nodogsplash Documentation

Release 2.0.0

the nodogsplash contributors

Aug 16, 2018

Contents

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 Frequenctly 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 I would like to use QoS or TrafficControl on OpenWrt . . . 9
- 4.5 Is https:// redirection supported? . . . 10

5 How nodogsplash works **11**
- 5.1 Packet filtering . . . 11
- 5.2 Traffic control . . . 12

6 Forwarding Authentication Service (FAS) **13**
- 6.1 Overview . . . 13
- 6.2 FAS Installation . . . 13

7 BinAuth Option **15**

8 Using ndsctl **17**

9 Customizing nodogsplash **19**

10 Debugging nodogsplash **21**

11 TODO List **23**

12 Indices and tables **25**

Nodogsplash offers a simple way to provide restricted access to an internet connection. It is derived 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:

CHAPTER 1

Overview

Nodogsplash (NDS) offers a solution to this problem: You want to provide controlled and reasonably secure public access to an internet connection; and while you want to require users to give some acknowledgment of the service you are providing, you don't need or want the complexity of user account names and passwords and maintaining a separate database-backed authentication server. When installed and running, Nodogsplash implements a simple 'authentication' protocol. First, it detects any user attempting to use your internet connection to request a web page. It captures the request, and instead serves back a 'splash' web page using its own builtin web server. The splash page contains a link which, when the user clicks on it, opens limited access for them to the internet via your connection, beginning by being redirected to their originally requested page. This access expires after a certain time interval. Nodogsplash also permits limiting the aggregate bandwidth provided to users, if you don't want to grant all of your available upload or download bandwidth. Specific features of Nodogsplash are configurable, by editing the configuration file and the splash page. The default installed configuration may be all you need, though.

Nodogsplash supports multiple means of authentication:

- hit the submit button (default)
- call an external script that may accept username/password
- forwarding authentication to an external service

CHAPTER 2

Installing Nodogsplash

2.1 OpenWrt

- Have a router working with OpenWrt. At the time of writing, Nodogsplash has been tested with OpenWrt 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) [avaiable 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.

CHAPTER 3

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 definiton 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 appropiate “Target System” and “Target Profile” in the menuconfig menu and build the image.

```
make defconfig
make menuconfig
make
```

CHAPTER 4

Frequenctly 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 usally also valid for v0.9.

v2 was developed while version v1 wasn't released. In v2 the http code got replaced by libmicrohttpd as well the template engine got rewritten. Many feature were defunct because of this procedure.

v3 cleans up the source code and adds the binauth feature to be able to call an external script for authentication. This is similar to the old binvoucher feature, but more flexible. 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

This is a very smooth update with full compatibility.

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

You can, if you don't use:

- BinVoucher (there is a PR#144)

4.4 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 OpenWrt removed the IMQ (Intermediate queueing device) support. We're looking for somebody who to fix this!

However the OpenWrt package, SQM Scripts, 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.

4.5 Is https:// redirection supported?

No. We believe this is the wrong way to do it, 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.

CHAPTER 5

How nodogsplash works

A wireless router running OpenWrt has two or more interfaces; nodogsplash manages one of them. This will typically be br-lan, the bridge to both the wireless and wired LAN; or the wireless lan interface may be named something else if you have broken the br-lan bridge to separate the wired and wireless LAN's.

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 builtin 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

Nodogsplash also optionally implements basic traffic control on its managed interface. This feature lets you specify the maximum aggregate upload and download bandwidth that can be taken by clients connected on that interface. Nodogsplash implements this functionality by enabling two intermediate queue devices (IMQ's), one for upload and one for download, and attaching simple rate-limited HTB qdiscs to them. Rules are inserted in the router's iptables mangle PREROUTING and POSTROUTING tables to jump to these IMQ's. The result is simple but effective tail-drop rate limiting (no packet classification or fairness queueing is done).

Note: IMQ is not included anymore by OpenWrt Attitude Adjustment (12.09).

CHAPTER 6

Forwarding Authentication Service (FAS)

6.1 Overview

Nodogsplash (NDS) can support external (to NDS) authentication. The BinVoucher process was derived to support this and has been called Forwarding Authentication. This is a non trivial function and although partially implemented in early versions, is not implemented at all in version 2, at the time of writing.

Fortunately, Forwarding Authentication can be done without any modification to the core NDS code and in a way that is compatible with all versions, pre v1 beta through to the current release of v2.

The defacto industry standard Captive Portal Detection (CPD), present on almost all devices these days, invokes the NDS splash page with various parameters passed to the splash page by NDS, including the client access token.

It is a simple matter to pass this token to an external Forwarding Authentication Service (FAS) by using a redirect in the splash page.

For a client to access this external service, the ip address and port number of the service must be added to the NDS walled garden in nodogsplash.conf or its equivalent UCI config file if running under LEDE/OpenWrt.

Included are various configuration files and remote php scripts, intended as an example implementation of FAS to demonstrate the methods.

6.2 FAS Installation

NOTE: USING HTTPS. Your FAS can be an https server, but self signed certificates will throw dire "Here Be Dragons" warnings on your client devices when the redirection to your FAS takes place. Also even if using a registered CA all browsers will still return a security error on returning to Nodogsplash. This can be prevented by using wget to return to Nodogsplash from your FAS script instead of an html GET.

The contents of the FAS etc folder should be placed in the /etc folder of your NoDogSplash router, overwriting existing files.

The following two files should be edited as follows.

1: /etc/config/nodogsplash should be edited to reflect the ip address and port of your FAS service as described in the comments in the example file. Your FAS can reside on your Nodogsplash router, a web server on your LAN, or a web server on the internet.

2: /etc/nodogsplash/htdocs/splash.html should also be edited to reflect the URL of your FAS service as indicated in the comments in the example file. Take note of the USING HTTPS warning above. A typical URL could be http://my-fas.net/nodog/fas.php?auth.... etc.

Running FAS on your Nodogsplash router: The example FAS service will run fairly well on uhttpd (the web server that serves Luci) on an LEDE/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 16MB flash and 64MB ram is recommended.

Running on uhttpd: Install the modules php7 and php7-cgi on LEDE for the simple example. Further modules may be required when you write your own php scripts 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.

Finally, reboot the router to start NoDogSplash in FAS mode.

The example file "users.dat" contains a list of usernames and passwords.

NOTE: /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.

CHAPTER 7

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
    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"
    # 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.
```

(continues on next page)

(continued from previous page)

```
    # ndsctl_auth: Client was authenticated manually by the ndsctl tool.
    # ndsctl_deauth: Client was deauthenticated by the ndsctl tool.
    # shutdown_deauth: Client was deauthenticated by Nodogsplash terminating.
    ;;
esac

exit 0
```

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 acknowlege 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.

CHAPTER 8

Using ndsctl

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 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 to persist across nodogsplash restarts, you have to edit the configuration file.)

CHAPTER 9

Customizing nodogsplash

The default shipped configuration is intended to be usable and reasonably secure as-is for basic internet sharing applications, but it is customizable.

- To change basic nodogsplash settings, edit the configuration file:

  `/etc/nodogsplash/nodogsplash.conf`

In the configuration file, a FirewallRule has the form:

```
FirewallRule permission [protocol [port portrange] [to ip]
```

where

- *permission* is required and must be allow, block, drop, log, or ulog.
- *protocol* is optional. If present, it must be tcp, udp, icmp, or all. Defaults to all.
- port *portrange* is optional. If present, protocol must be tcp or udp. portrange can be a single integer port number, or a colon-separated port range, e.g. 1024:1028. Defaults to all ports.
- *to ip* is optional. If present, ip must be a decimal dotted-quad IP address with optional mask. Defaults to 0.0.0.0/0, i.e. all addresses.
- To change the contents of the splash page, edit the splash page file:

  `/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. (Alternatively, you can use a GET-method HTML form to send this information to the nodogsplash server; see below.) As a simple example:

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

- *$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 also available and can be useful if you want 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. As a simple example:

```
<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 *$gatewaymac* The respective addresses of the client or gateway. This might be usefull in cases where the data needs to be forwarded to some other place by the plash page itself.
- *$nclients* and *$maxclients* User stats. Usefull when you need to display something like "n of m users online" on the splash site.
- *$uptime* The time Nodogsplash is running.
- A list of all available variables are included in the splash.html file.
- If the user accesses the splash page while being 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.

CHAPTER 10

Debugging nodogsplash

- To see maximally verbose debugging output from nodogsplash, edit the /etc/init.d/nodogsplash file to set the OPTIONS variable to the flags "-s -d 7", restart or reboot, and view messages with logread. The -s flag logs to syslog; the -d 7 flag sets level 7, LOG_DEBUG, for debugging messages (see syslog.h). You don't want to run with these flags routinely, as it will quickly fill the syslog circular buffer, unless you enable remote logging. A lower level of logging, for example level 5, LOG_NOTICE, is more appropriate for routine use (this is the default). Logging level can also be set using ndsctl as shown above. Alternatively, you can set the flag -f instead of -s, and restart. This will run nodogsplash in the foreground, logging to stdout.
- 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.
- Nodogsplash operates by marking packets (and, if traffic control is enabled, passing packets through intermediate queueing devices). Most QOS packages will also mark packets and use IMQ's. Therefore one or both of Nodogsplash and a QOS package may malfunction if used together. Potential conflicts may be investigated by looking at your overall iptables setup. To check to see all the rules in, for example, the mangle table chains, run

  ```
  iptables -t mangle -v -n -L
  ```

  For extensive suggestions on debugging iptables, see for example Oskar Andreasson's_tutorial.

CHAPTER 11

TODO List

Not all features are finished or working as properly 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).
- Show a site when the users authentication was rejected, e.g. because the user exeeded the quota

CHAPTER 12

Indices and tables

- genindex
- search