

HTTPS & TLS

HTTPS

```
HTTPSWebServerEnableHTTPS(certFile, keyFile string) errorHTTPS
```

openssl

1. RSA

```
openssl genrsa -out server.key 2048
```

ECDSA

```
openssl ecparam -genkey -name secp384r1 -out server.key
```

2.

```
openssl req -new -x509 -key server.key -out server.crt -days 365
```

3. ()

```
openssl rsa -in server.key -out server.key.public
```

```
opensslman openssl(Ubuntu)
```

```
$ openssl genrsa -out server.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
unable to write 'random state'
e is 65537 (0x10001)

$ openssl req -new -x509 -key server.key -out server.crt -days 365
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CH
State or Province Name (full name) [Some-State]:SiChuan
Locality Name (eg, city) []:Chengdu
Organization Name (eg, company) [Internet Widgits Pty Ltd]:John.cn
Organizational Unit Name (eg, section) []:Dev
Common Name (e.g. server FQDN or YOUR name) []:John
Email Address []:john@johnng.cn

$ openssl rsa -in server.key -out server.key.public
writing RSA key

$ ll
total 20
drwxrwxr-x  2 john john 4096 Apr 23 21:26 ./
drwxr-xr-x 90 john john 4096 Apr 23 20:55 ../
-rw-rw-r--  1 john john 1383 Apr 23 21:26 server.crt
-rw-rw-r--  1 john john 1675 Apr 23 21:25 server.key
-rw-rw-r--  1 john john 1675 Apr 23 21:26 server.key.public
```

Content Menu

- [HTTPS](#)
 -
 -
- [HTTPSHTTP](#)
- [Let's Encrypt](#)
 - [Certbot](#)
 -
 -
 -

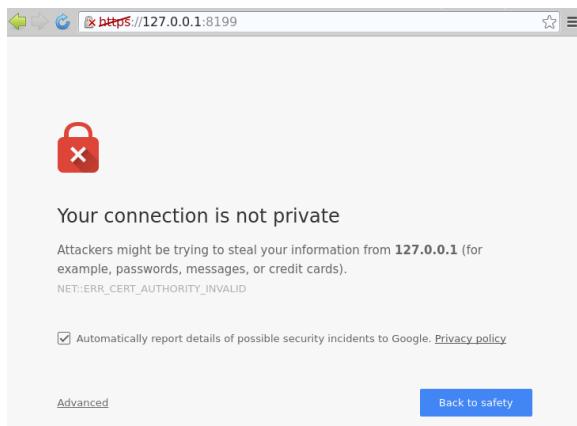
ghttp.ServerHTTPS

```
package main

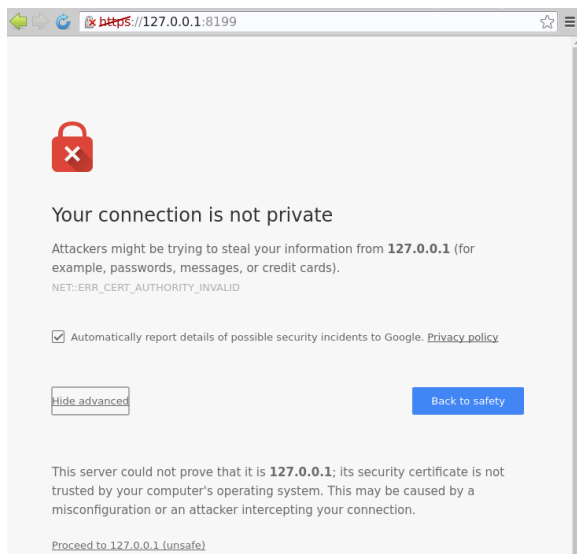
import (
    "github.com/gogf/gf/v2/net/ghttp"
)

func main() {
    s := ghttp.GetServer()
    s.BindHandler("/", func(r *ghttp.Request){
        r.Response.Writeln("HTTPS")
    })
    s.EnableHTTPS("/home/john/https/server.crt", "/home/john/https/server.
key")
    s.SetPort(8199)
    s.Run()
}
```

EnableHTTPSs.SetPort(8199)HTTPSs.SetHTTPSPort(8199)WebServerHTTPTTPS <https://127.0.0.1:8199/>



HTTPTTPS



AdvancedProceed to 127.0.0.1 (unsafe)



HTTPSHTTP

HTTPHTTPSWebServerghttp "WebServerHTTPSHTTP

```
package main

import (
    "github.com/gogf/gf/v2/net/ghttp"
)

func main() {
    s := ghttp.GetServer()
    s.BindHandler("/", func(r *ghttp.Request){
        r.Response.WriteLine("HTTPHTTPS")
    })
    s.EnableHTTPS("/home/john/https/server.crt", "/home/john/https/server.
key")
    s.SetHTTPSPort(443)
    s.SetPort(80)
    s.Run()
}
```

<http://127.0.0.1/> <https://127.0.0.1/> WebServer80443root/

HTTPS

```
func (s *Server) EnableHTTPS(certFile, keyFile string) error
func (s *Server) SetHTTPSPort(port ...int) error
```

HTTPSWebServerHTTPSSetPortHTTPWebServerHTTPSHTTP

Let's Encrypt

SSL

1. DV SSL : <https://cloud.tencent.com/product/ssl>
2. Let's Encrypt : <https://letsencrypt.org/>
3. CloudFlare SSL : <https://www.cloudflare.com/>
4. StartSSL : <https://www.startcomca.com/>
5. WosignSSL : <https://www.wosign.com/>
6. [loovit.net](https://www.loovit.net) AlphaSSL : <https://www.lowendtalk.com/entry/register?Target=discussion%2Fcomment%2F2306096>

Let's Encrypt

Let's Encrypt<https://letsencrypt.org/>

UbuntuLet's EncryptGoFrame

Certbot

Certbot<https://certbot.eff.org/>

Let's Encryptcertbot

```
sudo apt-get update
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:certbot/certbot
sudo apt-get update
sudo apt-get install certbot
```

```
certbot certonly --standalone -d --staple-ocsp -m --agree-tos
```

```
root@ip-172-31-41-204:~# certbot certonly --standalone -d goframe.org --
staple-ocsp -m john@goframe.org --agree-tos
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator standalone, Installer None
Starting new HTTPS connection (1): acme-v02.api.letsencrypt.org
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for goframe.org
Waiting for verification...
Cleaning up challenges
```

IMPORTANT NOTES:

- Congratulations! Your certificate and chain have been saved at:
/etc/letsencrypt/live/goframe.org/fullchain.pem
Your key file has been saved at:
/etc/letsencrypt/live/goframe.org/privkey.pem
Your cert will expire on 2019-01-25. To obtain a new or tweaked
version of this certificate in the future, simply run certbot
again. To non-interactively renew **all** of your certificates, run
"certbot renew"
- If you like Certbot, please consider supporting our work by:

Donating to ISRG / Let's Encrypt: <https://letsencrypt.org/donate>
Donating to EFF: <https://eff.org/donate-le>

/etc/letsencrypt/

```
/etc/letsencrypt/live/goframe.org/fullchain.pem
/etc/letsencrypt/live/goframe.org/privkey.pem
```

```
package main

import (
    "github.com/gogf/gf/v2/net/ghttp"
)

func main() {
    s := ghttp.GetServer()
    s.BindHandler("/", func(r *ghttp.Request){
        r.Response.Writeln("HTTPS")
    })
    s.EnableHTTPS("/etc/letsencrypt/live/goframe.org/fullchain.pem", "/etc
/etc/letsencrypt/live/goframe.org/privkey.pem")
    s.Run()
}
```

3

```
certbot renew
```

1crontab

```
# `GoFrame`WebServer
0 3 * * * certbot renew --quiet --renew-hook "kill -SIGUSR1 $(pidof )"
```

2nginx

```
# 80WebServer
0 3 * * * service nginx stop && certbot renew --quiet --renew-hook
"service nginx start"
```

certbot renewnginx

```
# 80WebServer
0 3 * * * service nginx stop && certbot renew --quiet --renew-hook
"service nginx start"
5 3 * * * service nginx start
```