sales@candelatech.com
support@candelatech.com
+1 (360) 380-1618 [PST, GMT -8]

Create Layer 4-7 HTTP Upload Traffic

Goal: using curl, it is possible to not only download content, but UPLOAD content as well. There must be a web source that accepts uploaded data. Below we will cover how to construct an PHP and NGINX web page that will allow a LANforge system to accept web uploads.

Architecture of Uploads

By default, web servers (like Apache or Nginx) are designed to respond by giving content. Receiving content requires some add-on to accept incoming uploads. The notable methods are CGI and WebDAV. WebDAV configurations are specific to either Apache or Nginx, so we will focus this cookbook on the CGI method, with PHP as the CGI engine. PHP is installed by default with LANforge systems.

Deciding on the Upload Destination

Continual traffic generation is a scenario that would fill the space of any web server. This cookbook describes a technique that will not save uploaded data, and thus should allow an indefinite duration of upload traffic. Specifically, processes that accept uploaded data **do not need to save it**. The technique used by up.php relies on the files being uploaded to `/tmp` which on most recent LANforge systems is a `tmpfs` RAM-based filesystem.

> i Using `tmpfs` for very large uploads, (or hundreds of small uploads) can consume LANforge system memory quickly.

Upload Considerations

1. There is no good way to ignore the uploaded output. It goes immediately to the filesystem.
2. The default fedora upload location is /tmp. This is a RAM-based filesystem. You will crash your LANforge *nginx* server, and possibly the machine itself if your uploads take up more than half your ram.
3. If you want to upload a lot of data per upload, it is going to have to live on the LANforge's SSD.
    1. In `/etc/php.ini`, set `upload_tmp_dir = "/home/lanforge/tmp"`
    2. Consider that SSDs have a writes-per-lifetime limit. Continual writing adds wear to the drive.
    3. SSDs can only write between 30-500MB/s (4000 Mbps). This depends on the model of SSD, the speed of the SATA bus, and the write-caching and wear leveling techniques the SSD uses.
    4. If doing HTTP uploads is a performance test, we suggest building a system with hundreds of gigabytes of RAM for `tmpfs` space.
4. To handle large numbers of stations doing uploads, set the upload size limit small to avoid filling up RAM.
    1. Monitor free RAM with **htop**. On a ct521a, there is 4GB of RAM, total. Thus you have up to 2GB of free space in */tmp*:
    2. Remeber that as while the LANforge GUI runs, it consumes RAM itself. Depending on the duration of the report, it can consume most of the remaining ram. *Consider running the LANforge GUI on a separate machine*.
    3. In `/etc/php.ini`, set your *post_max_size* and *upload_max_filesize* parameters to (2GB / number of Layer 4 connections):
        1. For 16 stations with one Layer 4 connection each:
           `2 x 1024 x 1024 x 1024 / 16 = 134,217,728 ~=134MB`
        2. For 32 stations with two Layer 2 connections each:
           `2 x 1024 x 1024 x 1024 / 64 = 33,554,432 ~=33MB`

3. The *php.ini* default max upload size is set at 2MB:

```
post_max_size = 8M
upload_max_filesize = 2M
```

4. These system defaults **permit 1024 simultaneous uploads**.

Configure nginx to use *up.php*

1. Copy *up.php* to `/usr/local/lanforge/nginx/html`
2. In the Port Mgr tab, modify your port and select the ☑**HTTP** checkbox and click **Apply**. In this example, the port is **eth1** (192.168.48.92). Keep the window open.
3. In a terminal, edit `/home/lanforge/vr_conf/nginx_eth1.conf`
   1. Remove the first line
   2. At the bottom of the *server {}* stanza, add this configuration:

```
location /up.php {
     include /usr/local/lanforge/nginx/conf/fastcgi_params;
     fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
     fastcgi_pass unix:/run/php-fpm/www.sock;
     fastcgi_index index.php;
}
```

Example of file:

```
# Remove the first line '# Autogenerated by ...' and edit the file as
# desired for a custom config file.

worker_processes        1;
error_log logs/eth1_error.log;
pid             /home/lanforge/vr_conf/nginx_eth1.pid;
events {
    worker_connections  1024;
}

http {
    include       /usr/local/lanforge/nginx/conf/mime.types;
    default_type  application/octet-stream;
    access_log  logs/eth1_access.log;
    sendfile            on;
    keepalive_timeout  65;

    server {
        listen       192.168.48.92:80 bind_dev=eth1;
        server_name  localhost;
        access_log  logs/eth1_host.access.log;

        location / {
            root   html;
            index  index.html index.htm;
        }
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }
        location /up.php {
             include /usr/local/lanforge/nginx/conf/fastcgi_params;
             fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
             fastcgi_pass unix:/run/php-fpm/www.sock;
             fastcgi_index index.php;
        }
    }
}
```

1. Save the file.
   1. Edit the `/etc/php-fpm.d/www.conf` file
      1. Look for the *listen.acl_users* parameter, and add the user `nobody`
      2. `listen.acl_users = apache,nginx,nobody`
      3. save
   2. Restart the php-fpm service:

```
# systemctl restart php-fpm.service
```

Monitor nginx errors

1. To monitor the nginx log file:

1. Open a terminal
2. Become root using `sudo -s`
3. ```
   # cd /usr/local/lanforge/nginx/logs
   ```
4. ```
   # tail -F *.log
   ```

2. Permission Denied errors:
   1. For this error: `[crit] connect() to unix:/run/php-fpm/www.sock failed (13: Permission denied) while connecting to upstream, client: 192.168.48.18, server: localhost, request: "GET /up.php HTTP/1.1", upstream: "fastcgi://unix:/run/php-fpm/www.sock:"`
   2. Edit the `/etc/php-fpm/www.conf` file mentioned above and restart the php-fpm service.

Apply changed nginx settings

1. In the **eth1 Configure Settings** window, you will toggle the HTTP setting to re-apply the changed configuration:
   1. Unselect ☐**HTTP**
   2. Click **Apply**
   3. Select ☑**HTTP**
   4. Click **Apply**

Checking for the *up.php* page with Curl

1. Open a terminal and become root:
   1. ```
      $ sudo -s
      ```
   2. ```
      # cd /home/lanforge
      ```
   3. ```
      # ./vrf_exec.bash wlan0 curl -sqv http ://192.168.48.92/up.php
      ```
   4. Look for errors, like *Sorry, the page you are looking for is currently unavailable.*

Browsing the *up.php* page with Firefox

1. To see this page from the LANforge desktop, you will use the `vrf_exec.bash` script.
2. Open a terminal and issue these commands:
   1. ```
      # xhost +
      ```
   2. ```
      # su -
      ```
   3. ```
      # cd /home/lanforge
      ```
   4. ```
      # export DISPLAY=:1
      ```
   5. ```
      # ./vrf_exec.bash eth1 firefox http ://192.168.48.92/up.php
      ```

Creating a Generic Endpoint to upload with Curl

1. There a token files to upload, located in `/var/www/html` named *dataslug{size}.bin*.
2. The `curl -F` command will issue a POST in the multipart-form manner.
3. The `filename=` parameter is the name of the form's `[input]` parameter that matches the `[input`

`type=file]` element in the page.

4. The `@/var/www/html/data_slug_256K.bin` is curl syntax for opening the named file.
5. Start off by practicing the *vrf_exec.bash* command from a terminal:
    1. `./vrf_exec.bash wlan0 curl -F 'filename=@/var/www/html/data_slug_256K.bin' http://192.168.48.92/up.php`
6. Once that works, go to the LANforge GUI *Generic tab* and create a new endpoint:
    1. Enter a name
    2. Select your resource and port (wlan0)
    3. Select *Command Builder: Generic*
    4. Enter the command:

```
./vrf_exec.bash wlan0 curl -F 'filename=@/var/www/html/data_slug_256K.bin' http://192.168.48.92/up.php
```

    5. Click **OK**
7. In the *Generic tab*, select your connection and click **Start**
    1. Double click the connection
    2. In the *Create/Modify Generic Endpoint* window, click the **Sync** button (at the bottom).
    3. You will see the HTML text of the result.

```
Dload  Upload   Total   Spent    Left  Speed
0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100  256k    0   622  100  256k  24960  10.0M --:--:-- --:--:-- --:--:-- 10.4M
<!DOCTYPE html>
<html lang="en">
<head>
 <title>Layer 4 Upload Test</title>
 <meta charset="utf-8">
 <style>
     body {
         font-family: "Century Gothic", "DejaVu Sans",Arial,Helvetica,sans-serif;
         margin: 1em auto;
     }
     h1 {
         text-align: center;
     }
 </style>
</head>
<body>
 <h1>Layer 4 File Upload Test</h1>
     <h1>Result:</h1>
     <ul>
         <li>File name: <tt><data_slug_256K.bin></tt></li>
         <li>File size: <tt><262144></tt></li>
         <li>File tmp: <tt></tmp/phpRLHVa2></tt></li>
     </ul>
</body></html>
```

    4. A command that might provide better output could look like:

```
./vrf_exec.bash wlan0 curl -s -o/tmp/$$.html -F 'filename=@/var/www/html/data_slug_256K.bin' http://192.168.48.92/up.php; grep File /tmp/$$.html
```

    5. It provides terse output like:

```
<h1>Layer 4 File Upload Test</h1>
     <li>File name: <tt>[data_slug_256K.bin]</tt></li>
     <li>File size: <tt>[262144]</tt></li>
     <li>File tmp: <tt>[/tmp/phpfhDFMf]</tt></li>
```

The *up.php* Page

You can copy the up.php page as text from here: