnerability instead.

The vulnerable code is executed in case the output_image request
parameter isn't present or is set to false. In this case the requested
class name is provided in the dataset request parameter. If this class
isn't (yet) in scope of the PHP page, an attempt is made to load it.
This is done by iterating though a list of predefined folder names, if a
file exists with the same name of the requested class, it will be
included after another which check is done to see if the class is in
scope.

```
/opt/ibm/forensics/html/graphs.php:
$haveDataClass = class_exists($dataClass);
if(!$haveDataClass) {
  foreach(array('', $DEJAVU_URL. 'Reports/','reports/') as $path) {
    $module = $path . $dataClass . ".php";
    if(file_exists($module)) {
    try {
      require_once($module);
      $haveDataClass = class_exists($dataClass);
      if($haveDataClass)
        break;
    } catch (Exception $e) {
      // Do nothing
      $msg = $e->getMessage();
    }
  }
}
```

As no validation is done on the class name, it is possible to include
files outside of these folder using path traversal. However this isn't
really needed as the first folder that is searched is empty, thus
allowing for absolute path names. In addition, it is also possible to
provide URL type paths. The call to file_exists() will block most PHP
wrappers. Some built-in wrappers will pass through the file_exists()
call, including the ftp:// [18] and ssh2.sftp:// [19] wrappers. In
theory, it should be able to include a file over (S)FTP were it not that
including files from remote locations has been disabled in the PHP
configuration.

```
/etc/php.ini:
; http://php.net/allow-url-include
allow_url_include = Off
```

Because it is possible to upload arbitrary files via the case upload
functionality, it is not that difficult to run arbitrary PHP code
regardless of these restrictions. Although other methods also exists, we
can just upload a PHP file to a known location and abuse this local file
inclusion vulnerability to execute the uploaded file.

References

[1] https://www.securify.nl/advisory/SFY20200407/arbitrary-class-instantiation-_-local-file-inclusion-
vulnerability-in-qradar-forensics-web-application.html
[2] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-4272
[3] https://www.ibm.com/support/pages/node/6189645
[4] https://developer.ibm.com/qradar/ce/
[5] https://www.ibm.com/support/fixcentral/swg/downloadFixes?
parent=IBM%20Security&product=ibm/Other+software/IBM+Security+QRadar+SIEM&release=7.4.0&platform=Linux&function=
QRADAR-QRSIEM-20200304205308&includeRequisites=1&includeSupersedes=0&downloadMethod=http
[6] https://www.ibm.com/support/fixcentral/swg/downloadFixes?
parent=IBM%20Security&product=ibm/Other+software/IBM+Security+QRadar+SIEM&release=7.3.0&platform=Linux&function=
QRADAR-QRSIEM-20200409085709&includeRequisites=1&includeSupersedes=0&downloadMethod=http
[7] https://www.ibm.com/support/fixcentral/swg/downloadFixes?
parent=IBM%20Security&product=ibm/Other+software/IBM+Security+QRadar+SIEM&release=7.3.0&platform=Linux&function=
QRADAR-QRSIEM-20200406171249&includeRequisites=1&includeSupersedes=0&downloadMethod=http
[8] https://www.ibm.com/support/fixcentral/swg/downloadFixes?
parent=IBM%20Security&product=ibm/Other+software/IBM+Security+QRadar+Incident+Forensics&release=7.4.0&platform=1
QRADAR-QIFFULL-2019.18.0.20200304205308&includeRequisites=1&includeSupersedes=0&downloadMethod=http
[9] https://www.ibm.com/support/fixcentral/swg/downloadFixes?
parent=IBM%20Security&product=ibm/Other+software/IBM+Security+QRadar+Incident+Forensics&release=7.4.0&platform=1
QRADAR-QIFSFS-2019.18.0.20200304205308&includeRequisites=1&includeSupersedes=0&downloadMethod=http
[10] https://www.ibm.com/security/security-intelligence/qradar
[11] https://en.wikipedia.org/wiki/Security_information_and_event_management
[12] https://github.com/s-n-t/presentations/blob/master/us-18-Thomas-It's-A-PHP-Unserialization-Vulnerability-
Jim-But-Not-As-We-Know-It.pdf
[13] https://www.php.net/manual/en/wrappers.php
[14] https://www.php.net/manual/en/splfileobject.construct.php
[15] https://www.php.net/manual/en/book.phar.php
[16] https://twitter.com/_s_n_t
[17] https://www.php.net/manual/en/language.oop5.magic.php
[18] https://www.php.net/manual/en/wrappers.ftp.php
[19] https://www.php.net/manual/en/wrappers.ssh2.php

Spoof (2,166)
SQL Injection (16,102)
TCP (2,379)
Trojan (686)
UDP (876)
Virus (662)
Vulnerability (31,136)
Web (9,365)
Whitepaper (3,729)
x86 (946)
XSS (17,494)
Other
SUSE (1,444)
Ubuntu (8,199)
UNIX (9,159)
UnixWare (185)
Windows (6,511)
Other