

Introduction

This document provides a comprehensive methodology for modifying Open5GS to support TLS 1.0. The objective is to introduce specific cryptographic vulnerabilities for controlled penetration testing and security evaluation. This process involves modifying Open5GS dependencies, configuring its cryptographic parameters, and verifying the implementation using security testing tools.

Environment Setup

- **Operating System:** Ubuntu 20.04
- **Open5GS Version:** v2.7.2-149-gbbfd462 (latest at the time of writing)
- **OpenSSL Version:** 1.1.1 (custom-built to enable TLS 1.0)
- **Verification Tool:** TLS-Scanner & TestSSL
- **Testing Utilities:** openssl s_client, curl

Methodology for TLS Version Downgrade

1. Install Required Dependencies

Ensure the system has the essential build tools and libraries:

```
sudo apt update && sudo apt install -y build-essential cmake git pkg-config libgnutls28-dev  
sudo apt install meson ninja-build gcc g++ flex bison git libtalloc-dev
```

2. Remove Existing OpenSSL

```
sudo apt remove --purge openssl libssl-dev -y  
sudo apt autoremove -y  
sudo rm /usr/bin/openssl  
sudo ln -s /usr/local/openssl/bin/openssl /usr/bin/openssl  
sudo ldconfig  
sudo rm /usr/lib/x86_64-linux-gnu/libssl.so.1.1  
sudo rm /usr/lib/x86_64-linux-gnu/libcrypto.so.1.1
```

3. Compile and Install OpenSSL 1.1.1

Since Open5GS is linked by default to OpenSSL 1.1.1f, it must be replaced with a custom-compiled version to permit the use of TLS 1.0.

```
cd /usr/local/src  
wget https://www.openssl.org/source/openssl-1.1.1.tar.gz  
mkdir openssl-1.1.1 && tar -xvzf openssl-1.1.1.tar.gz -C openssl-1.1.1 --strip-components=1  
cd openssl-1.1.1  
./config --prefix=/usr/local/openssl --openssldir=/usr/local/openssl  
make -j$(nproc)  
sudo make install
```

4. Create new symbolic links

```
sudo ln -s /usr/local/openssl/bin/openssl /usr/bin/openssl
sudo ln -s /usr/local/openssl/lib/libssl.so.1.1 /usr/lib/x86_64-linux-gnu/libssl.so.1.1
sudo ln -s /usr/local/openssl/lib/libcrypto.so.1.1 /usr/lib/x86_64-linux-gnu/libcrypto.so.1.1
sudo ldconfig
```

5. Update System's CA certificate

```
sudo apt update
sudo apt install --reinstall ca-certificates
sudo update-ca-certificates
```

6. Download Open5GS

```
git clone --recurse-submodules https://github.com/open5gs/open5gs.git
cd open5gs
```

7. Configure System to Use the Custom OpenSSL Build

Define environment variables to ensure the system prioritizes the newly compiled OpenSSL version:

```
export OPENSSL_ROOT_DIR=/usr/local/openssl
export OPENSSL_LIBRARIES=/usr/local/openssl/lib
export OPENSSL_INCLUDE_DIR=/usr/local/openssl/include
export PKG_CONFIG_PATH=/usr/local/openssl/lib/pkgconfig:$PKG_CONFIG_PATH
export PATH=/usr/local/openssl/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/openssl/lib:$LD_LIBRARY_PATH
```

8. Verify OpenSSL Version Alignment

Confirm that the system and Open5GS utilize the expected OpenSSL version:

```
openssl version -a
ldd $(which openssl) | grep ssl
```

Check If Meson Recognizes Your OpenSSL

```
pkg-config --modversion openssl
pkg-config --cflags openssl
pkg-config --libs openssl
```

Expected output:

```
1.1.1
-I/usr/local/openssl/include
-L/usr/local/openssl/lib -lssl -lcrypto
```

9. Configure Open5GS

Under /lib/sbi modify the nhttp2-server.c source code to explicitly allow TLS 1.0.

Lines 259-264:

Change "OGS_TLS_MAX_VERSION" to the highest version you would like to enable, this also counts for communication between the NF's.

```
#define OGS_TLS_MIN_VERSION TLS1_VERSION
#define OGS_TLS_MAX_VERSION TLS1_3_VERSION
```

Line 277 from:

```
if (SSL_CTX_set_cipher_list(ssl_ctx, DEFAULT_CIPHER_LIST) == 0)
```

To this:

```
if (SSL_CTX_set_cipher_list(ssl_ctx, "ALL:!aNULL:!eNULL:@STRENGTH") == 0)
```

10. Build Open5GS to apply these modifications:

```
meson build --prefix=`pwd`/install \
  -Dssl=true \
  -Dlibdir=lib \
  -Ddefault_library=shared \
  -Dcpp_args="-I/usr/local/openssl/include" \
  -Dc_args="-I/usr/local/openssl/include" \
  -Dcpp_link_args="-L/usr/local/openssl/lib -Wl,-rpath,/usr/local/openssl/lib" \
  -Dc_link_args="-L/usr/local/openssl/lib -Wl,-rpath,/usr/local/openssl/lib" \
  -Dpkg_config_path=/usr/local/openssl/lib/pkgconfig
ninja -C build
sudo ninja -C build install
```

11. Modify YAML files:

Enable TLS in the NF's configuration file

```
- plmn_id:
  mcc: 999
  mnc: 70
@@ -16,4 +16,4 @@
- sbi:
-   server:
-     address: 127.0.0.10
-     port: 7777
+## sbi:
+##   server:
+##     address: 127.0.0.10
+##     port: 7777

#####
# HTTPS scheme with TLS
#####
# o Set as default if not individually set
@@ -54,12 +54,12 @@
-# default:
```

```

-#   tls:
-#       server:
-#           scheme: https
-#           private_key: @sysconfdir@/open5gs/tls/nrf.key
-#           cert: @sysconfdir@/open5gs/tls/nrf.crt
-#       client:
-#           scheme: https
-#           cacert: @sysconfdir@/open5gs/tls/ca.crt
-#   sbi:
-#       server:
-#           - address: nrf.localdomain
+ default:
+   tls:
+       server:
+           scheme: https
+           private_key: /home/open5gs/open5gs/install/etc/open5gs/tls/nrf.key
+           cert: /home/open5gs/open5gs/install/etc/open5gs/tls/nrf.crt
+       client:
+           scheme: https
+           cacert: /home/open5gs/open5gs/install/etc/open5gs/tls/ca.crt
+   sbi:
+       server:
+           - address: nrf.localdomain

```

Make sure to edit /etc/hosts to declare the IP address of every localdomain:

```

sudo nano /etc/hosts
# Open5GS Network Functions
192.168.2.197 nrf.localdomain
127.0.0.11 ausf.localdomain
127.0.0.12 udm.localdomain
127.0.0.13 pcf.localdomain
127.0.0.14 nssf.localdomain
127.0.0.15 bsf.localdomain
127.0.0.20 udr.localdomain
127.0.0.200 scp.localdomain
127.0.0.4 smf.localdomain
192.168.2.197 amf.localdomain

```

11. Restart Services and Validate Configuration

Initiate the Open5GS NRF component and verify the enabled TLS version:

```
sudo ~/open5gs/install/bin/open5gs-nrfd
```

Perform connectivity tests to confirm TLS 1.0 support:

```
openssl s_client -connect nrf.localdomain:443 -tls1
curl -v --tlsv1.0 --insecure https://nrf.localdomain
```

Perform TLS Testing:

```
java -jar TLS-Server-Scanner.jar -connect 192.168.2.197:443
./testssl.sh 192.168.2.197:443
```

Output from TLS-Scanner which shows TLS1 support:

```
INFO : Main - Scanned in: 32933s
Report for 192.168.2.197:443

-----
Versions

DTLS 1.0           : could not test
DTLS 1.2           : could not test
SSL 2.0            : false
SSL 3.0            : false
TLS 1.0            : true
TLS 1.1            : true
TLS 1.2            : true
TLS 1.3            : true
```

Figure 1: TLS-Scanner TLS Support

And the Vulnerabilities:

Attack Vulnerabilities

```

Padding Oracle : vulnerable
Bleichenbacher : not vulnerable
Raccoon : not vulnerable
Direct Raccoon : could not test (not vulnerable)
CRIME : not vulnerable
Breach : not vulnerable
Invalid Curve : not vulnerable
Invalid Curve (ephemeral) : not vulnerable
Invalid Curve (twist) : not vulnerable
SSL Poodle : not vulnerable
Logjam : not vulnerable
Sweet 32 : vulnerable
General DROWN : could not test (not vulnerable)
Extra Clear DROWN : could not test (not vulnerable)
Heartbleed : could not test (not vulnerable)
EarlyCcs : not vulnerable
ALPACA : not mitigated
Renegotiation Attack (ext)
-1.hs without ext, 2.hs with ext : not vulnerable
-1.hs with ext, 2.hs without ext : not vulnerable
Renegotiation Attack (cs)
-1.hs without cs, 2.hs with cs : not vulnerable
-1.hs with cs, 2.hs without cs : not vulnerable

```

Figure 2: TLS-Scanner Attack Vulnerabilities

```

Testing vulnerabilities
Heartbleed (CVE-2014-0160) not vulnerable (OK), no heartbeat extension
CCS (CVE-2014-0224) not vulnerable (OK)
Ticketbleed (CVE-2016-9244), experiment. (applicable only for HTTPS)
ROBOT not vulnerable (OK)
Secure Renegotiation (RFC 5746) supported (OK)
Secure Client-Initiated Renegotiation not vulnerable (OK)
CRIME, TLS (CVE-2012-4929) not vulnerable (OK) (not using HTTP anyway)
POODLE, SSL (CVE-2014-3566) not vulnerable (OK), no SSLv3 support
TLS_FALLBACK_SCSV (RFC 7507) Downgrade attack prevention supported (OK)
SWEET32 (CVE-2016-2183, CVE-2016-6329) VULNERABLE, uses 64 bit block ciphers
FREAK (CVE-2015-0204) not vulnerable (OK)
DROWN (CVE-2016-0800, CVE-2016-0703) not vulnerable on this host and port (OK)
5E280DBAD771FE64322F29A35FA7 make sure you don't use this certificate elsewhere with SSLv2 enabled services, see
LOGJAM (CVE-2015-4000), experimental https://search.censys.io/search?resource=hosts&virtual_hosts=INCLUDE&q=0ECAEDC75E171554557194938C3967546C18
BEAST (CVE-2011-3389) not vulnerable (OK): no DH EXPORT ciphers, no DH key detected with <= TLS 1.2
TLS1: ECDHE-RSA-AES256-SHA AES256-SHA CAMELLIA256-SHA ECDHE-RSA-AES128-SHA AES128-SHA SEED-SHA
CAMELLIA128-SHA IDEA-CBC-SHA
LUCKY13 (CVE-2013-0169), experimental VULNERABLE -- but also supports higher protocols TLSv1.1 TLSv1.2 (likely mitigated)
Winshock (CVE-2014-6321), experimental potentially VULNERABLE, uses cipher block chaining (CBC) ciphers with TLS. Check patches
RC4 (CVE-2013-2566, CVE-2015-2808) not vulnerable (OK)
no RC4 ciphers detected (OK)

```

Figure 3: TestSSL Attack Vulnerabilities

Observations and Resolutions

- **Open5GS Defaulting to OpenSSL 1.1.1f:** Resolved by explicitly defining the OpenSSL build path during compilation and deleting previous OpenSSL version.
- **Cipher Suites Not Recognized:** Addressed through adjustments to nghttp2-server.c.
- **TLS-Scanner Detection Issues:** Required complete recompilation of Open5GS following modifications.

Conclusion

This procedure successfully downgrades Open5GS to support TLS 1.0, introducing security vulnerabilities relevant for penetration testing and cryptographic analysis. This configuration should never be used in production environments due to its exposure to known attacks such as Padding Oracle, BEAST, SWEET32, LUCKY13, and downgrade-based exploits.