

Forum:[Hack Forums](#)

Thread: [Setting Up Your Own SS7 Signaling System NO .7 Test A Comprehensive Guide](#)

Clumsy - July 13, 2024, 11:59 am

Hello DEF CON! My name is Taylor Christian Newsome, and today, we're diving into the fascinating and complex world of SS7 signaling systems. Whether you're a security researcher, a telecom enthusiast, or someone looking to learn about the inner workings of mobile networks, this talk is for you. We're going to walk through setting up an SS7 signaling system from scratch on a Debian server, including 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

Q&A

1. Overview of SS7 and Its Importance

SS7, or Signaling System No. 7, is the set of protocols used by telecommunication networks to exchange information and manage the setup, routing, and teardown of calls and text messages. It's crucial for network functionality but also a significant target for attacks due to its vulnerabilities. Understanding and testing SS7 can help improve network security and resilience.

Title: "Setting Up Your Own SS7 Signaling System: A Comprehensive Guide"

Introduction:

Hello DEF CON! My name is [Your Name], and today, we're diving into the fascinating and complex world of SS7 signaling systems. Whether you're a security researcher, a telecom enthusiast, or someone looking to learn about the inner workings of mobile networks, this talk is for you. We're going to walk through setting up an SS7 signaling system from scratch on a Debian server, including 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

Q&A

1. Overview of SS7 and Its Importance

SS7, or Signaling System No. 7, is the set of protocols used by telecommunication networks to exchange information and manage the setup, routing, and teardown of calls and text messages. It's crucial for network functionality but also a significant target for attacks due to its vulnerabilities. Understanding and testing SS7 can help improve network security and resilience.

2. Setting Up the Environment

First, we need to prepare our Debian server. This includes updating the system, installing necessary dependencies, and setting up a working environment for our components.

Here's a sneak peek of what this involves

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

Installing Dependencies: This includes libraries and tools like git, gcc, make, automake, autoconf, and various development libraries.

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

3. Installing and Configuring Key Components:

Now, let's get into the nitty-gritty of installing and configuring the components necessary for our SS7 setup. We'll be working with several critical tools:

Osmocom Libraries and Tools: These provide a foundation for our SS7 stack.

libosmocore

libosmo-sccp

osmo-juh

osmo-hlr

osmo-mgw

osmo-bsc

osmo-msc

osmo-ggsn

osmo-sgsn

SS7Mapper: A powerful tool for mapping and analyzing SS7 networks.

SIGTRAN: Provides IP-based signaling for SS7.

GNU Radio: A software development toolkit that provides signal processing blocks to implement software radios.

TTCN-3: A testing technology for telecommunication systems.

Each of these components will be cloned, built, and installed on our server. We'll ensure that they're configured correctly and that any dependencies are handled seamlessly.

Title: "Setting Up Your Own SS7 Signaling System: A Comprehensive Guide"

Introduction:

Hello DEF CON! My name is [Your Name], and today, we're diving into the fascinating and complex world of SS7 signaling systems. Whether you're a security researcher, a telecom enthusiast, or someone looking to learn about the inner workings of mobile networks, this talk is for you. We're going to walk through setting up an SS7 signaling system from scratch on a Debian server, including 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

Q&A

1. Overview of SS7 and Its Importance:

SS7, or Signaling System No. 7, is the set of protocols used by telecommunication networks to exchange information and manage the setup, routing, and teardown of calls and text messages. It's crucial for network functionality but also a significant target for attacks due to its vulnerabilities. Understanding and testing SS7 can help improve network security and resilience.

2. Setting Up the Environment:

First, we need to prepare our Debian server. This includes updating the system, installing necessary dependencies, and setting up a working environment for our components.

Here's a sneak peek of what this involves:

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

Installing Dependencies: This includes libraries and tools like git, gcc, make, automake, autoconf, and various development libraries.

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

3. Installing and Configuring Key Components:

Now, let's get into the nitty-gritty of installing and configuring the components necessary for our SS7 setup. We'll be working with several critical tools:

Osmocom Libraries and Tools: These provide a foundation for our SS7 stack.

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

SS7Mapper: A powerful tool for mapping and analyzing SS7 networks.

SIGTRAN: Provides IP-based signaling for SS7.

GNU Radio: A software development toolkit that provides signal processing blocks to implement software radios.

TTCN-3: A testing technology for telecommunication systems.

Each of these components will be cloned, built, and installed on our server. We'll ensure that they're configured correctly and that any dependencies are handled seamlessly.

4. Testing and Validation

With everything set up, it's time to test our SS7 signaling system. This involves

Running Basic Tests: Ensuring each component is functioning correctly.

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

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

This step is crucial for ensuring that our setup is not only functional but also secure and robust against potential attacks.

Conclusion

Setting up an SS7 signaling system is no small feat, but with the right guidance and tools, it's entirely achievable. This setup provides a solid foundation for further research, testing, and improving the security of telecommunication networks.

Thank you for attending this talk. Let's make our networks more secure, one setup at a time!

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`

`# Clone and build libosmocore`

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

`cd libosmocore`

`autoreconf -fi`

`./configure`

`make -j$(nproc)`

`sudo make install`

`sudo ldconfig`

`cd ..`

`# Clone and build libosmo-sccp`

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

`cd libosmo-sccp`

`autoreconf -fi`

`./configure`

`make -j$(nproc)`

`sudo make install`

`sudo ldconfig`

`cd ..`

`# Clone and build osmo-iuh`

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

`cd osmo-iuh`

`autoreconf -fi`

`./configure`

`make -j$(nproc)`

```
sudo make install
sudo ldconfig
cd ..
```

```
# Clone and build osmo-hlr
git clone https://git.osmocom.org/osmo-hlr
cd osmo-hlr
autoreconf -fi
./configure
make -j$(nproc)
sudo make install
sudo ldconfig
cd ..
```

```
# Clone and build osmo-mgw
git clone https://git.osmocom.org/osmo-mgw
cd osmo-mgw
autoreconf -fi
./configure
make -j$(nproc)
sudo make install
sudo ldconfig
cd ..
```

```
# Clone and build osmo-bsc
git clone https://git.osmocom.org/osmo-bsc
cd osmo-bsc
autoreconf -fi
./configure
make -j$(nproc)
sudo make install
sudo ldconfig
cd ..
```

```
# Clone and build osmo-msc
git clone https://git.osmocom.org/osmo-msc
cd osmo-msc
autoreconf -fi
./configure
make -j$(nproc)
sudo make install
sudo ldconfig
cd ..
```

```
# Clone and build osmo-ggsn
git clone https://git.osmocom.org/osmo-ggsn
cd osmo-ggsn
autoreconf -fi
./configure
make -j$(nproc)
sudo make install
sudo ldconfig
cd ..
```

```
# Clone and build osmo-sgsn
git clone https://git.osmocom.org/osmo-sgsn
cd osmo-sgsn
autoreconf -fi
./configure
make -j$(nproc)
sudo make install
sudo ldconfig
cd ..
```

```
# Clone and build ss7mapper
git clone https://github.com/ernw/ss7MAPer.git
cd ss7MAPer
sudo pip3 install -r requirements.txt
cd ..
```

```
# Clone and build sigtran
git clone https://github.com/RestComm/sigtran.git
cd sigtran
./build.sh
sudo ./install.sh
cd ..
```

```
# Clone and build gnuradio
git clone https://github.com/gnuradio/gnuradio.git
cd gnuradio
mkdir build
cd build
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-juh osmo-hlr osmo-mgw osmo-bsc osmo-msc osmo-ggsn
osmo-sgsn ss7mapper sigtran gnuradio
```

```
echo "SS7 signaling system, SS7Mapper, SIGTRAN, GNU Radio, and TTCN-3 setup is complete."
```

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

Note I am not on twitter if you need to contact me you can do so via

<https://www.linkedin.com/in/clumsy/>

Ignore the hello defcon rofl I just copy pasted it from my thread over on their forums