LiteServer v1.0

LiteServer is a GO implementation of an HTTP server that supports (enforces) authentication, compaction and encryption to server locally files for LiteFoil.

Usage

We implement a basic web server that runs locally on your computer with the following options:

- `-f Path` (games directory)
- `-l Port` (default 8080)
- `-m MOTD` (default "LiteServer")
- `-p Password` (default "foil")
- `-s IP` (server IP or FQDN)
- `-u User` (default "lite")

Example

Suppose that your files are inside `/home/ahoy/mygames`. This will scan the directory for files with valid extensions (`nsp` , `nsz` , `xci` , `xcz`) and index then. We normally use the names in the following form: `NAME [TID][vN].EXT`. Where `NAME` is the name of the game, `TID` is it's *title identification* (a huge number), `N` is the release version, normally `0` , and `EXT` a valid extension.

```
$ ./liteserver-linux-amd64 -s 192.168.0.10 -f /home/ahoy/mygames -m "My Local
LiteServer" -u lite -p foil -l 8080
2026/05/09 14:20:57 LiteServer running on 0.0.0.0:8080
```

So, on your LiteFoil, you just need to fullfill the URL to `http://192.168.0.10:8080/` , and the user and password's defined to load your own local catalog.

What LiteFoil and LiteServer suports

Custom Index

A custom LiteFoil index is a special JSON file that contains a list of a files. To load this index, you can either serve it over a HTTP web server.

Basic Format

```
{
    "files": ["https://url1", "sdmc:/url2"],
    "success": "motd text here"
}
```

Detailed Format

```
{
    "files": [
        {
            "url": "https://url1",
            "size": 1000
        },
        {
            "url": "https://url2",
            "size": 3000
        }
    ],
    "success": "motd text here"
}
```

Supported Protocols

We do not implement all supported protocols from the original Tinfoil as we expect LiteFoil be used in the common situations, not everyone.

HTTP / HTTPS

LiteFoil downloads json and parses out the links. LiteFoil is known to work with LiteServer, Apache, and Nginx. Though it should work with any HTTP server that supports ranged requests.

Headers Sent

LiteFoil will send a few custom headers when requesting a directory only (not files):

- A unique user fingerprint is sent via `UID: XXXXXXXXXXXXXX`.
- The client's LiteFoil version is sent via `Version: 20.0`.

LiteFoil also supports basic autorization:

- Basic HTTP authentication is supported, to prevent unauthorized users from accessing your files.
- OAuth for Google Drive files (need to create the files `credentials.json` and `gdrive.token`).

URL Format

Both files and folders follow the same format. If specifying by the google file id, use `gdrive:AAAAAAAAAAAAAAAA` (notice lack of forward slash). LiteFoil supports OAuth, so just provide the index with the links and put the files `credentials.json` and `gdrive.token` inside `/switch/LiteFoil/`, so LiteFoil will read it before try to download the file.

Overriding File Names in URLs

Some URLs, like those from Google Drive, do not contain the correct filename. You can instruct LiteFoil to use a specific filename by appending a trailing fragment (hash):

```
gdrive:/abc102234098#filename.nsp
```

Or, if you are using Google Drive encrypted with `rclone`:

```
gdrivecrypt:/abc102234098#filename.nsp
```

To allow LiteFoil to correctly process an encrypted Google Drive, you need to add the following section to your LiteFoil `config.json` file:

```
...
"rcloneCrypt": {
        "catalogSizes": "plain",
        "enabled": true,
        "password": "PASSWORD_1",
        "password2": "PASSWORD_2"
    },
...
```

The passwords should match the obfuscated/encrypted ones generated by rclone in your rclone.conf file. Because the drive is encrypted and the files are compressed, please expect performance limits when using LiteFoil this way.

DRM

LitFoil supports encrypting custom index jsons, to prevent unauthorized redistribution of direct links to your private content.

How It Works

It works by generating a random AES-128-ECB key, encrypting your content with that key, and then wrapping the key with asymetrical RSA OAEP 2048-bit and sending it to LiteFoil.

Usage

You can use `encrypt.py` python script inside the `tools` (with the supplied public key) to encrypt your index file! Run `encrypt.py input.json output.tfl` to encrypt via command line.

DISCLAIMER: AS-IS. NO WARRANTIES. USE AT YOUR OWN RISK.