
nodogsplash Documentation

Release 3.2.1

the nodogsplash contributors

Sep 28, 2018

1	Overview	3
2	Installing Nodogsplash	5
2.1	OpenWrt	5
2.2	Debian	6
3	How to compile Nodogsplash	7
3.1	Linux/Unix	7
3.2	OpenWrt	7
4	Frequently Asked Questions	9
4.1	What's the difference between v0.9, v1, v2 and v3?	9
4.2	Can I update from v0.9 to v1	9
4.3	Can I update from v0.9/v1 to v2.0.0	9
4.4	Can I update from v0.9/v1/v2 to v3.0.0	10
4.5	I would like to use QoS or TrafficControl on OpenWrt	10
4.6	Is https capture supported?	10
5	How Nodogsplash (NDS) works	11
5.1	Packet filtering	12
5.2	Traffic control	12
6	Forwarding Authentication Service (FAS)	13
6.1	Overview	13
6.2	Using FAS	13
6.3	Running FAS on your Nodogsplash router	14
6.4	Using the simple example files	15
7	BinAuth Option	17
8	Using ndsctl	19
9	Customising nodogsplash	21
9.1	The Configuration File	21
9.2	The Splash Page	22
10	Debugging Nodogsplash	25

11	TODO List	27
12	Indices and tables	29

Nodogsplash is a high performance, small footprint Captive Portal, offering by default a simple splash page restricted Internet connection, yet incorporates an API that allows the creation of sophisticated authentication applications.

It was derived originally from the codebase of the Wifi Guard Dog project.

Nodogsplash is released under the GNU General Public License.

- Mailing List: <http://ml.ninux.org/mailman/listinfo/nodogsplash>
- Original Homepage *down*: <http://kokoro.ucsd.edu/nodogsplash>
- Wifidog: <http://dev.wifidog.org/>
- GNU GPL: <http://www.gnu.org/copyleft/gpl.html>

The following describes what Nodogsplash does, how to get it and run it, and how to customize its behavior for your application.

Contents:

Nodogspash (NDS) is a high performance, small footprint Captive Portal, offering by default a simple splash page restricted Internet connection, yet incorporates an API that allows the creation of sophisticated authentication applications.

If you want to provide simple and immediate public access to an Internet connection with users giving some acknowledgment of the service, Nodogsplash does this by default. Customising the page seen by users is a simple matter of editing the simple default html splash page file.

If you want to enforce use of a set of preset usernames and passwords with perhaps a limited connection time, the addition of a simple shell script is all that is required.

If you want a more sophisticated authentication system providing a dynamic web interface you can do that too by providing your own web service written in a language such as php running on its own server.

Taking this to the extreme, if you want to link Nodogsplash to your own centralised Internet based authentication service with user account self generation and access charging, you can do that too, or anything in between.

All modern mobile devices, most desktop operating systems and most browsers now have a Captive Portal Detection process that automatically issues a port 80 request on connection to a network. Nodogsplash detects this and serves a 'splash' web page.

The splash page in its most basic form, contains a *Continue* button. When the user clicks on it, access to the Internet is granted subject to a preset time interval.

Nodogsplash does not currently support traffic control but is fully compatible with other stand alone systems such as Smart Queue Management (SQM).

Nodogsplash supports multiple means of authentication:

- Click the *Continue* button (default)
- Call an external script that may accept username/password and set session durations per user.
- Forwarding authentication to an external service

Installing Nodogsplash

2.1 OpenWrt

- Have a router working with OpenWrt. At the time of writing, Nodogsplash has been tested with OpenWrt 17.01.4/5 and 18.06.0

It may or may not work on older versions of OpenWrt or on other kinds of Linux-based router firmware.

- Make sure your router is basically working before you try to install Nodogsplash. In particular, make sure your DHCP daemon is serving addresses on the interface that nodogsplash will manage.

The default is br-lan but can be changed to any interface by editing the `/etc/config/nodogsplash` file.

- To install Nodogsplash, you may use the OpenWrt Luci web interface or alternatively, ssh to your router and run the command:

```
opkg update
```

followed by

```
opkg install nodogsplash
```

- Nodogsplash is enabled by default and will start automatically on reboot or can be started and stopped manually.
- If the interface that you want Nodogsplash to manage is not br-lan, edit `/etc/config/nodogsplash` and set `GatewayInterface`.
- To start Nodogsplash, run the following, or just reboot the router:

```
/etc/init.d/nodogsplash start
```

- To test the installation, connect a client device to the interface on your router that is managed by Nodogsplash (for example, connect to the router's wireless lan).

Most client device operating systems and browsers support Captive Portal Detection (CPD) and the operating system or browser on that device will attempt to contact a pre defined port 80 web page.

CPD will trigger Nodogsplash to serve the default splash page where you can click or tap Continue to access the Internet.

See the Authentication section for details of setting up a proper authentication process.

If your client device does not display the splash page it most likely does not support CPD.

You should then manually trigger Nodogsplash by trying to access a port 80 web site (for example, google.com:80 is a good choice).

- To stop Nodogsplash:

```
/etc/init.d/nodogsplash stop
```

- To uninstall Nodogsplash:

```
opkg remove nodogsplash
```

2.2 Debian

There isn't a package in the repository (yet). But we have support for a debian package.

Requirements beside debian tools are:

- libmicrohttpd-dev (>= 0.9.51) [available in **stretch**]

But you can also compile libmicrohttpd on your own if you're still running jessie or older.

```
sudo apt-get install debhelper dpkg-dev dh-systemd libmicrohttpd-dev
```

```
apt-get install build-essential debhelper devscripts hardening-includes
```

Run this command in the repository root folder to create the package:

```
dpkg-buildpackage
```

The package will be created in the parent directory.

Use this command if you want to create an unsigned package:

```
dpkg-buildpackage -b -rfakeroot -us -uc
```

You will find the .deb packages in parent directory.

How to compile Nodogsplash

3.1 Linux/Unix

Install libmicrohttpd including the header files (often call -dev package).

```
git clone https://github.com/nodogsplash/nodogsplash.git
cd nodogsplash
make
```

If you installed the libmicrohttpd to another location (e.g. /tmp/libmicrohttpd_install/) replace path in the make call with

```
make CFLAGS="-I/tmp/libmicrohttpd_install/include" LDFLAGS="-L/tmp/libmicrohttpd_
↪install/lib"
```

After compiling you can call `make install` to install nodogsplash to /usr/

3.2 OpenWrt

To compile nodogsplash please use the package definition from the feeds package.

```
git clone git://git.openwrt.org/trunk/openwrt.git
cd openwrt
./scripts/feeds update
./scripts/feeds install
./scripts/feeds install nodogsplash
```

Select the appropriate “Target System” and “Target Profile” in the menuconfig menu and build the image.

```
make defconfig
make menuconfig
make
```

Frequently Asked Questions

4.1 What's the difference between v0.9, v1, v2 and v3?

v0.9 and v1 are the same codebase with the same feature set. If the documentation says something about v1, this is usually also valid for v0.9.

v2 was developed before version v1 was released. In v2 the http code was replaced by libmicrohttpd and the template engine was rewritten. Many features became defunct because of this procedure.

v3 cleans up the source code and adds two major new features,

- FAS enabling an external forwarding authentication service to be called,

and

- binauth, enabling an external script to be called for simple username/password authentication as well as doing post authentication processing such as setting session durations. This is similar to the old binvoucher feature, but more flexible.

In addition, in v3, the ClientTimeout setting was split into PreauthIdleTimeout and AuthIdleTimeout and for the ClientForceTimeout setting, SessionTimeout is now used instead.

4.2 Can I update from v0.9 to v1

Updating to v1.0.0 and v1.0.1, this is a very smooth update with full compatibility.

Updating to 1.0.2 requires iptables v1.4.21 or above.

4.3 Can I update from v0.9/v1 to v2.0.0

You can, if:

- You don't use BinVoucher

- You have iptables v1.4.21 or above

4.4 Can I update from v0.9/v1/v2 to v3.0.0

You can, if:

- You don't use BinVoucher
- You have iptables v1.4.21 or above
- You use the new options contained in the version 3 configuration file

4.5 I would like to use QoS or TrafficControl on OpenWrt

The original pre version 1 feature has been broken since OpenWrt 12.09 (Attitude Adjustment), because the IMQ (Intermediate queueing device) is no longer supported.

Pull Requests are welcome!

However the OpenWrt package, SQM Scripts (Smart Queue Management), is fully compatible with Nodogsplash and if configured to operate on the Nodogsplash interface (br-lan by default) will provide efficient IP connection based traffic control to ensure fair usage of available bandwidth.

4.6 Is https capture supported?

No. Because all connections would have a critical certificate failure.

HTTPS web sites are now more or less a standard and to maintain security and user confidence it is essential that captive portals **DO NOT** attempt to capture port 443.

Captive Portal Detection (CPD) has evolved as an enhancement to the network manager component included with major Operating Systems (Linux, Android, iOS/macOS, Windows). Using a pre-defined port 80 web page (depending on the vendor) the network manager will detect the presence of a captive portal hotspot and notify the user. In addition, most major browsers now support CPD.

How Nodogsplash (NDS) works

A wireless router, typically running OpenWrt or some other Linux distribution, has two or more interfaces; NDS manages one of them. This will typically be br-lan, the bridge to both the wireless and wired LAN; or could be for example wlan0 if you wanted NDS to work just on the wireless interface.

A simplified summary of operation is as follows:

By default, NDS blocks everything, but intercepts port 80 requests.

An initial port 80 request will be generated on a client device, either by the user manually browsing to an http web page, or automatically by the client device's built in Captive Portal Detection (CPD).

As soon as this initial port 80 request is received, NDS will redirect the client to either its own splash page, or a splash page on a configured Forwarding Authentication Service (FAS).

The user of the client device will then be expected to complete some actions on the splash page, such as accepting terms of service, entering a username and password etc. (this will of course be on either the basic NDS splash.html or the page presented by the FAS, depending on the NDS configuration).

Once the user on the client device has successfully completed the splash page actions, the page then links directly, with a query string, to an NDS virtual http directory provided by NDS's built in web server.

For security, NDS expects to receive the same valid token it allocated when the client issued its initial port 80 request. If the token received is valid, NDS then "authenticates" the client device, allowing access to the Internet.

However if Binauth is enabled, NDS first calls the Binauth script, passing if required a username and password to that script.

If the binauth script returns positively (ie return code 0), NDS then "authenticates" the client device, allowing access to the Internet.

In FAS secure mode, it is the responsibility of the FAS to obtain the client token in a secure manner from NDS.

When FAS is disabled, the token is supplied to the basic splash.html page served by NDS and passed back in clear text in the query string along with any username and password required for Binauth.

Note: FAS and Binauth can be enabled together. This can give great flexibility with FAS providing authentication and Binauth providing post authentication processing closely linked to NDS.

5.1 Packet filtering

Nodogsplash considers four kinds of packets coming into the router over the managed interface. Each packet is one of these kinds:

1. **Blocked**, if the MAC mechanism is block, and the source MAC address of the packet matches one listed in the BlockedMACList; or if the MAC mechanism is allow, and source MAC address of the packet does not match one listed in the AllowedMACList or the TrustedMACList. These packets are dropped.
2. **Trusted**, if the source MAC address of the packet matches one listed in the TrustedMACList. By default, these packets are accepted and routed to all destination addresses and ports. If desired, this behavior can be customized by FirewallRuleSet trusted-users and FirewallRuleSet trusted-users-to-router lists in the nodogsplash.conf configuration file, or by the EmptyRuleSetPolicy trusted-users EmptyRuleSetPolicy trusted-users-to-router directives.
3. **Authenticated**, if the packet's IP and MAC source addresses have gone through the nodogsplash authentication process and has not yet expired. These packets are accepted and routed to a limited set of addresses and ports (see FirewallRuleSet authenticated-users and FirewallRuleSet users-to-router in the nodogsplash.conf configuration file).
4. **Preauthenticated**. Any other packet. These packets are accepted and routed to a limited set of addresses and ports (see FirewallRuleSet preauthenticated-users and FirewallRuleSet users-to-router in the nodogsplash.conf configuration file). Any other packet is dropped, except that a packet for destination port 80 at any address is redirected to port 2050 on the router, where nodogsplash's built in libhttpd-based web server is listening. This begins the 'authentication' process. The server will serve a splash page back to the source IP address of the packet. The user clicking the appropriate link on the splash page will complete the process, causing future packets from this IP/MAC address to be marked as Authenticated until the inactive or forced timeout is reached, and its packets revert to being Preauthenticated.

Nodogsplash implements these actions by inserting rules in the router's iptables mangle PREROUTING chain to mark packets, and by inserting rules in the nat PREROUTING, filter INPUT and filter FORWARD chains which match on those marks.

Because it inserts its rules at the beginning of existing chains, nodogsplash should be insensitive to most typical existing firewall configurations.

5.2 Traffic control

Data rate control on an IP connection basis can be achieved using Smart Queue Management (SQM) configured separately, with NDS being fully compatible.

It should be noted that while setup options and binauth do accept traffic/quota settings, these values currently have no effect and are reserved for future development.

Forwarding Authentication Service (FAS)

6.1 Overview

Nodogsplash (NDS) supports external (to NDS) authentication service via simple configuration options.

These options are:

1. **fasport**. This enables Forwarding Authentication Service (FAS). Redirection is changed from splash.html to a FAS. The value is the IP port number of the FAS.
2. **fasremoteip**. If set, this is the remote ip address of the FAS, if not set it will take the value of the NDS gateway address.
3. **faspath**. This is the path to the login page on the FAS.
4. **fas_secure_enable**. If set to “1”, authentication and the client token are not revealed and it is the responsibility of the FAS to request the token from NDSCTL. If set to “0”, the client token is sent to the FAS in clear text in the query string of the redirect along with authentication and redirect.

6.2 Using FAS

Note: All addresses (with the exception of fasremoteip) are relative to the *client* device, even if the FAS is located remotely.

When FAS is enabled, NDS automatically configures access to the FAS service.

The FAS service must serve an http splash of its own to replace the NDS splash.html. Typically, the FAS service will be written in PHP or any other language that can provide dynamic web content.

FAS can then provide an action form for the client, typically requesting login, or self account creation for login.

The FAS can be on the same device as NDS, on the same local area network as NDS, or on an Internet hosted web server.

Security.

If FAS Secure is enabled (`fas_secure_enabled = 1`, the default), NDS will supply only the gateway name, the client IP address and the originally requested URL in the query string in the redirect to FAS.

For example:

```
http://fasremoteip:fasport/faspath?gatewayname=[gatewayname]&clientip=[clientip]&redir=[requested-url]
```

It is the responsibility of FAS to obtain the unique client token allocated by NDS as well as constructing the return URL to NDS.

The return url will be constructed by FAS from predetermined knowledge of the configuration of NDS using gateway-name as an identifier.

The client's unique access token will be obtained from NDS by the FAS making a call to the `ndsctl` tool.

For example, the following command returns just the token with newline stripped:

```
ndsctl json [clientip] | grep token | tr -d "'\n'
```

If the client successfully authenticates in the FAS, FAS will return the unique token to NDS to finally allow the client access to the Internet.

A Secure Internet based FAS is best implemented as a two stage process, first using a local FAS, that in turn accesses an https remote FAS using tools such as `curl` or `wget`.

If FAS Secure is disabled (`fas_secure_enabled = 0`), NDS sends the token and other information to FAS as clear text.

For example:

```
http://fasremoteip:fasport/faspath?authaction=http://gatewayaddress:gatewayport/nodogsplash_auth/?clientip=[clientip]&gatewayname=[gatewayname]
```

Clearly in this case, a knowledgeable user could bypass FAS, so running `fas_secure_enabled = 1`, the default, is recommended.

Post FAS processing.

Once the client has been authenticated by the FAS, NDS must then be informed to allow the client to have access to the Internet.

This is done by accessing NDS at a special virtual URL. This is of the form:
`http://gatewayaddress:gatewayport/nodogsplash_auth/?tok=[token]&redir=[landing_page_url]`

This is most commonly done using an html form of method GET. The parameter `redir` can be the client's originally requested URL sent by NDS, or more usefully, the URL of a suitable landing page.

However, be aware that many client CPD processes will **automatically close** the landing page as soon as Internet access is detected.

Manual Access of NDS Virtual URL

If the user of an already authenticated client device manually accesses the NDS Virtual URL, they will be redirected back to FAS with the "status" query string.

This will be of the form:

```
http://fasremoteip:fasport/faspath?clientip=[clientip]&gatewayname=[gatewayname]&status=authenticated
```

FAS should then serve a suitable error page informing the client user that they are already logged in.

6.3 Running FAS on your Nodogsplash router

A FAS service will run quite well on `uhttpd` (the web server that serves `Luci`) on an OpenWrt supported device with 8MB flash and 32MB ram but shortage of ram may well be an issue if more than two or three clients log in at the same time.

For this reason a device with a minimum of 8MB flash and 64MB ram is recommended.

Running on uhttpd with PHP:

Install the modules `php7` and `php7-cgi` on OpenWrt for a simple example. Further modules may be required depending on your requirements.

To enable php in uhttpd you must add the line:

```
list interpreter ".php=/usr/bin/php-cgi"
```

to the `/etc/config/uhttpd` file in the config uhttpd 'main' or first section.

The two important NDS options to set will be:

1. `fasport`. By default this will be port 80 for uhttpd
2. `faspath`. Set to, for example, `/myfas/fas.php`, your FAS files being placed in `/www/myfas/`

Note 1:

A typical Internet hosted Apache/PHP shared server will be set up to serve multiple domain names.

To access yours, use:

`fasremoteip` = the ip address of the remote server

and, for example,

`faspath` = `/domainname/pathto/myfas/fas.php`

or

`faspath` = `/accountname/pathto/myfas/fas.php`

If necessary, contact your hosting service provider.

Note 2:

The configuration file `/etc/config/nodogsplash` contains the line "option enabled 1".

If you have done something wrong and locked yourself out, you can still SSH to your router and stop NoDogSplash (`ndsctl stop`) to fix the problem.

6.4 Using the simple example files

Assuming you want to run the FAS example demo locally under uhttpd on the same OpenWrt device that is running NDS, configured as above, do the following.

(Under other operating systems you may need to edit the `nodogsplash.conf` file in `/etc/nodogsplash` instead, but the process is very similar.)

First you should obtain the demo files by downloading the Nodogsplash zip file from

<https://github.com/nodogsplash/nodogsplash/>

Then extract the php files from the folder

```
"forward_authentication_service/nodog/"
```

OpenWrt and uhttpd:

- Create a folder
 - `/www/nodog/`
- Place the files `fas.php`, `landing.php`, `css.php`, `querycheck.php`, `tos.php`, `users.dat` in

`/www/nodog/`

- Edit

`/etc/config/nodogsplash`

adding the lines:

- `option fasport '80'`
- `option faspath '/nodog/fas.php'`
- `option fas_secure_enabled '0'`
- Restart NDS using the command “`service nodogsplash restart`”.

BinAuth Option

Key: BinAuth**Value: /path/to/executable/script**

Authenticate a client using an external program that get passed the (optional) username and password value. The exit code and output values of the program decide if and how a client is to be authenticated.

The program will also be called on client authentication and deauthentication.

For the following examples, *binauth* is set to */etc/nds_auth.sh* in *nodogsplash.conf*:

```
#!/bin/sh

METHOD="$1"
MAC="$2"

case "$METHOD" in
  auth_client)
    USERNAME="$3"
    PASSWORD="$4"
    if [ "$USERNAME" = "Bill" -a "$PASSWORD" = "tms" ]; then
      # Allow client to access the Internet for one hour (3600 seconds)
      # Further values are upload and download limits in bytes. 0 for no limit.
      echo 3600 0 0
      exit 0
    else
      # Deny client to access the Internet.
      exit 1
    fi
    ;;
  client_auth|client_deauth|idle_deauth|timeout_deauth|ndsctl_auth|ndsctl_
↪deauth|shutdown_deauth)
    INGOING_BYTES="$3"
    OUTGOING_BYTES="$4"
    SESSION_START="$5"
    SESSION_END="$6"
```

(continues on next page)

(continued from previous page)

```
# client_auth: Client authenticated via this script.
# client_deauth: Client deauthenticated by the client via splash page.
# idle_deauth: Client was deauthenticated because of inactivity.
# timeout_deauth: Client was deauthenticated because the session timed out.
# ndsctl_auth: Client was authenticated by the ndsctl tool.
# ndsctl_deauth: Client was deauthenticated by the ndsctl tool.
# shutdown_deauth: Client was deauthenticated by Nodogsplash terminating.
;;
esac
```

The *SESSION_START* and *SESSION_END* values are the number of seconds since 1970 or may be 0 for unknown/unlimited.

The splash.html page contains the following code:

```
<form method='GET' action='$authaction'>
<input type='hidden' name='tok' value='$tok'>
<input type='hidden' name='redir' value='$redir'>
username: <input type='text' name='username' value='' size='12' maxlength='12'>
<br>
password: <input type='password' name='password' value='' size='12' maxlength='10'>
<br>
<input type='submit' value='Enter'>
</form>
```

If a client enters a username ‘Bill’ and password ‘tms’, then the configured *binauth* script is executed:

```
/etc/nds_auth.sh auth_client 12:34:56:78:90 'Bill' 'tms'
```

For the authentication to be successful, the exit code of the script must be 0. The output can be up to three values. First the number of seconds the client is to be authenticated, second and third the maximum number of upload and download bytes limits. Values not given to NDS will resort to default values. Note that the traffic shaping feature that uses the upload/download values does not work right now.

After initial authentication by the script, Nodogsplash will immediately acknowledge by calling the *binauth* script again with:

```
/etc/nds_auth.sh client_auth 12:34:56:78:90 <incoming_bytes> <outgoing_bytes>
↔<session_start> <session_end>
```

Nodogsplash will also call the script when the client is authenticated and deauthenticated in general.

A nodogsplash install includes `ndsctl`, a separate application which provides some control over a running nodogsplash process by communicating with it over a unix socket. Some command line options:

- To print to stdout some information about your nodogsplash process:

```
/usr/bin/ndsctl status
```

- To print to stdout the list of clients in human readable format:

```
/usr/bin/ndsctl clients
```

- To print to stdout the list of clients in json format:

```
/usr/bin/ndsctl json
```

- To print to stdout the details of a particular client in json format (This is particularly useful if called from a FAS or Binauth script.):

```
/usr/bin/ndsctl json [mac|ip|token]
```

- To block a MAC address, when the MAC mechanism is block:

```
/usr/bin/ndsctl block MAC
```

- To unblock a MAC address, when the MAC mechanism is block:

```
/usr/bin/ndsctl unblock MAC
```

- To allow a MAC address, when the MAC mechanism is allow:

```
/usr/bin/ndsctl allow MAC
```

- To unallow a MAC address, when the MAC mechanism is allow:

```
/usr/bin/ndsctl unallow MAC
```

- To deauthenticate a currently authenticated user given their IP or MAC address:

```
/usr/bin/ndsctl deauth IP|MAC
```

- To set the verbosity of logged messages to n:

```
/usr/bin/ndsctl loglevel n
```

For more options, run `ndsctl -h`. (Note that if you want the effect of `ndsctl` commands to persist across `nodogsplash` restarts, you have to edit the configuration file.)

Customising nodogsplash

After initial installation, Nogogsplash (NDS) should be working in its most basic mode and client Captive Portal Detection (CPD) should pop up the default splash page.

Before attempting to customise NDS you should ensure it is working in this basic mode before you start.

NDS reads its configuration file when it starts up but the location of this file varies depending on the operating system.

As NDS is a package that requires hardware configured as an IP router, perhaps the most common installation is using OpenWrt. However NDS can be compiled to run on most Linux distributions, the most common being Debian or one of its popular variants (eg Raspbian).

If NDS is working in the default, post installation mode, then you will have met the NDS dependencies and can now move on to your own customisation.

9.1 The Configuration File

In OpenWrt, or operating systems supporting UCI (such as LEDE) the configuration is kept in the file:

```
/etc/config/nodogsplash
```

In other operating systems the configuration is kept in the file:

```
/etc/nodogsplash/nodogsplash.conf
```

Both of these files contain a full list of options and can be edited directly. A restart of NDS is required for any changes to take effect.

In the case of OpenWrt though, once you are confident in your configuration requirements you can use UCI to read and set any of the configuration options using simple commands, making this very convenient if making changes from scripts, such as those you may write to use with Binauth and FAS.

For example, to list the full configuration, at the command line type:

```
uci show nodogsplash
```

To display the Gateway Name, type:

```
uci get nodogsplash.@nodogsplash[0].gatewayname
```

To set the Gateway Name to a new value, type:

```
uci set nodogsplash.@nodogsplash[0].gatewayname='my new gateway'
```

To add a new firewall rule allowing access to another service running on port 8888 on the router, type:

```
uci add_list nodogsplash.@nodogsplash[0].users_to_router='allow
tcp port 8888'
```

Finally you must tell UCI to commit your changes to the configuration file:

```
uci commit nodogsplash
```

9.2 The Splash Page

The default simple splash page can be found at:

```
/etc/nodogsplash/htdocs/splash.html
```

When the splash page is served, the following variables in the page are replaced by their values:

- *\$gatewayname* The value of GatewayName as set in nodogsplash.conf.
- *\$authtarget* A URL which encodes a unique token and the URL of the user's original web request. If nodogsplash receives a request at this URL, it completes the authentication process for the client and replies to the request with a "302 Found" to the encoded originally requested URL.

It should be noted however that, depending on vendor, the client's built in CPD may not respond to simple html links.

An href link example that may prove to be problematical:

```
<a href="$authtarget">Enter</a>
```

(You should instead use a GET-method HTML form to send this information to the nodogsplash server; see below.)

- *\$imagesdir* The directory in nodogsplash's web hierarchy where images to be displayed in the splash page must be located.
- *\$tok*, *\$redir*, *\$authaction*, and *\$denyaction* are available and should be used to write the splash page to use a GET-method HTML form instead of using *\$authtarget* as the value of an href attribute to communicate with the nodogsplash server.

\$authaction and *\$denyaction* are virtual urls used to inform NDS that a client should be authenticated or deauthenticated and are of the form:

```
http://gatewayaddress:gatewayport/nodogsplash_auth/
```

and

```
http://gatewayaddress:gatewayport/nodogsplash_deny/
```

A simple example of a GET-method form:

```
<form method='GET' action='$authaction'>
  <input type='hidden' name='tok' value='$tok'>
  <input type='hidden' name='redir' value='$redir'>
  <input type='submit' value='Click Here to Enter'>
</form>
```

- *\$clientip*, *\$clientmac* and *\$gatewaymac* The respective addresses of the client or gateway. This might be useful in cases where the data needs to be forwarded to some other place by the splash page itself.
- *\$nclients* and *\$maxclients* User stats. Useful when you need to display something like “n of m users online” on the splash site.
- *\$uptime* The time Nodogsplash has been running.

A list of all available variables are included in the splash.html file.

If the user accesses the virtual url *\$authaction* when already authenticated, a status page is shown:

```
/etc/nodogsplash/htdocs/status.html
```

In the status.html file, the same variables as in the splash.html site can be used.

It should be noted when designing a custom splash page that for security reasons many client device CPD implementations:

- Immediately close the browser when the client has authenticated.
- Prohibit the use of href links.
- Prohibit downloading of external files (including .css and .js, even if they are allowed in NDS firewall settings).
- Prohibit the execution of javascript.

Also, note that any images you reference should reside in the subdirectory that is defined by *\$imagesdir* (default: “images”).

Debugging Nodogsplash

To see maximally verbose debugging output from nodogsplash, set log level to 7. This can be done in the UCI configuration file on OpenWrt adding the line:

```
option debuglevel '7'
```

or by editing the file

```
/etc/init.d/nodogsplash
```

and setting the `OPTIONS` variable to the flags “-s -d 7”.

Restart or reboot, and view messages with `logread`. Debug messages are logged to `syslog`.

The default level of logging is 5, `LOG_NOTICE`, and is more appropriate for routine use.

Logging level can also be set using `ndsctl`.

When stopped, nodogsplash deletes its iptables rules, attempting to leave the router’s firewall in its original state. If not (for example, if nodogsplash crashes instead of exiting cleanly) subsequently starting and stopping nodogsplash should remove its rules.

On OpenWrt, restarting the firewall will overwrite Nodogsplash’s iptables rules, so when the firewall is restarted it will automatically restart Nodogsplash if it is running.

Nodogsplash operates by marking packets. Many packages, such as `mwan3` and `SQM` scripts, also mark packets.

By default, Nodogsplash marks its packets in such a way that conflicts are unlikely to occur but the masks used by Nodogsplash can be changed if necessary in the configuration file.

Potential conflicts may be investigated by looking at your overall iptables setup. To list all the rules in all the chains, run

```
iptables -L
```

For extensive suggestions on debugging iptables, see for example, Oskar Andreasson’s tutorial at:

<https://www.frozentux.net/iptables-tutorial/iptables-tutorial.html>

CHAPTER 11

TODO List

Not all features are finished or working as properly or as efficiently as they should. Here is a list of things that need to be improved:

- While (un-) block/trust/allow via the `ndsctl` tool take effect, the state object of the client in NDS is not affected. Both systems still need to be connected (in `src/auth.c`).
- Include blocked and trusted clients in the client list - so that they can be managed.
- Extend Status processing to display a page when a user's authentication is rejected, e.g. because the user exceeded a quota or is blocked etc.
- Implement Traffic control on a user by user basis. This functionality was originally available but has been broken for many years.
- The code in `src/http_microhttpd.c` has evolved from previous versions and possibly has some missed edge cases. It would benefit from a rewrite to improve maintainability as well as performance.
- ip version 6 is not currently supported by NDS. It is not essential or advantageous to have in the short term but should be added at some time in the future.

CHAPTER 12

Indices and tables

- `genindex`
- `search`