Innopolis University

SYSTEM AND NETWORKING ENGINEERING

Classical Internet Applications

LABORATORY REPORT 8

Web Servers

Student Name	**Student ID**
Sergey Grebennikov	47611

Lecturer:

Rasheed Hussain

Submission Date : October 12, 2017

Contents

1 Introduction

A web server is a computer system that processes an HTTP request from a client. Known open source web servers are Apache HTTP Server, Nginx and Lighttpd. The web server stores, processes and delivers web pages to clients. The web page is usually an HTML document. Clients use an user agent to query the web server. The user agent is most often a web browser.

Initial Settings:

- Web Server: **nginx**
- IP-address: **188.130.155.46**
- DNS implementation: **Unbound+NSD**
- Email: **sergey@st13.os3.su**

2 History

1. Web servers appeared with more functionality, performance, reliability, and security. In addition, there were some marketing strategies for proprietary Web servers.

3 Installation

3.1 Download the Nginx and Check the Signature

```
# cd /usr/local/src/
# wget -c https://nginx.org/download/nginx-1.13.5.tar.gz
# wget https://nginx.org/download/nginx-1.13.5.tar.gz.asc
# gpg nginx-1.13.5.tar.gz.asc
gpg: assuming signed data in 'nginx-1.13.5.tar.gz'
gpg: Signature made Tu 05 sep 2017 18:31:16 MSK using RSA key ID A1C052F8
gpg: Can't check signature: public key not found
# gpg --keyserver pgpkeys.mit.edu --recv-key A1C052F8
gpg: requesting key A1C052F8 from hkp server pgpkeys.mit.edu
gpg: key A1C052F8: public key "Maxim Dounin <mdounin@mdounin.ru>" imported
gpg: key A1C052F8: public key "Maxim Dounin <mdounin@mdounin.ru>" imported
gpg: no ultimately trusted keys found
gpg: Total number processed: 2
gpg:               imported: 2  (RSA: 2)
# gpg nginx-1.13.5.tar.gz.asc
gpg: assuming signed data in 'nginx-1.13.5.tar.gz'
gpg: Signature made Tu 05 sep 2017 18:31:16 MSK using RSA key ID A1C052F8
gpg: Good signature from "Maxim Dounin <mdounin@mdounin.ru>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner
Primary key fingerprint: B0F4 2533 73F8 F6F5 10D4  2178 520A 9993 A1C0 52F8
# wget https://nginx.org/keys/mdounin.key
# gpg --import mdounin.key
gpg: key A1C052F8: "Maxim Dounin <mdounin@mdounin.ru>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1
# gpg --edit-key mdounin@mdounin.ru trust
...
Your decision? 5
...
gpg> q
# gpg nginx-1.13.5.tar.gz.asc
gpg: assuming signed data in 'nginx-1.13.5.tar.gz'
gpg: Signature made Tu 05 sep 2017 18:31:16 MSK using RSA key ID A1C052F8
gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: Good signature from "Maxim Dounin <mdounin@mdounin.ru>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: B0F4 2533 73F8 F6F5 10D4  2178 520A 9993 A1C0 52F8
```

2. Old branches of Apache and nginx are still supported because they were installed long ago and they have highly available web resources for which the downtime is very critical, which can be caused by the integration of a new version of the web server.

what else can be the reasons?

These projects are open source software that can be maintained by anyone interested in doing this. Also, they are a vital part of every Linux distributions and each distribution maintainers patch bugs in these branches on their own.

3.2 Nginx Installation

Installing **nginx** dependencies

```
# wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-8.41.tar.gz
# tar -zxf pcre-8.41.tar.gz
# cd pcre-8.41
# ./configure
# make
# make install
# cd ..
# wget http://zlib.net/zlib-1.2.11.tar.gz
# tar xvf zlib-1.2.11.tar.gz
# cd zlib-1.2.11
# ./configure
# make
# make install
# cd ..
# wget http://www.openssl.org/source/openssl-1.0.2k.tar.gz
# tar xvf openssl-1.0.2k.tar.gz
# cd openssl-1.0.2k
# ./config --prefix=/usr
# make
# make install
# cd ..
```

Configuring the Build Options

```
# tar xvf nginx-1.13.5.tar.gz
# cd nginx-1.13.5
# ./configure --sbin-path=/usr/local/nginx/nginx --conf-path=/usr/local/nginx/nginx.conf
 ↪  --pid-path=/usr/local/nginx/nginx.pid --with-pcre=../pcre-8.41
 ↪  --with-zlib=../zlib-1.2.11 --with-http_ssl_module --with-stream --with-mail=dynamic
# make
# make install
# exit
$ sudo /usr/local/nginx/nginx
nginx: [emerg] listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)
nginx: [emerg] listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)
nginx: [emerg] listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)
nginx: [emerg] listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)
nginx: [emerg] listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)
nginx: [emerg] still could not bind()
```

Troubleshooting

```
$ netstat -anpt | grep 80
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp  0  0 188.130.155.46:39156    213.180.204.179:443     ESTABLISHED 2789/yandex-disk
tcp6 0  0 :::80                   :::*                    LISTEN      -
$ sudo kill 2789
$ netstat -anpt | grep 80
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp  0      0 188.130.155.46:39156    213.180.204.179:443     TIME_WAIT   -
tcp6 0      0 :::80                   :::*                    LISTEN      -
$ sudo netstat -anpt | grep 80
tcp        0      0 188.130.155.46:39156    213.180.204.179:443     TIME_WAIT   -
tcp6        0        0 :::80        :::*        LISTEN        1526/apache2
$ sudo kill 1526
$ sudo netstat -anpt | grep 80
tcp        0      0 188.130.155.46:39156    213.180.204.179:443     TIME_WAIT   -
$ sudo apt remove apache2
...
The following packages will be REMOVED:
  apache2
...
```

Starting **nginx**

```
$ sudo /usr/local/nginx/nginx
$ sudo netstat -anpt | grep 80
tcp   0    0 0.0.0.0:80     0.0.0.0:*     LISTEN     4097/nginx
$ curl -I 127.0.0.1
HTTP/1.1 200 OK
Server: nginx/1.13.5
Date: Thu, 21 Sep 2017 03:52:52 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Wed, 20 Sep 2017 19:01:07 GMT
Connection: keep-alive
ETag: "59c2baf3-264"
Accept-Ranges: bytes
```

NGINX systemd service file

```
$ sudo vim /lib/systemd/system/nginx.service
```

```
[Unit]
Description=The NGINX HTTP and reverse proxy server
After=syslog.target network.target remote-fs.target nss-lookup.target

[Service]
Type=forking
PIDFile=/usr/local/nginx/nginx.pid
ExecStartPre=/usr/sbin/nginx -t
ExecStart=/usr/sbin/nginx
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s QUIT $MAINPID
```

```
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

```
$ sudo ln -s /lib/systemd/system/nginx.service
 ↪  /etc/systemd/system/multi-user.target.wants/
$ sudo systemctl daemon-reload
```

Troubleshooting

```
$ sudo systemctl start nginx.service
Job for nginx.service failed because the control process exited with error code.
 ↪  See "systemctl status nginx.service" and "journalctl -xe" for details.
$ sudo nginx -s stop
nginx: [error] open() "/usr/local/nginx/nginx.pid" failed (2: No such file or
 ↪  directory)
$ sudo touch nginx.pid
$ sudo nginx -s stop
nginx: [error] invalid PID number "" in "/usr/local/nginx/nginx.pid"
$ ps ax | grep nginx
 4002 ?        Ss     0:00 nginx: master process ./nginx
 4004 ?        S      0:00 nginx: worker process
24081 pts/1    S+     0:00 grep --color=auto nginx
$ sudo kill 4002
$ sudo systemctl start nginx
$ sudo systemctl status nginx
nginx.service - The NGINX HTTP and reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset:
 ↪  enabled)
   Active: active (running) since Th 2017-09-21 15:00:41 MSK; 5s ago
```

4 Virtual Hosts

3. Implement virtual hosting in the web server for three virtual domains (sergey.st13.os3.us, inno.st13.os3.su, sport.st13.os3.su)

   Set Up New Document Root Directories

```
$ sudo mkdir -p /var/www/sergey.st13.os3.su/html
$ sudo mkdir -p /var/www/inno.st13.os3.su/html
$ sudo mkdir -p /var/www/sport.st13.os3.su/html
$ sudo chown -R $USER:$USER /var/www/inno.st13.os3.su/html/
$ sudo chown -R $USER:$USER /var/www/sergey.st13.os3.su/html/
$ sudo chown -R $USER:$USER /var/www/sport.st13.os3.su/html/
$ sudo chmod -R 755 /var/www
```

   Configure **nginx.conf** file

```
$ sudo vim /usr/local/nginx/nginx.conf
```

```
...
http {
...
    server {
        listen 80;
        listen [::]:80;

        root /var/www/sergey.st13.os3.su/html;
        index index.html index.htm index.nginx-debian.html;

        server_name sergey.st13.os3.su www.sergey.st13.os3.su;

        location / {
                try_files $uri $uri/ =404;
        }
}

    server {
        listen 80;
        listen [::]:80;

        root /var/www/inno.st13.os3.su/html;
        index index.html index.htm index.nginx-debian.html;

        server_name inno.st13.os3.su www.inno.st13.os3.su;

        location / {
                try_files $uri $uri/ =404;
        }
    }

    server {
        listen 80;
        listen [::]:80;

        root /var/www/sport.st13.os3.su/html;
        index index.html index.htm index.nginx-debian.html;

        server_name sport.st13.os3.su www.sprot.st13.os3.su;

        location / {
                try_files $uri $uri/ =404;
        }
    }
...
}
```

Avoid a possible hash bucket memory problem

```
$ sudo vim /usr/local/nginx/nginx.conf
```

```
http {
    . . .

    server_names_hash_bucket_size 64;

    . . .
}
```

Check settings and restart **nginx**

```
$ sudo nginx -t
nginx: the configuration file /usr/local/nginx/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/nginx.conf test is successful
$ sudo systemctl restart nginx
```

Add new records in the zone file

```
$ sudo vim /usr/local/etc/nsd/st13.os3.su.zone
```

```
...
sergey  IN      A       188.130.155.46
inno    IN      A       188.130.155.46
sport   IN      A       188.130.155.46
...
```

Signing zone file

```
$ sudo ldns-signzone st13.os3.su.zone Kst13.os3.su.+008+15514
 ↪  Kst13.os3.su.+008+36381
$ sudo systemctl stop nsd
$ sudo systemctl start nsd
```

4. Create a simple, unique HTML page for each virtual host

   sergey virtual host

```
$ vim /var/www/sergey.st13.os3.su/html/index.html
```

```
<html>
    <head>
        <title>Welcome</title>
                <style>
                a:link {
                        color: green;
                        background-color: transparent;
                        text-decoration: none;
                }
                a:visited {
                        color: pink;
                        background-color: transparent;
                        text-decoration: none;
                }
                a:hover {
                        color: red;
                        background-color: transparent;
                        text-decoration: underline;
                }
                a:active {
                        color: yellow;
                        background-color: transparent;
                        text-decoration: underline;
                }
                h1 {
                         text-align: center;
                }
                </style>
    </head>
    <body>
          <h1>Success!  The <font color="red">sergey.st13.os3.su</font> virtual host is working!</h1>
          <form method="POST" action="junk.cgi">
          <input type=text name="birthyear">
          <input type=submit name=press value=" OK ">
          </form>
          <br>
          <br>
          <a href="SSI.shtml">SSI</a>
          <br>
          <a href="CGI.html">CGI</a>
    </body>
</html>
```

inno virtual host

```
$ vim /var/www/inno.st13.os3.su/html/index.html
```

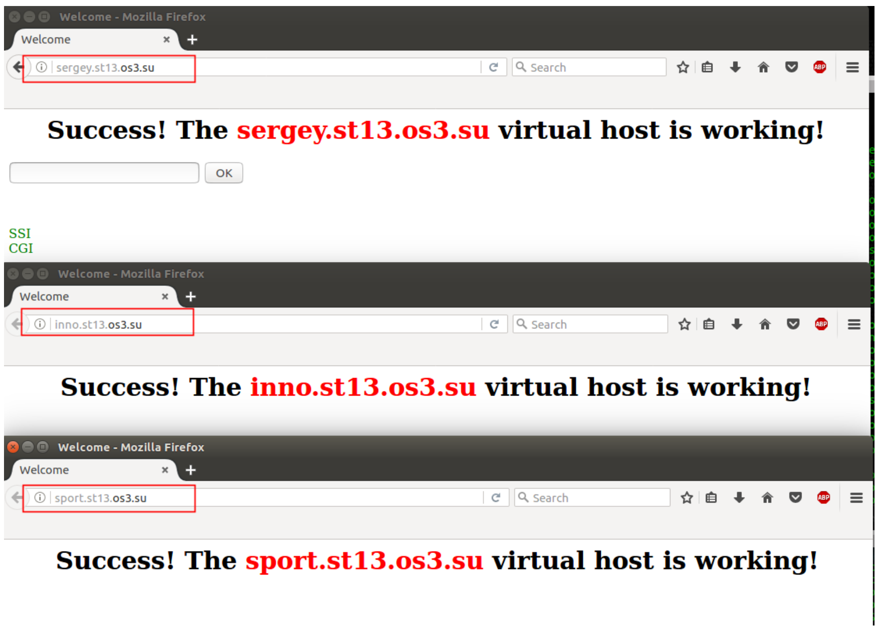
```
<html>
    <head>
        <title>Welcome</title>
    </head>
    <body>
        <h1>Success!  The inno.st13.os3.su virtual host is working!</h1>
    </body>
</html>
```

sport virtual host

```
$ vim /var/www/sport.st13.os3.su/html/index.html
```

```
<html>
    <head>
        <title>Welcome</title>
    </head>
    <body>
        <h1>Success!  The sport.st13.os3.su virtual host is working!</h1>
    </body>
</html>
```

Result: Figure 1



Success! The sergey.st13.os3.su virtual host is working!

Success! The inno.st13.os3.su virtual host is working!

Success! The sport.st13.os3.su virtual host is working!

Figure 1: Virtual hosts

5. HTTP/1.1 requests

   GET: used to load a resource

```
$ curl http://sergey.st13.os3.su
<html>
    <head>
        <title>Welcome</title>
    </head>
    <body>
        <h1>Success!  The sergey.st13.os3.su virtual host is
 ↪  working!</h1>
        <form method="POST" action="junk.cgi">
 <input type=text name="birthyear">
 <input type=submit name=press value=" OK ">
 </form>
    </body>
</html>
```

   HEAD: used to fetch only the headers

```
$ curl --head http://sergey.st13.os3.su
HTTP/1.1 200 OK
Server: nginx/1.13.5
Date: Thu, 21 Sep 2017 13:33:57 GMT
Content-Type: text/html
Content-Length: 167
Last-Modified: Thu, 21 Sep 2017 09:10:36 GMT
Connection: keep-alive
ETag: "59c3820c-a7"
Accept-Ranges: bytes
```

PUT: used to store a resource

```
$ curl --upload-file file  http://sergey.st13.os3.su
<html>
<head><title>404 Not Found</title></head>
<body bgcolor="white">
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.13.5</center>
</body>
</html>
```

POST: used to provide input in the body to server side scripts

```
$ curl --data "birthyear=1988&press=%20OK%20"
↪  http://sergey.st13.os3.su/
<html>
<head><title>405 Not Allowed</title></head>
<body bgcolor="white">
<center><h1>405 Not Allowed</h1></center>
<hr><center>nginx/1.13.5</center>
</body>
</html>
```

DELETE: used to delete a resource

```
$ curl -X DELETE http://sergey.st13.os3.su
<html>
<head><title>405 Not Allowed</title></head>
<body bgcolor="white">
<center><h1>405 Not Allowed</h1></center>
<hr><center>nginx/1.13.5</center>
</body>
</html>
```

OPTIONS: used to query the server options

```
$ curl -X OPTIONS http://sergey.st13.os3.su
<html>
<head><title>405 Not Allowed</title></head>
<body bgcolor="white">
<center><h1>405 Not Allowed</h1></center>
<hr><center>nginx/1.13.5</center>
</body>
</html>
```

5 Encryption

6. Configure **nginx** to support TLS

```
$ sudo vim /usr/local/nginx/nginx.conf
```

```
...
http {
...
  # SSL Settings
  ssl_session_cache shared:SSL:10m;
  ssl_session_timeout 5m;
  ssl_prefer_server_ciphers on;
  ssl_stapling on;
  resolver 8.8.8.8;

  server {
    listen 443 ssl;

    root /var/www/sergey.st13.os3.su/html;
    index index.html index.htm index.nginx-debian.html;

    server_name sergey.st13.os3.su www.sergey.st13.os3.su;

    keepalive_timeout   60;
    ssl_certificate      /usr/local/nginx/ssl/nginx.crt;
    ssl_certificate_key  /usr/local/nginx/ssl/nginx.key;
    ssl_protocols TLSv1.2;
    ssl_ciphers EECDH+CHACHA20:EECDH+AES128:RSA+AES128:EECDH+AES256:RSA+AES256:
     ↪  EECDH+3DES:RSA+3DES:!MD5;
    add_header Strict-Transport-Security 'max-age=604800';

    location / {
        try_files $uri $uri/ =404;
    }
  }
        ...
}
```

Check the syntax of the configuration file

```
$ sudo nginx -t
nginx: [warn] "ssl_stapling" ignored, issuer certificate not found for
 ↪  certificate "/usr/local/nginx/ssl/nginx.crt"
nginx: the configuration file /usr/local/nginx/nginx.conf syntax is ok
```

7. List of encryption standards that the Web server supports:

- EECDH+CHACHA20
- EECDH+AES128
- RSA+AES128
- EECDH+AES256
- RSA+AES256
- EECDH+3DES

- RSA+3DES

EECDH+AES128 means that the Diffie-Hellman algorithm on elliptical curves will be used to generate the session keys, and AES with a key length of 128 bits will be used as the traffic encryption algorithm.

8. Create the self-signed certificate

```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
 ↪  /usr/local/nginx/ssl/nginx.key -out /usr/local/nginx/ssl/nginx.crt
Generating a 2048 bit RSA private key
................................+++
...................................
...................................
...................................
....................+++
writing new private key to '/usr/local/nginx/ssl/nginx.key'
-----
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]:RU
State or Province Name (full name) [Some-State]:Innopolis
Locality Name (eg, city) []:Innopolis
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Innopolis University
Organizational Unit Name (eg, section) []:SNE
Common Name (e.g. server FQDN or YOUR name) []:st13.os3.su
Email Address []:sergey@st13.os3.su
```

Increasing key exchange security

```
$ sudo openssl dhparam -out /usr/local/nginx/ssl/dhparam.pem 2048
$ sudo vim /usr/local/nginx/nginx.conf
```

```
http {
...
  server {
    listen 443 ssl;
    root /var/www/sergey.st13.os3.su/html;
    index index.html index.htm index.nginx-debian.html;
    server_name sergey.st13.os3.su www.sergey.st13.os3.su;
    ...
    ssl_dhparam /usr/local/nginx/ssl/dhparam.pem;
    ...
  }

# Redirect all HTTP Request to HTTPS
  server {
    listen          80;
    listen     [::]:80;
    server_name     sergey.st13.os3.su;
    return          301 https://$server_name$request_uri;
   }
           ...
}
```

Check the syntax of the configuration file and reload **nginx**

```
$ sudo nginx -t
nginx: [warn] "ssl_stapling" ignored, issuer certificate not found for
 ↪  certificate "/usr/local/nginx/ssl/nginx.crt"
nginx: the configuration file /usr/local/nginx/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/nginx.conf test is
 ↪  successful

$ sudo systemctl reload nginx.service
```

9. Test secure web server using **openssl** and **curl**

```
$ curl https://sergey.st13.os3.su
curl: (60) SSL certificate problem: self signed certificate
More details here: https://curl.haxx.se/docs/sslcerts.html

curl performs SSL certificate verification by default, using a "bundle"
 of Certificate Authority (CA) public keys (CA certs). If the default
 bundle file isn't adequate, you can specify an alternate file
 using the --cacert option.
If this HTTPS server uses a certificate signed by a CA represented in
 the bundle, the certificate verification probably failed due to a
 problem with the certificate (it might be expired, or the name might
 not match the domain name in the URL).
If you'd like to turn off curl's verification of the certificate, use
 the -k (or --insecure) option.

$ openssl s_client -connect sergey.st13.os3.su:443
CONNECTED(00000003)
depth=0 C = RU, ST = Innopolis, L = Innopolis, O = Innopolis University, OU = SNE, CN = st13.os3.su, emailAddress = sergey@st13.os3.su
verify error:num=18:self signed certificate
verify return:1
depth=0 C = RU, ST = Innopolis, L = Innopolis, O = Innopolis University, OU = SNE, CN = st13.os3.su, emailAddress = sergey@st13.os3.su
verify return:1
---
Certificate chain
 0 s:/C=RU/ST=Innopolis/L=Innopolis/O=Innopolis University/OU=SNE/CN=st13.os3.su/emailAddress=sergey@st13.os3.su
   i:/C=RU/ST=Innopolis/L=Innopolis/O=Innopolis University/OU=SNE/CN=st13.os3.su/emailAddress=sergey@st13.os3.su
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIECzCCAvOgAwIBAgIJAI1iztqiNuONMA0GCSqGSIb3DQEBCwUAMIGbMQswCQYD
VQQGEwJSVTESMBAGA1UECAwJSW5ub3BvbGlzMRIwEAYDVQQHDAlJbm5vcG9saXMx
HTAbBgNVBAoMFElubm9wb2xpcyBVbml2ZXJzaXR5MQwwCgYDVQQLDANTTkUxFDAS
BgNVBAMMC3N0MTMub3MzLnN1MSEwHwYJKoZIhvcNAQkBFhJzZXJnZXlAc3QxMy5v
czMuc3UwHhcNMTcwOTIxMTgxMjA3WhcNMTgwOTIxMTgxMjA3WjCBmzELMAkGA1UE
BhMCUlUxEjAQBgNVBAgMCUlubm9wb2xpczESMBAGA1UEBwwJSW5ub3BvbGlzMR0w
GwYDVQQKDBRJbm5vcG9saXMgVW5pdmVyc2l0eTEMMAoGA1UECwwDU05FMRQwEgYD
VQQDDAtzdDEzLm9zMy5zdTEhMB8GCSqGSIb3DQEJARYSc2VyZ2V5QHN0MTMub3Mz
LnN1MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA128v+XxV8A0aGoez
fBFF2kuJvcaGFA0zV7RBEB6uGMgfB4SLKPidbsGuyAdSQxOsc/y0Xw87GIisv4+2
MW6/0qXekGybiyOgjLKhMf9JSF+LaN3/w0r2eMu0S/lGQWc13JLfGmzGmGGSP9HS
SWV33AArw72xvBISeDDlWXAQvDu9KwwhAYmG9erOEGGZNZPaue8QJqKrGRwegF7i
S4wEYVk3XUDd30Ih2Zy+wtaDhjCajgZUcCmHgLcuMUM5ODybZZmhM7EPiuw5nW99
DtLUCgHj6tHe47D3SQY9F0ysYhFe/cX19p515ed2x5W/VjAqpr2okT4o4CMLITCJ
eggszwIDAQABo1AwTjAdBgNVHQ4EFgQU0Cvg8L8k16kN90bx7whSC9xycuEwHwYD
VR0jBBgwFoAU0Cvg8L8k16kN90bx7whSC9xycuEwDAYDVR0TBAUwAwEB/zANBgkq
hkiG9w0BAQsFAAOCAQEAPJtRON8lkLOTfsUu1Vn25C33BImNq8b24BUg/biJzofv
8sIOp2NA8X0qIhdxeFaeamrrw0B5k89eoz6P5lrAuP8kjpm1Ue6m6rWwRkSY3C2s
pmtxIt7AcqOmlJISWu0/EXCLroSwJMGOab5xEeR9YsxE+frhXQuKvvffEFdtZQun
Y7E7wSAb4pH4rNor/oDeTTp+2AXthepeC++2GHBvbTUp7IHut/35WlUAswlubLbg
OS8wEIWGDjl50B0vGFCUUkB4HbiFB9aPD5JP9nYvQZOvSW/G2sJExjVG+swxWGe2
13QI2SyDZbowt2n5v4rGD3iON90in0FIOMUc15WJ2w==
-----END CERTIFICATE-----
subject=/C=RU/ST=Innopolis/L=Innopolis/O=Innopolis University/OU=SNE/CN=st13.os3.su/emailAddress=sergey@st13.os3.su
issuer=/C=RU/ST=Innopolis/L=Innopolis/O=Innopolis University/OU=SNE/CN=st13.os3.su/emailAddress=sergey@st13.os3.su
---
No client certificate CA names sent
Peer signing digest: SHA512
Server Temp Key: ECDH, P-256, 256 bits
---
SSL handshake has read 1714 bytes and written 433 bytes
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES128-GCM-SHA256
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES128-GCM-SHA256
    Session-ID: 866B819EEA0CD05F37B7E490362F6ADBE0237DEAEF4EF30BED865E478CB73C01
    Session-ID-ctx:
    Master-Key: 3BC26A8D864AFAFC38825DF502E69056B7F9EA2E9CAC5B165222ADE287B278FFDB18CEC038A8F271728BB5773A6B402D
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 300 (seconds)
    TLS session ticket:
    0000 - 80 81 d2 d7 09 e0 80 50-e2 77 3e d3 ac c5 54 0e   .......P.w>...T.
    0010 - 7a 72 55 fe 51 dd 83 04-76 f8 e8 43 b5 6e 8d 38   zrU.Q...v..C.n.8
    0020 - a3 9f fa d8 d8 39 b2 bc-04 c3 ce f9 17 8d e1 f0   .....9..........
    0030 - c8 13 3d f4 91 ed f0 df-15 c1 cc f9 7c 70 a4 5c   ..=.........|p.\
    0040 - 62 99 40 4b ca 99 80 24-d1 8c dd e0 89 b8 3e 0d   b.@K...$......>.
    0050 - df 6b d7 22 b0 3c 70 e2-94 ba 83 ce 51 87 89 0c   .k.".<p.....Q...
    0060 - bb ca b2 1d 14 51 79 97-49 41 22 81 b6 c6 b0 c3   .....Qy.IA".....
    0070 - 27 d7 2a d6 0d 1d 73 7c-86 d6 f4 04 fb ab ea 8c   '.*...s|........
    0080 - a1 71 cc c9 90 5d cd 35-e9 37 b6 7b 38 18 8b bc   .q...].5.7.{8...
    0090 - 89 17 01 61 d1 bd ac 46-8e 27 85 9d 86 b9 7d 95   ...a...F.'....}.
    00a0 - 8d 6c e9 09 76 24 22 f8-23 9e c8 92 e5 f4 e4 42   .l..v$".#......B

    Start Time: 1506020092
    Timeout   : 300 (sec)
    Verify return code: 18 (self signed certificate)
---
GET / HTTP/1.0

HTTP/1.1 200 OK
Server: nginx/1.13.5
Date: Thu, 21 Sep 2017 18:55:01 GMT
Content-Type: text/html
Content-Length: 297
Last-Modified: Thu, 21 Sep 2017 16:27:59 GMT
Connection: close
ETag: "59c3e88f-129"
Strict-Transport-Security: max-age=604800
Accept-Ranges: bytes

<html>
    <head>
        <title>Welcome</title>
    </head>
    <body>
        <h1>Success!  The sergey.st13.os3.su virtual host is working!</h1>
        <form method="POST" action="junk.cgi">
 <input type=text name="birthyear">
 <input type=submit name=press value=" OK ">
 </form>
    </body>
</html>
closed
```

10. HTTPS support can be enabled on all virtual hosts, but it's better to put information about the certificate file with its secret key on the **http** configuration level so that all servers inherit their single copy in memory.

rethink

Yes, I can enable HTTPS for all my virtual hosts by putting information about the certificate file with its secret key on the **http** configuration level or the **virtual hosts** configuration level.

6 Web Server Security

11. Nginx has several configuration options that govern access rights. We have the following ways to use these options on documents:

    - Limiting access to proxied HTTP resources
    - Restricting access with HTTP Basic authentication
    - Configuring authentication based on subrequest results
    - Restricting access by geographical location
    - Dynamic IP address blacklisting

12. SSI and CGI

    SSI

    Configuration file **nginx.conf**

```
http {
...
  server {
        listen 443 ssl;
        root /var/www/sergey.st13.os3.su/html;
        index index.html index.htm index.nginx-debian.html;
        server_name sergey.st13.os3.su   www.sergey.st13.os3.su;
        ...
# Include SSI
        location / {
                ssi on;
        }
   }
   ...
}
```

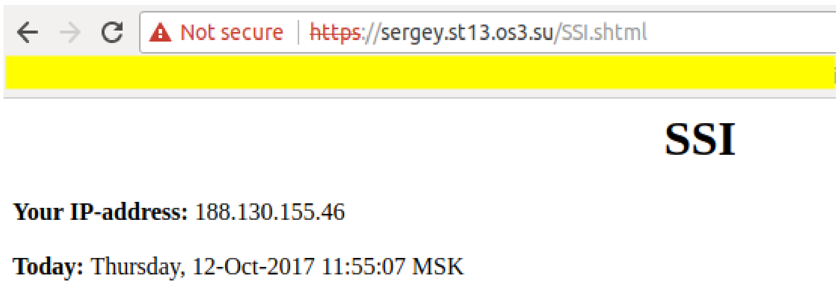
```
$ sudo nginx -t
nginx: [warn] "ssl_stapling" ignored, issuer certificate not found for
 ↪  certificate "/usr/local/nginx/ssl/nginx.crt"
nginx: the configuration file /usr/local/nginx/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/nginx.conf test is successful
$ sudo systemctl reload nginx.service
```

Create the web-page with the simple SSI instruction:

```
$ vim /var/www/sergey.st13.os3.su/html/SSI.shtml
```

```
<html>
    <head>
        <title>SSI</title>
    </head>
    <body>
                <center><h1>SSI</h1></center>
                <b>Server IP-address: </b><!--#echo var="REMOTE_ADDR" -->
               <br>
                <br>
                <b>Today: </b><!--#config timefmt="%A, %d-%b-%Y %H:%M:%S %Z" -->
                 ↪  <!--#echo var="DATE_LOCAL" -->
    </body>
</html>
```

Result: Figure 2


Not secure | https://sergey.st13.os3.su/SSI.shtml

SSI

Your IP-address: 188.130.155.46

Today: Thursday, 12-Oct-2017 11:55:07 MSK

Figure 2: The page with SSI instruction

CGI

Install **fcgiwrap**

```
$ sudo apt-get install fcgiwrap
```

Configuration file **nginx.conf**

```
...
http {
...
  server {
        listen 443 ssl;
        root /var/www/sergey.st13.os3.su/html;
        index index.html index.htm index.nginx-debian.html;
        server_name sergey.st13.os3.su   www.sergey.st13.os3.su;
        ...
    location /cgi-bin/ {
        fastcgi_pass unix:/var/run/fcgiwrap.socket;
        include /usr/local/nginx/fastcgi_params;
        fastcgi_param SCRIPT_FILENAME
         ↪  /var/www/sergey.st13.os3.su/html$fastcgi_script_name;
    }
  }
  ...
}
```

```
$ sudo nginx -t
nginx: [warn] "ssl_stapling" ignored, issuer certificate not found for certificate
 ↪  "/usr/local/nginx/ssl/nginx.crt"
nginx: the configuration file /usr/local/nginx/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/nginx.conf test is successful
$ sudo systemctl reload nginx.service
```

Create **form.py** file

```
$ mkdir /var/www/sergey.st13.os3.su/html/cgi-bin/
$ vim var/www/sergey.st13.os3.su/html/cgi-bin/form.py
```

```
#!/usr/bin/env python3
import cgi
import html

form = cgi.FieldStorage()
text1 = form.getfirst("TEXT_1", "Not Set")
text2 = form.getfirst("TEXT_2", "Not Set")
text1 = html.escape(text1)
text2 = html.escape(text2)

print("Content-type: text/html\n")
print("""<!DOCTYPE HTML>
        <html>
        <head>
            <meta charset="utf-8">
            <title>Form data processing</title>
        </head>
        <body>""")
```

```
print("<h1>Form data processing!</h1>")
print("<p>TEXT_1: {}</p>".format(text1))
print("<p>TEXT_2: {}</p>".format(text2))

print("""</body>
        </html>""")
```

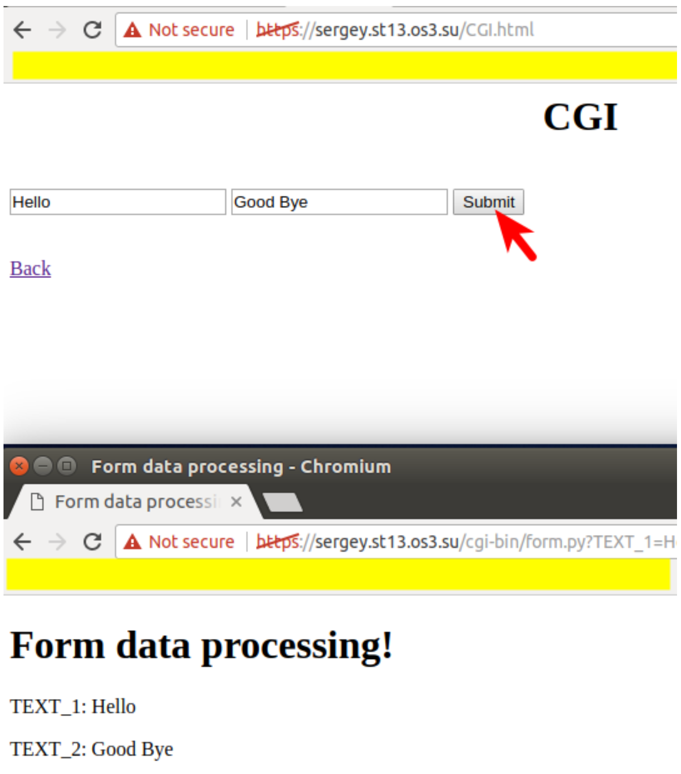
```
$ chmod 755 /var/www/sergey.st13.os3.su/html/cgi-bin/form.py
$ sudo service fcgiwrap restart
```

Create the web-page for Python CGI script:

```
$ vim /var/www/sergey.st13.os3.su/html/CGI.html
```

```
<html>
    <head>
        <title>CGI</title>
    </head>
    <body>
                <center><h1>CGI</h1></center>
                <br>
                <form action="/cgi-bin/form.py">
                <input type="text" name="TEXT_1">
                <input type="text" name="TEXT_2">
                <input type="submit">
                </form>
                <br>
                <a href="index.html">Back</a>
    </body>
</html>
```

Result: Figure 3


Not secure
https://sergey.st13.os3.su/CGI.html
CGI
Hello
Good Bye
Submit
Back
Form data processing - Chromium
Form data processi
Not secure
https://sergey.st13.os3.su/cgi-bin/form.py?TEXT_1=H

Form data processing!

TEXT_1: Hello

TEXT_2: Good Bye

Figure 3: Python CGI script operation

7 Conclusion

The primary function of a web server is to store, process and deliver web pages to clients. The communication between client and server takes place using the Hypertext Transfer Protocol (HTTP) and HTTPS(HTTP Secure).

While the primary function is to serve content, a full implementation of HTTP also includes ways of receiving content from clients. This feature is used for submitting web forms, including uploading of files.

Apache, IIS and Nginx are the most used web servers on the Internet. nginx [engine x] is an HTTP and reverse proxy server, a mail proxy server, and a generic TCP/UDP proxy server, originally written by Igor Sysoev.

8 References

[1] Commercial Nginx [https://www.nginx.com/?utm_source=from_http]

[2] Nginx Open Source [https://nginx.org/]

[3] Using curl to automate HTTP jobs [https://curl.haxx.se/docs/httpscripting.html#Certificates]

[4] Qualys. SSL Labs [https://www.ssllabs.com/index.html]

[5] How To Create an SSL Certificate on Nginx for Ubuntu 14.04 [https://www.digitalocean.com/community/tutorials/how-to-create-an-ssl-certificate-on-nginx-for-ubuntu-14-04]

[6] How To Set Up Nginx with HTTP/2 Support on Ubuntu 16.04 [https://www.digitalocean.com/community/tutorials/how-to-set-up-nginx-with-http-2-support-on-ubuntu-16-04]

[7] NGINX ADMIN GUIDE AND TUTORIAL [https://www.nginx.com/resources/admin-guide/?_ga=2.225970526.510425039.1506026136-1818424442.1506026136]

[8] Module ngx-http-ssi module [https://nginx.ru/en/docs/http/ngx_http_ssi_module.html]

[9] cgi — Common Gateway Interface support [https://pythonworld.ru/web/cgi-2.html]