Job's NGINX Hardening Notes (2023)

Job B.P. Douma

January 2, 2024

Contents

1 Introduction **1**
1.1 Installation media 1
1.2 Source code 2
1.3 Updating Docker Images 2
1.4 Save and load image into Docker 2
1.5 Interactive shell 2

2 Methodology & Results **2**
2.1 Update NGINX 2
2.2 Disable Server Tokens 2
2.3 Control Resources and Limits 2
2.4 Disable Unwanted HTTP Methods 3
2.5 Include Security Headers 3
2.6 Configure SSL on NGINX 3
2.7 Monitoring 4
2.8 Optimizing performance 4
2.9 Unprivileged user 4
2.10 Snyk Vulnerability report 5

1 Introduction

NGINX is a popular web server that is widely used in the industry. However, the default configuration of NGINX may have many vulnerabilities that can be exploited by attackers. Therefore, it is essential to harden the NGINX server to make it more secure. There are several ways to harden NGINX, such as disabling unwanted HTTP methods, disabling weak SSL/TLS protocols, and disabling weak cipher suites. Additionally, you can disable any unwanted modules and recompile NGINX to make it more secure.

Definition 1 (**NGINX**). *A BSD-licensed high-performance web server and reverse proxy server that supports HTTP, SMTP, POP3, and IMAP protocols on various platforms.*

In this article, we will explain the steps to achieve such hardening coming from a base NGINX image `nginx:1.25.3-alpine` (v1.25.3).

1.1 Installation media

1. Docker Desktop (v4.26.1): Docker (v24.0.7), build `afdd53b`
2. Docker Extension: NGINX Development Center (v0.0.3)
3. VS Code (v1.85.1)

4. Docker for VS Code (v1.28.0)
5. Git (v2.43.0)

1.2 Source code

The source code can be found on GitHub: `JobDouma/NGINX-Hardened`. Additionally the Docker image `jobdouma/nginx-hardening` can be run using the following command:

```
$ docker run --rm -it -p 8443:443 jobdouma/nginx-hardening
```

1.3 Updating Docker Images

```
$ docker images
$ docker inspect <image_id>
$ docker build -t <base_img:latest> -f <base_dockerfile> .
```

1.4 Save and load image into Docker

Save the Docker image as a tar/zip file. We can now copy the tar/zip file to a different machine, and run `docker load` to load the docker image from tar file.

```
$ docker save -o nginx-hardening.tar jobdouma/nginx-hardening:latest
$ docker load -i nginx-hardening.tar
```

1.5 Interactive shell

Create a new container with interactive shell:

```
$ docker ps
$ docker exec -it <container_id> /bin/bash
```

2 Methodology & Results

2.1 Update NGINX

```
$ apt update && apt upgrade -y
```

2.2 Disable Server Tokens

The `server_tokens` directive in NGINX, which displays the NGINX version number, can be vulnerable to information disclosure, so it's recommended to disable it.

```
server_tokens off;
```

2.3 Control Resources and Limits

To prevent DoS and potential buffer overflow attacks on NGINX, set buffer size limitations for all clients in the NGINX configuration file using the following directives:

```
client_body_buffer_size     1k;
client_header_buffer_size   1k;
client_max_body_size        1k;
large_client_header_buffers 2 1k;
```

The directives client_body_buffer_size, client_header_buffer_size, and client_max_body_size, specify the client request body, header, and maximum accepted body size.

The directive large_client_header_buffers 2 1k sets the maximum number and size of buffers for reading large client request headers, accepting a 2 kB data URI.

2.4 Disable Unwanted HTTP Methods

Disable unnecessary HTTP methods in the NGINX virtual host configuration file, allowing only GET, HEAD, and POST methods and filtering out DELETE and TRACE.

```
location / {
    limit_except GET HEAD POST { deny all; }
}
```

2.5 Include Security Headers

To enhance your NGINX web server, consider adding multiple HTTP headers.

2.5.1 X-Frame-Options

Nginx can reduce clickjacking attacks on browsers rendering pages within frames or iframes by configuring the X-Frame-Options header with ”SAMEORIGIN” value.

```
add_header X-Frame-Options "SAMEORIGIN";
```

2.5.2 HTTP Strict Transport Security

HTTP Strict Transport Security (HSTS) ensures pages load on encrypted connections, ensuring modern browsers support HTTPS. Adding the below line to the server section of your SSL config file, limits the age of connections to 63072000, to prevent plain text transmission.

```
add_header Strict-Transport-Security "max-age=63072000; includeSubdomains;
↪  preload";
```

2.5.3 CSP and XSS Protection

Content Security Policy (CSP) safeguards web servers from XSS and data injection attacks by adding a Content-Security-Policy header:

```
add_header Content-Security-Policy "default-src 'self' http: https: data:
↪  blob:
'unsafe-inline'" always;
```

The HTTP X-XSS-Protection header is supported by IE and Safari, but is not necessary for modern browsers unless a strong CSP is in place:

```
add_header X-XSS-Protection "1; mode=block";
```

2.6 Configure SSL on NGINX

Using the Dockerfile, a self-signed SSL certificate for the domain localhost with a validity of 365 days and a 2048-bit Diffie-Hellman parameter file is generated.

```
$ openssl req -subj '/CN=localhost' -x509 -newkey rsa:4096 -nodes -keyout
↪  /etc/nginx/ssl/default_key.pem -out /etc/nginx/ssl/default_cert.pem
↪  -days 365
$ openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048
```

The self-signed SSL certificate and Diffie-Hellman parameter for DHE cipher suites are stored in `/etc/nginx/ssl.conf` as follows:

```
ssl_certificate /etc/nginx/ssl/default_cert.pem;
ssl_certificate_key /etc/nginx/ssl/default_key.pem;
ssl_dhparam /etc/nginx/ssl/dhparam.pem;
```

We additionally configure SSL parameters to disable weak TLS protocols, weak ciphers and manage SSL sessions:

```
ssl_protocols TLSv1.2 TLSv1.3;
ssl_prefer_server_ciphers on;
ssl_ciphers
↪  'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECD
ssl_session_timeout 1d;
ssl_session_tickets off;
ssl_session_cache shared:ssl-http:50m;
```

2.7 Monitoring

Regularly monitor Nginx web server by enabling access and error logs.

```
access_log /var/log/nginx/access.log basic;
error_log /var/log/nginx/error.log debug;
```

In addition, monitoring tools such as Nagios, Zabbix or Prometheus can also be used to monitor the performance of the Nginx web server and detect any anomalies or potential security threats. However, due to time constraints I did not have enough time to implement this.

2.8 Optimizing performance

Optimizing performance for serving content in `/etc/nginx/http.conf`:

```
sendfile on;
tcp_nopush on;
tcp_nodelay  on;
keepalive_timeout 65;
types_hash_max_size 2048;
```

2.9 Unprivileged user

By default, NGINX image use "root" user but there is an "nginx" user in the same base image. So we need to specify this user with "USER" and give a permission some files for non-root nginx user. I changed ownership and mod using the following below and `USER nginx`.

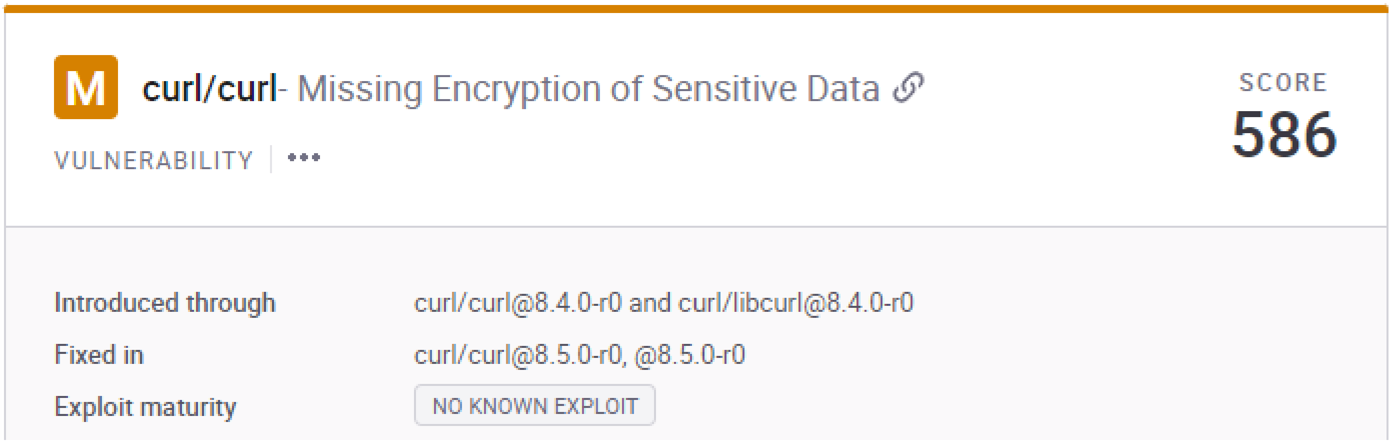
```
RUN chown -R nginx:nginx /app && chmod -R 755 /app && \
         chown -R nginx:nginx /var/cache/nginx && \
         chown -R nginx:nginx /var/log/nginx && \
         chown -R nginx:nginx /etc/nginx/conf.d
RUN touch /var/run/nginx.pid && \
         chown -R nginx:nginx /var/run/nginx.pid
```

However, unfortunately I was unable to install a minimal privileged user (non-root) in NGINX.

2.10 Snyk Vulnerability report

Snyk found two vulnerabilities (Figure 1 & 2). Both vulnerabilities were addressed by updating the currently installed version of `curl`:

```
curl 8.5.0 (x86_64-alpine-linux-musl) libcurl/8.5.0 OpenSSL/3.1.4
↪  zlib/1.2.13 brotli/1.0.9 libidn2/2.3.4 nghttp2/1.57.0
Release-Date: 2023-12-06
```



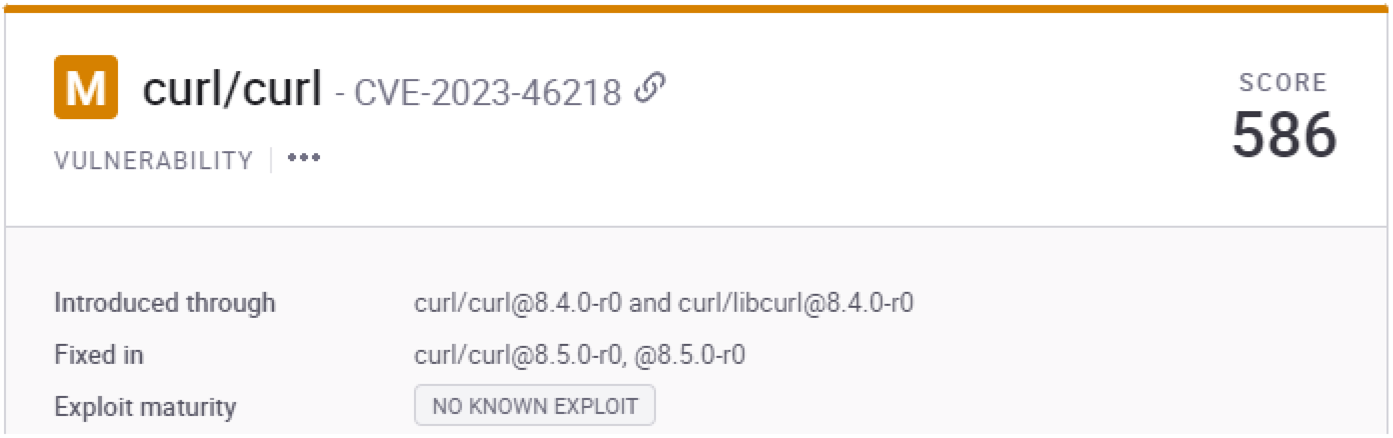
M **curl/curl**- Missing Encryption of Sensitive Data

VULNERABILITY | •••

SCORE
586

Introduced through	curl/curl@8.4.0-r0 and curl/libcurl@8.4.0-r0
Fixed in	curl/curl@8.5.0-r0, @8.5.0-r0
Exploit maturity	NO KNOWN EXPLOIT

Figure 1: curl/curl Missing Encryption of Sensitive Data.



M **curl/curl** - CVE-2023-46218

VULNERABILITY | •••

SCORE
586

Introduced through	curl/curl@8.4.0-r0 and curl/libcurl@8.4.0-r0
Fixed in	curl/curl@8.5.0-r0, @8.5.0-r0
Exploit maturity	NO KNOWN EXPLOIT

Figure 2: curl/curl CVE-2023-46218.