

Forum:[Hack Forums](#)

Thread: [A comprehensive guide for setting up Ss7 attacks](#)

Clumsy - September 29, 2024, 1:22 am

Introduction

Hello HackForums! My name is Clumsy, and today we're diving into the complex world of SS7 signaling systems. Whether you're a security researcher, a telecom enthusiast, or someone curious about mobile networks, this talk is for you. We'll walk through setting up an SS7 signaling system from scratch on a Debian server, covering components like SS7Mapper, SIGTRAN, GNU Radio, and TTCN-3 for testing SS7 attacks and defenses.

Agenda

- Overview of SS7 and Its Importance
- Setting Up the Environment
- Installing and Configuring Key Components
- Testing and Validation

Overview of SS7 and Its Importance

SS7, or Signaling System No. 7, is a set of protocols used by telecommunication networks to manage call setup, routing, and teardown. While critical for network functionality, SS7 is also a significant target for attacks due to its vulnerabilities. Understanding and testing SS7 can enhance network security and resilience.

2. Setting Up the Environment

We need to prepare our Debian server by updating the system and installing necessary dependencies. Here's what this involves:

System Update and Upgrade: Ensure the server is up-to-date with the latest security patches and software versions.

Installing Dependencies: Install libraries and tools like git, gcc, make, and others.

Environment Configuration: Set up environment variables for tools like GNU Radio.

Installing and Configuring Key Components

Now we'll install and configure the essential components for our SS7 setup, which includes:

Osmocom Libraries and Tools: The foundation for our SS7 stack:

- libosmocore
- libosmo-sccp
- osmo-iuh
- osmo-hlr
- osmo-mgw
- osmo-bsc
- osmo-msc
- osmo-ggsn
- osmo-sgsn

SS7Mapper: A tool for mapping and analyzing SS7 networks.

SIGTRAN: For IP-based signaling for SS7.

GNU Radio: A software development toolkit for signal processing.

TTCN-3: A testing technology for telecom systems.

Each component will be cloned, built, and installed, ensuring correct configuration and seamless handling of dependencies.

4. Testing and Validation

With everything set up, it's time to test our SS7 signaling system:

Running Basic Tests: Ensure each component is functioning correctly.

Setting Up Test Scenarios: Use TTCN-3 to simulate various attack and defense scenarios.

Validating Configuration: Confirm the system behaves as expected under different conditions.

This step is crucial for ensuring our setup is functional, secure, and robust against potential attacks.

Conclusion

Setting up an SS7 signaling system is a complex task, but with the right guidance and tools, it's achievable. This setup provides a solid foundation for further research and improving the security of telecommunications networks.

Thank you for attending this talk! Let's work together to enhance network security, one setup at a time!

Automated Setup Script

Below is an automated script to help you set this up on Debian.

Code `#!/bin/bash`

`# Update and upgrade the system`

`sudo apt update && sudo apt upgrade -y`

`# Install required dependencies`

`sudo apt install -y \`

`git build-essential gcc g++ make automake autoconf libtool pkg-config \`
`libtalloc-dev libpcsc-lite-dev libortp-dev libsctp-dev libssl-dev libdbi-dev \`
`libdbd-sqlite3 libsqlite3-dev libpcap-dev libc-ares-dev libgnutls28-dev \`
`libsctp-dev sqlite3 libusb-1.0-0-dev libmnl-dev libsofia-sip-ua-glib-dev \`
`libxml2-dev libncurses5-dev libncursesw5-dev libsctp-dev gnuradio \`
`python3-pip python3-venv cmake libfftw3-dev libcppunit-dev swig doxygen \`
`liblog4cpp5-dev libzmq3-dev`

`# Install TTCN-3 dependencies`

`sudo apt install -y \`

`eclipse-titan titan.core titan.ttcn3 titan.eclipse titan.simulators titan.protocolmodules.SCCP`

`# Function to clone, build, and install a component`

`install_component() {`

`git clone "$1"`

`cd "$(basename "$1" .git)" || exit`

`autoreconf -fi`

`./configure`

`make -j"$(nproc)"`

`sudo make install`

`sudo ldconfig`

`cd ..`

`}`

```
# Clone and build Osmocom components
install_component https://git.osmocom.org/libosmocore.git
install_component https://git.osmocom.org/libosmo-sccp.git
install_component https://git.osmocom.org/osmo-iuh.git
install_component https://git.osmocom.org/osmo-hlr.git
install_component https://git.osmocom.org/osmo-mgw.git
install_component https://git.osmocom.org/osmo-bsc.git
install_component https://git.osmocom.org/osmo-msc.git
install_component https://git.osmocom.org/osmo-ggsn.git
install_component https://git.osmocom.org/osmo-sgsn.git
```

```
# Clone and build additional tools
install_component https://github.com/ernw/ss7MAPer.git
install_component https://github.com/RestComm/sigtran.git
```

```
# Clone and build GNU Radio
git clone https://github.com/gnuradio/gnuradio.git
cd gnuradio || exit
mkdir build
cd build || exit
cmake ..
make -j"$(nproc)"
sudo make install
sudo ldconfig
cd ../../
```

```
# Add GNU Radio environment variables
echo "export PYTHONPATH=/usr/local/lib/python3/dist-packages:$PYTHONPATH" >> ~/.bashrc
echo "export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH" >> ~/.bashrc
source ~/.bashrc
```

```
# Clean up sudoers file
echo "$USER ALL= NOPASSWD: /sbin/ldconfig" | sudo tee /etc/sudoers.d/ldconfig >/dev/null
sudo rm /etc/sudoers.d/ldconfig
```

```
# Clean up the build directories
rm -rf libosmocore libosmo-sccp osmo-iuh osmo-hlr osmo-mgw osmo-bsc osmo-msc osmo-ggsn
osmo-sgsn ss7MAPer sigtran gnuradio
```

```
echo "SS7 signaling system setup is complete."
```

```
# References
echo "References:"
echo "Osmocom Project: https://osmocom.org"
echo "SS7Mapper: https://github.com/ernw/ss7MAPer"
echo "GNU Radio: https://github.com/gnuradio/gnuradio"
echo "TTCN-3: https://www.eclipse.org/titan/"
echo "SIGTRAN: https://github.com/RestComm/sigtran"
References
```

Osmocom Project: <https://osmocom.org> | <https://downloads.osmocom.org> | <https://gitea.osmocom.org/explore/repos>

SS7Mapper: <https://github.com/ernw/ss7MAPer>

GNU Radio: <https://github.com/gnuradio/gnuradio>

TTCN-3: <https://www.eclipse.org/titan/>

SIGTRAN: <https://github.com/RestComm/sigtran>

https://www.youtube.com/watch?v=T_Qx7YmhSg4

1. Make sure you have the latest version of libosmocore:

If you don't have the library:

```
$ git clone https://gitea.osmocom.org/osmocom/libosmocore
```

```
$ cd libosmocore/
```

If you do, but need to update:

```
$ cd libosmocore/
```

```
$ sudo make uninstall
```

```
$ make distclean
```

```
$ git pull --rebase
```

Then:

```
$ autoreconf -f -i
```

```
$ ./configure
```

```
$ make
```

```
$ sudo make install
```

2. Install libosmo-dsp:

```
$ git clone https://gitea.osmocom.org/sdr/libosmo-dsp
```

```
$ cd libosmo-dsp/
```

```
$ autoreconf -i
```

```
$ ./configure
```

```
$ make
```

```
$ sudo make install
```

Firmware and host software

1. Clone the 'fixeria/trx' branch:

```
$ git clone https://gitea.osmocom.org/phone-side/osmocom-bb -b fixeria/trx
```

```
$ cd trx/src/
```

2. Enable TX support in firmware:

Uncomment 'CFLAGS += -DCONFIG_TX_ENABLE' in 'target/firmware/Makefile'.

3. Compile OsmocomBB with transceiver support:

```
$ make HOST_layer23_CONFARGS=--enable-transceiver
```

At this step your transceiver should be ready.

Clock source

Time-division (TDMA) systems require very accurate counting of the time segments (frames and

timeslots) and when they start and stop. Also, multiple base stations within a network must be time-synchronized between themselves to make subscribers able to perform handover operation (switching from one BTS to another). If the towers clocking were out of sync, then communications would falter as each node would be trying to deal with segments that were slightly offset and this would introduce errors. This is why a good clock source is required to operate a base station.

The satellite systems, such as GPS and Iridium, do have pretty accurate clock that is being distributed in addition to the prime services (location, calls, etc). They could be used (and often used in commercial networks) for synchronization purposes. This approach would require one to have a corresponding receiver and a direct sky view, but there is a simpler way. We can use the SCH clock indications from existing base stations, eventually a phone is designed to do that!

So, use RSSI or cell_log applications to find cells with good signal and remember the ARFCN numbers. One of them will be used as a clock source for our own base station.

Usage

1. First, you need to run the transceiver firmware:

```
$ cd trx/src/  
$ host/osmocon/osmocon -m c123xor -p /dev/ttyUSB0 -c  
target/firmware/board/compal_e88/trx.higham.bin
```

Additionally, if you have two (or more) phones, load the firmware to a second phone (optional):

```
$ cd trx/src/  
$ host/osmocon/osmocon -m c123xor -p /dev/ttyUSB1 -s /tmp/osmocomb_l2.2 -c  
target/firmware/board/compal_e88/trx.higham.bin
```

2. The host application could be found in 'src/host/layer23/transceiver/':

```
$ cd trx/src/host/layer23/src/transceiver/  
$ ./transceiver -h
```

Usage: ./transceiver -a arfcn_sync

Some useful options:

- h --help this text
- d --debug MASK Enable debugging (e.g. -d DL1C:DTRX)
- e --log-level LOGL Set log level (1=debug, 3=info, 5=notice)
- D --daemonize For the process into a background daemon
- s --disable-color Don't use colors in stderr log output
- a --arfcn-sync ARFCN Set ARFCN to sync to
- p --arfcn-sync-pcs The ARFCN above is PCS
- 2 --second-phone Use second phone for TS 1
- r --realtime PRIO Set realtime scheduler with given prio

The --arfcn-sync or --arfcn-sync-pcs are used to specify the ARFCN of a clock source cell. The high priority scheduling is recommended for low-performance PCs.

3. Let's try to synchronize with a BTS:

If you have two (or more) phones, add '-2' option.

```
$ ./transceiver -a <ARFCN>
```

You should see something like this:

```
<0012> l1ctl.c:383 Reset received: Starting sync.  
<0012> l1ctl.c:338 Sync acquired, setting BTS mode ...
```

```
<0011> trx.c:194 TRX CLK Indication 1255520
<0011> trx.c:194 TRX CLK Indication 1255571
<0011> trx.c:194 TRX CLK Indication 1255622
<0011> trx.c:194 TRX CLK Indication 1255673
<0011> trx.c:194 TRX CLK Indication 1255724
<0011> trx.c:194 TRX CLK Indication 1255775
<0011> trx.c:194 TRX CLK Indication 1255826
<0011> trx.c:194 TRX CLK Indication 1255877
<0011> trx.c:194 TRX CLK Indication 1255928
<0011> trx.c:194 TRX CLK Indication 1255979
<0011> trx.c:194 TRX CLK Indication 1256030
<0011> trx.c:194 TRX CLK Indication 1256081
```

...

If something goes wrong, find another ARFCN and try again.
As soon as you get the transceiver synchronized, just keep it running.

CalypsoBTS with OsmoNITB

The simplest way to setup a GSM network is to use OsmoBTS with OpenBSC in OsmoNITB mode. OsmoNITB is a simple core network implementation - network in the box. It emulates basic core elements like MSC, HLR, VLR, etc. OsmoBTS is a part of Osmocom GSM core network implementation, that supports a variety of different hardware back-ends, including TRX Interface transceivers like OsmoTRX. Unlike OpenBTS, this software stack is well maintained, so I would recommend you to choose it.

Installation

This guide is not about building the core network software, so please follow Building OpenBSC. Feel free to browse the wiki and check out the official documentation <http://ftp.osmocom.org/docs/>.

You will need to build / install the following components:

openbsc (legacy Network in the Box)

osmo-bts (with --enable-trx flag)

Configuration

Almost each program has the configuration examples. Just look for 'doc/examples/'. I recommend you to customize them as you need and put the updated files in a separate directory. Let's use the '~/.osmocom/' for such purposes:

```
$ mkdir ~/.osmocom/
```

You can use the following configuration examples:

OsmoNITB: 'doc/examples/osmo-nitb/sysmobts/openbsc.cfg'

OsmoBTS: 'doc/examples/calypso/osmo-bts.cfg'

There are some important things you need to pay attention.

The recommended values are listed below.

OsmoNITB configuration

Network identification

network

...

network country code 901

mobile network code 70

short name OpenBSC

long name OpenBSC

Network access

Warning! Setting the access policy to 'accept-all' may be dangerous. It's recommended to use the 'closed' policy, adding your test subscriber(s) info to the HLR database.

```
network
```

```
...
```

```
auth policy (closed | accept-all | regexp | token)
```

```
authorized-regexp .*
```

Encryption

To be able to use any encryption algorithm, you need to know the Ki of subscribers. So, it's recommended not to use any encryption (A5/0) for testing.

```
network
```

```
...
```

```
encryption a5 (0 | 1 | 2 | 3)
```

Channel configuration

Set a proper ARFCN for running BTS (don't confuse with a clock source). Also, set a corresponding band name.

```
network
```

```
...
```

```
bts 0
```

```
...
```

```
band (GSM900 | DCS1800)
```

```
...
```

```
trx 0
```

```
...
```

```
arfcn ARFCN
```

MS power control

If the normal phones are only one or few meters away, the maximal power should be set to 0. In case of long distance test it can be set to 30 (DCS) or 33 (GSM 900).

```
network
```

```
...
```

```
bts 0
```

```
...
```

```
ms max power 0
```

Logical channel mapping

I recommend you to use the combined channel configuration (CCCH+SDCCH4) for TS0, because CalypsoBTS at the moment cannot serve all timeslots.

```
network
```

```
...
```

```
bts 0
```

```
...
```

```
trx 0
```

```
...
```

```
timeslot 0
```

```
phys_chan_config CCCH+SDCCH4
```

```
hopping enabled 0
```

If you have two or more Calypso based phones, it's also possible to serve the second timeslot (TS1). For example, 'TCH/H' channel could be used to serve one voice call between two subscribers.

network

```
...
bts 0
...
trx 0
...
timeslot 1
  phys_chan_config TCH/H
  hopping enabled 0
```

Subscriber management policy

Of course, you can manually edit HLR database and configure as much subscribers as you need. But the simplest way is to create subscriber records on fly, during the Location Update Request.

network

```
...
bts 0
...
trx 0
...
nitb
  subscriber-create-on-demand
  assign-tmsi
```

OsmoBTS configuration

GSM band configuration

The band value should match one in the OsmoNITB configuration.

bts 0

```
...
band (900 | 1800)
```

A-bis link configuration

In the most cases both OsmoNITB and OsmoBTS are working on the same host. So, the loopback address should be used for 'remote-ip'. The 'unit-id' should be the same as in the OsmoNITB configuration.

bts 0

```
...
oml remote-ip 127.0.0.1
ipa unit-id 1801 0
```

Transceiver configuration

In order to prevent the input saturation, 'ms-power-loop' should be set to -65.

phy 0

```
...
osmotrx ms-power-loop -65
```

Since the CalypsoBTS transceiver is obsolete, it still requires BSIC to be set instead of TSC.

phy 0

```
...
osmotrx legacy-setbsic
```

Running

Tip: feel free to use tmux or screen to avoid a mess with multiple windows

0. Make sure your transceiver is still running...

1. Start OsmoNITB:

```
$ osmo-nitb -c ~/.osmocom/openbsc.cfg -l ~/.osmocom/hlr.sqlite3 -P -C  
--debug=DRLL:DCC:DMM:DRR:DRSL:DNM
```

2. Start OsmoBTS:

```
$ osmo-bts-trx -c ~/.osmocom/osmo-bts.cfg
```

Congratulations! Now you should get the network running. You can use telnet interface to control both OsmoNITB and OsmoBTS:

```
# OsmoNITB telnet interface  
$ telnet localhost 4242
```

```
# OsmoBTS telnet interface  
$ telnet localhost 4241
```

You should hopefully see the BTS by performing a manual network search with your phone. Monitor the output of osmocon and the transceiver to see if all goes well. If anything should fail, reboot the phone and start over.

CalypsoBTS with OpenBTS

OpenBTS is another open source software project aimed to replace legacy telecommunication protocols and traditionally complex, proprietary hardware systems by IP a flexible software architecture. It implements the BTS side protocol stack and also some important core network elements.

Installation and configuration

Follow this howto in the project wiki.

By default, the stock transceiver is used by OpenBTS. It's designed to work on SDR based hardware (like USRP), but this is not our case. So, we need to replace it by OsmocomBB transceiver.

1. Replace the stock transceiver

Make sure that OpenBTS is not running. In the folder where the OpenBTS executable resides, create a script with the filename 'transceiver' with the following content:

```
#!/bin/bash  
exec <your path to osmocom-bb>/src/host/layer23/src/transceiver/transceiver -a <ARFCN>  
Where ARFCN is the channel of clock source cell. If you have two (or more phones), add '-2' option.
```

2. And make it executable:

```
sudo chmod +x ./transceiver
```

Running

Run the TRX firmware on the phone as described above. No need to start transceiver, OpenBTS will run it automatically.

The OpenBTS CLI allows you to monitor system status and change many operating parameters of OpenBTS and the Transceiver in real time.

Demo

<https://www.youtube.com/watch?v=Aj1A6F-FAGU>

<https://www.youtube.com/watch?v=FifvFov3Rsl>

<https://www.youtube.com/watch?v=PZ4-UCH2ED8>

References

<https://www.youtube.com/watch?v=xFjVcxMpA6c>

https://events.ccc.de/congress/2012/Fahrplan/attachments/2244_29c3_further_hacks_on_the_calypso_platform

Known limitations and issues

Two timeslots limitation. Due to some DSP limitations and the non-BTS nature of CalypsoBTS, one phone can serve only one timeslot. Moreover, despite the current transceiver code was extended to support multiple phones ('jolly/testing' branch), this feature is still incomplete. So, for now, up to two phones could be used to serve up to two physical timeslots.

Non-continuous C0 transmission. According to the GSM specifications, a C0 base station should keep transmission on all timeslots even if there is nothing to transmit. This makes the mobile stations able to detect them and measure the power levels. Again, a phone isn't BTS. It takes some time to tune between DL and UL frequencies.

<https://github.com/RangeNetworks/openbts>

And now a script to fully configure this

Code `#!/bin/bash`

`# Define paths`

`INSTALL_DIR="$HOME/SS7Setup"`

`OSMOCOM_DIR="$INSTALL_DIR/osmocom"`

`TTCN3_DIR="$INSTALL_DIR/ttcn3"`

`# Create installation directory`

`mkdir -p "$INSTALL_DIR"`

`cd "$INSTALL_DIR" || exit`

`# Update and upgrade (not applicable in MinGW, but you can manually ensure everything is updated)`

`# Install required dependencies (manually install if not using package manager)`

`# Note: This script does not use pacman or wget.`

`echo "Please ensure the following dependencies are installed:"`

`echo "1. Git"`

`echo "2. Make"`

`echo "3. GCC"`

`echo "4. CMake"`

`echo "5. GNU Radio (if applicable)"`

`echo "6. Any other necessary libraries/tools."`

`# Clone and build Osmocom components`

`git clone https://git.osmocom.org/libosmocore.git "$OSMOCOM_DIR/libosmocore"`

`cd "$OSMOCOM_DIR/libosmocore" || exit`

`autoreconf -fi`

```
./configure --prefix="$INSTALL_DIR"
make -j"$(nproc)"
make install
```

```
# Clone and build other Osmocom components
for repo in libosmo-sccp osmo-iuh osmo-hlr osmo-mgw osmo-bsc osmo-msc osmo-ggsn
osmo-sgsn; do
    git clone "https://git.osmocom.org/$repo.git" "$OSMOCOM_DIR/$repo"
    cd "$OSMOCOM_DIR/$repo" || exit
    autoreconf -fi
    ./configure --prefix="$INSTALL_DIR"
    make -j"$(nproc)"
    make install
    cd "$OSMOCOM_DIR" || exit
done
```

```
# Clone and set up SS7Mapper
git clone https://github.com/ernw/ss7MAPer.git "$INSTALL_DIR/ss7MAPer"
cd "$INSTALL_DIR/ss7MAPer" || exit
pip install -r requirements.txt
```

```
# Clone and build SIGTRAN
git clone https://github.com/RestComm/sigtran.git "$INSTALL_DIR/sigtran"
cd "$INSTALL_DIR/sigtran" || exit
./build.sh
./install.sh
```

```
# Clone and build GNU Radio if applicable
# Uncomment and adjust if you want to include GNU Radio setup
# git clone https://github.com/gnuradio/gnuradio.git "$INSTALL_DIR/gnuradio"
# cd "$INSTALL_DIR/gnuradio" || exit
# mkdir build
# cd build || exit
# cmake ..
# make -j"$(nproc)"
# make install
```

```
# Clean up and finish
echo "SS7 signaling system setup is complete."
echo "Make sure to add the installation directories to your PATH environment variable."
Code #!/bin/bash
```

```
# Update and upgrade the system
sudo apt update && sudo apt upgrade -y
```

```
# Install required dependencies
sudo apt install -y \
git build-essential gcc g++ make automake autoconf libtool pkg-config \
libtalloc-dev libpcsclite-dev libortp-dev libsctp-dev libssl-dev libdbi-dev \
libdbd-sqlite3 libsqlite3-dev libpcap-dev libc-ares-dev libgnutls28-dev \
libsctp-dev sqlite3 libusb-1.0-0-dev libmnl-dev libsofia-sip-ua-glib-dev \
libxml2-dev libncurses5-dev libncursesw5-dev libsctp-dev gnuradio \
python3-pip python3-venv cmake libfftw3-dev libcppunit-dev swig doxygen \
```

```
liblog4cpp5-dev libzmq3-dev
```

```
# Install TTCN-3 dependencies
```

```
sudo apt install -y \
```

```
eclipse-titan titan.core titan.ttcn3 titan.eclipse titan.simulators titan.protocolmodules.SCCP
```

```
# Function to clone, build, and install a component
```

```
install_component() {  
    git clone "$1"  
    cd "$(basename "$1" .git)" || exit  
    autoreconf -fi  
    ./configure  
    make -j"$(nproc)"  
    sudo make install  
    sudo ldconfig  
    cd ..  
}
```

```
# Clone and build Osmocom components
```

```
install_component https://git.osmocom.org/libosmocore.git  
install_component https://git.osmocom.org/libosmo-sccp.git  
install_component https://git.osmocom.org/osmo-iuh.git  
install_component https://git.osmocom.org/osmo-hlr.git  
install_component https://git.osmocom.org/osmo-mgw.git  
install_component https://git.osmocom.org/osmo-bsc.git  
install_component https://git.osmocom.org/osmo-msc.git  
install_component https://git.osmocom.org/osmo-ggsn.git  
install_component https://git.osmocom.org/osmo-sgsn.git
```

```
# Clone and build additional tools
```

```
install_component https://github.com/ernw/ss7MAPer.git  
install_component https://github.com/RestComm/sigtran.git
```

```
# Clone and build GNU Radio
```

```
git clone https://github.com/gnuradio/gnuradio.git  
cd gnuradio || exit  
mkdir build  
cd build || exit  
cmake ..  
make -j"$(nproc)"  
sudo make install  
sudo ldconfig  
cd ../..
```

```
# Add GNU Radio environment variables
```

```
echo "export PYTHONPATH=/usr/local/lib/python3/dist-packages:$PYTHONPATH" >> ~/.bashrc  
echo "export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH" >> ~/.bashrc  
source ~/.bashrc
```

```
# Clean up sudoers file
```

```
echo "$USER ALL= NOPASSWD: /sbin/ldconfig" | sudo tee /etc/sudoers.d/ldconfig >/dev/null  
sudo rm /etc/sudoers.d/ldconfig
```

```
# Clean up the build directories
rm -rf libosmocore libosmo-sccp osmo-iuh osmo-hlr osmo-mgw osmo-bsc osmo-msc osmo-ggsn
osmo-sgsn ss7MAPer sigtran gnuradio
```

```
echo "SS7 signaling system setup is complete."
```

```
# References
```

```
echo "References:"
echo "Osmocom Project: https://osmocom.org"
echo "SS7Mapper: https://github.com/ernw/ss7MAPer"
echo "GNU Radio: https://github.com/gnuradio/gnuradio"
echo "TTCN-3: https://www.eclipse.org/titan/"
echo "SIGTRAN: https://github.com/RestComm/sigtran"
some more things you will need
```

```
git clone https://git.osmocom.org/%G.git
```

```
git clone https://git.osmocom.org/libosmocore.git
```

```
git clone https://github.com/RestComm/sigtran.git
```

```
git clone https://github.com/ernw/ss7MAPer.git
```

```
git clone https://git.osmocom.org/libosmocore.git
```

```
git clone https://github.com/gnuradio/gnuradio
```

```
git clone https://github.com/ernw/ss7MAPer
```

```
git clone https://git.osmocom.org/libosmocore.git
```

```
git clone https://git.osmocom.org/libosmo-sccp.git
```

```
git clone https://git.osmocom.org/osmo-iuh.git
```

```
git clone https://git.osmocom.org/osmo-hlr.git
```

```
git clone https://git.osmocom.org/osmo-mgw.git
```

```
git clone git clone https://git.osmocom.org/osmo-bsc.git
```

```
git clone https://git.osmocom.org/osmo-msc.git
```

```
git clone https://git.osmocom.org/osmo-ggsn.git
```

```
git clone https://git.osmocom.org/osmo-sgsn.git
```

```
https://www.eclipse.org/titan/
```

m0narch - October 2, 2024, 1:11 am

(Sep 29, 2024 01:22 AM)Clumsy Wrote: [\[CITE\]](#) some more things you will need

```
git clone https://git.osmocom.org/%G.git
```

git clone <https://git.osmocom.org/libosmocore.git>

git clone <https://github.com/RestComm/sigtran.git>

git clone <https://github.com/ernw/ss7MAPer.git>

git clone <https://git.osmocom.org/libosmocore.git>

git clone <https://github.com/gnuradio/gnuradio>

git clone <https://github.com/ernw/ss7MAPer>

git clone <https://git.osmocom.org/libosmocore.git>

git clone <https://git.osmocom.org/libosmo-sccp.git>

git clone <https://git.osmocom.org/osmo-iuh.git>

git clone <https://git.osmocom.org/osmo-hlr.git>

git clone <https://git.osmocom.org/osmo-mgw.git>

git clone git clone <https://git.osmocom.org/osmo-bsc.git>

git clone <https://git.osmocom.org/osmo-msc.git>

git clone <https://git.osmocom.org/osmo-ggsn.git>

git clone <https://git.osmocom.org/osmo-sgsn.git>

<https://www.eclipse.org/titan/>

The first git clone link is invalid, any chance you could fix that?

Unknown crackhead - October 3, 2024, 4:59 pm

thank u soo much for helping us