Боремся с POODLE на CentOS 5

Введение

Уверен, что каждый системный администратор слышал об этой нашумевшей уязвимости - POODLE. Более детальную информацию можно получить по следующим ссылкам

- https://www.us-cert.gov/ncas/alerts/TA14-290A
- http://security.stackexchange.com/questions/70719/ssl3-poodle-vulnerability
- https://www.imperialviolet.org/2014/10/14/poodle.html
- http://habrahabr.ru/company/dsec/blog/240499/

наша цель закрыть уязвимость, но при этом иметь возможность использовать современные версии протоколов TLSv1.1/TLSv1.2. В openssl-0.9.x доступна лишь старая версия TLSv1. Напомню, что на CentOS 5, на момент написания статьи, была досупна следующая версия openssl

```
# openssl version
OpenSSL 0.9.8e-fips-rhel5 01 Jul 2008

# openssl ciphers -v 'TLSv1.2'
Error in cipher list
3674:error:140E6118:SSL routines:SSL_CIPHER_PROCESS_RULESTR:invalid
command:ssl_ciph.c:836:
```

В принципе это касается не только CentOS 5, а любой относительно старой версии ОС, например Debian 6. Именно по этой причине тесты SSLLabs дают такой низкий результат

Например, на CentOS 6, эти команды будут выдавать примерно следующий результат

```
# openssl version
OpenSSL 1.0.1e-fips 11 Feb 2013

# openssl ciphers -v 'TLSv1.2' | head -3
ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AESGCM(256)
Mac=AEAD
ECDHE-ECDSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AESGCM(256)
Mac=AEAD
ECDHE-RSA-AES256-SHA384 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AES(256)  Mac=SHA384
```

Так как поддержка TLSv1.1/TLSv1.2 была добавлена только в openssl 1.0.0h/1.0.1, то у нас есть следующие варианты выхода из ситуации:

1. Обновить ОС, например до CentOS 6. Но к сожалению такая возможность есть не всегда.
2. Обновить openssl, но так как на него завязано практически вся система, то это чревато последствиями не стабильной работы системы в целом, вплоть до ее полной неработоспособности.
3. Обновить конкретное приложение и собрать его статически с нужной версией openssl.

Вот 3й вариант мы как раз и рассмотрим. Еще раз оговорюсь, данный вариант рассматривается как временное решение.

Подготовка системы

Итак, в нашем распоряжении CentOS 5.11 со всеми установленными обновлениями и подключенными репозитариями EPEL/RPMForge/IUS Community

```
# cat /etc/redhat-release
CentOS release 5.11 (Final)

# uname -a
Linux www.example.net 2.6.18-400.1.1.el5 #1 SMP Thu Dec 18 00:59:53 EST 2014
x86_64 x86_64 x86_64 GNU/Linu
```

Как всегда, для сборки нужного нам ПО я использую виртуальную машину, чтобы не «засорять» рабочие сервера. Но и тут возникают проблемы, собрать apache можно, но вот переносить его на другой сервер удовольствие еще то. Поэтому мы поступим проще - в качестве терминации https у нас будет выступать nginx. Который состоит всего из одного бинарного файла, что значительно упрощает перенос его на другой сервер.

Итак скачиваем необходимые версии nginx/openssl, распаковываем архивы и приступаем к сборке.

```
# cd /usr/src/redhat/BUILD/
# wget https://www.openssl.org/source/openssl-1.0.1j.tar.gz
# wget http://nginx.org/download/nginx-1.4.7.tar.gz
# tar -xvf nginx-1.4.7.tar.gz
# tar -xvf openssl-1.0.1j.tar.gz
# cd nginx-1.4.7
```

Компилировать и устанавливать openssl нет необходимости, nginx достаточно указать лишь папку с исходниками openssl
Для сборки nginx на чистом CentOS 5 с минимальным набором пакетов необходимо дополнительно установить следующие пакеты

```
# yum install rpm-build make gcc-c++ zlib-devel pcre-devel redhat-rpm-config
buildsys-macros
```

Компиляция nginx

Ну и собственно самое интересное - параметры сборки nginx

```
# ./configure --prefix=/etc/nginx \
--sbin-path=/usr/sbin/nginx --conf-path=/etc/nginx/nginx.conf  \
--error-log-path=/var/log/nginx/error.log  \
--http-log-path=/var/log/nginx/access.log  \
```

```
--pid-path=/var/run/nginx.pid  \
--lock-path=/var/run/nginx.lock  \
--http-client-body-temp-path=/var/cache/nginx/client_temp  \
--http-proxy-temp-path=/var/cache/nginx/proxy_temp  \
--http-fastcgi-temp-path=/var/cache/nginx/fastcgi_temp  \
--http-uwsgi-temp-path=/var/cache/nginx/uwsgi_temp  \
--http-scgi-temp-path=/var/cache/nginx/scgi_temp  \
--user=nginx --group=nginx  \
--with-http_ssl_module  \
--with-http_realip_module  \
--with-http_addition_module  \
--with-http_sub_module  \
--with-http_gunzip_module --with-http_gzip_static_module  \
--with-http_random_index_module  \
--with-http_secure_link_module  \
--with-http_stub_status_module  \
--with-mail --with-mail_ssl_module  \
--with-file-aio \
--with-openssl=/usr/src/redhat/BUILD/openssl-1.0.1j  \
--with-openssl-opt="zlib enable-camellia enable-seed enable-tlsext enable-
rfc3779 enable-cms enable-md2 no-mdc2 no-rc5 no-ec2m no-srp"
```

Особый интерес представляют последние два параметра. Здесь мы указываем путь к нужной версии openssl, а так же передаем параметры сборки самой openssl. Как я упоминал ранее, openssl нет необходимости собирать, как это обычно требуется для других программ. Ну и так же вам никто не мешает собрать необходимые модули для вашего окружения, например, вы можете добавить поддержку rtmp и т.п.

В результате мы получим примерно такой вывод команды

```
checking for OS
 + Linux 2.6.18-400.1.1.el5 x86_64
checking for C compiler ... found
 + using GNU C compiler
 + gcc version: 4.1.2 20080704 (Red Hat 4.1.2-55)
checking for gcc -pipe switch ... found
...
...
...
Configuration summary
  + using system PCRE library
  + using OpenSSL library: /usr/src/redhat/BUILD/openssl-1.0.1j
  + md5: using OpenSSL library
  + sha1: using OpenSSL library
  + using system zlib library

  nginx path prefix: "/etc/nginx"
  nginx binary file: "/usr/sbin/nginx"
  nginx configuration prefix: "/etc/nginx"
  nginx configuration file: "/etc/nginx/nginx.conf"
  nginx pid file: "/var/run/nginx.pid"
```

```
  nginx error log file: "/var/log/nginx/error.log"
  nginx http access log file: "/var/log/nginx/access.log"
  nginx http client request body temporary files:
"/var/cache/nginx/client_temp"
  nginx http proxy temporary files: "/var/cache/nginx/proxy_temp"
  nginx http fastcgi temporary files: "/var/cache/nginx/fastcgi_temp"
  nginx http uwsgi temporary files: "/var/cache/nginx/uwsgi_temp"
  nginx http scgi temporary files: "/var/cache/nginx/scgi_temp"
```

Запускаем процесс компиляции nginx

```
# make
make -f objs/Makefile
make[1]: Entering directory `/usr/src/redhat/BUILD/nginx-1.4.7'
cd /usr/src/redhat/BUILD/openssl-1.0.1j \
        && make clean \
        && ./config --prefix=/usr/src/redhat/BUILD/openssl-1.0.1j/.openssl
no-shared zlib enable-camellia enable-seed enable-tlsext enable-rfc3779
enable-cms enable-md2 no-mdc2 no-rc5 no-ec2m no-srp no-threads \
        && make \
        && make install LIBDIR=lib
make[2]: Entering directory `/usr/src/redhat/BUILD/openssl-1.0.1j'
...
...
...
        -lpthread -lcrypt -lpcre
/usr/src/redhat/BUILD/openssl-1.0.1j/.openssl/lib/libssl.a
/usr/src/redhat/BUILD/openssl-1.0.1j/.openssl/lib/libcrypto.a -ldl -lz
make[1]: Leaving directory `/usr/src/redhat/BUILD/nginx-1.4.7'
make -f objs/Makefile manpage
make[1]: Entering directory `/usr/src/redhat/BUILD/nginx-1.4.7'
sed -e "s|%%PREFIX%%|/etc/nginx|" \
                -e "s|%%PID_PATH%%|/var/run/nginx.pid|" \
                -e "s|%%CONF_PATH%%|/etc/nginx/nginx.conf|" \
                -e "s|%%ERROR_LOG_PATH%%|/var/log/nginx/error.log|" \
                < man/nginx.8 > objs/nginx.8
make[1]: Leaving directory `/usr/src/redhat/BUILD/nginx-1.4.7'
```

Собственно вот и все, все что нам надо сделать - скопировать бинарный файл objs/nginx на нужный нам сервер и перезапустить nginx.

Настройка nginx

Для полноты картины привожу настройки nginx, относящиеся к https

```
# cat /etc/nginx/conf.d/example.net.conf
server {
    listen 192.168.172.182:443;
    server_name  example.net www.example.net;
```

```
    charset utf-8;
    root /vhosts/example.net;

    ssl                  on;
    ssl_certificate      /etc/pki/nginx/example.net.crt;
    ssl_certificate_key  /etc/pki/nginx/example.net.key;

    ssl_session_cache shared:SSL:60m;
    ssl_session_timeout 10m;

    # Включаем поддержку HSTS(HTTP Strict Transport Security)
    add_header Strict-Transport-Security "max-age=31536000;
includeSubdomains;";

    ssl_dhparam /etc/pki/nginx/dhparam.pem;

    ssl_prefer_server_ciphers on;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers kEECDH+AES128:kEECDH:kEDH:-3DES:kRSA+AES128:kEDH+3DES:DES-
CBC3-
SHA:!RC4:!aNULL:!eNULL:!MD5:!EXPORT:!LOW:!SEED:!CAMELLIA:!IDEA:!PSK:!SRP:!SS
Lv2;

    location / {
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto https;
    }
}
```

Для генерации файла dhparam.pem необходимо выполнить команду

```
# openssl dhparam -out /etc/pki/nginx/dhparam.pem 2048
```

Соответственно на 127.0.0.1:8080 у нас слушает apache. Для удобства, я переименовал собранный бинарный файл и указал путь к нему

```
# cat /etc/sysconfig/nginx
# Configuration file for the nginx service.

NGINX=/usr/sbin/nginx_tlsv12
CONFFILE=/etc/nginx/nginx.conf
```

Тестирование

После данных манипуляций получаем оценку A+ на тестах SSLLabs.

Так же вы можете проверить из командной строки, что появилась поддержка TLSv1.2

```
# openssl s_client -tls1_2 -connect www.example.net:443
CONNECTED(00000003)
depth=3 C = US, O = "The Go Daddy Group, Inc.", OU = Go Daddy Class 2
Certification Authority
verify return:1
depth=2 C = US, ST = Arizona, L = Scottsdale, O = "GoDaddy.com, Inc.", CN =
Go Daddy Root Certificate Authority - G2
...
...
...
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES128-GCM-SHA256
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES128-GCM-SHA256
    Session-ID:
A742588B969CE3E4ED1CF73558CB11A1316A24E4297142B294AF4D574531AB45
    Session-ID-ctx:
    Master-Key:
CEB4A26915C3F64468617FAD18347377E7458C337CF2B776F8F2F5F9624898ACDEDCEC229A9C
30A256D1869E36BF7086
    Key-Arg   : None
    Krb5 Principal: None
    PSK identity: None
    PSK identity hint: None
    TLS session ticket lifetime hint: 300 (seconds)
...
...
...
```

Единственный минус - каждый раз после выхода новой версии nginx/openssl придется пересобирать данную связку. Поэтому есть повод продумать план миграции сервера на более современную ОС.