HITBSECCONF2011
MALAYSIA
OCTOBER 10 - 13 @ INTERCONTINENTAL KL

ERNW
Living Security.

TODAY:

# HITB Labs: Practical Attacks Against 3G/4G Telecommunication Networks

**Jim Geovedi**

**Daniel Mende**

jim@geovedi.com, dmende@ernw.de

# Agenda

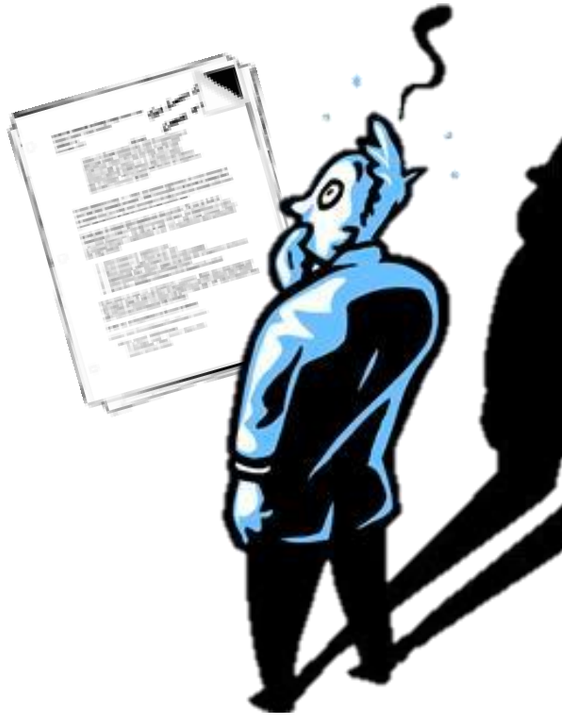
- **Overview 3G / 4G**
- **Backhaul Networks**
- **Backend Protocols in depth**
- **The Lab**
- **The Tools**
- **Exercises**
- **Conclusions**

# Overview 3G / 4G

# Standards

- **In mobile telco world everything standardized by *3GPP***
- **3GPP: collaboration between groups of telco standard orgs**
  - **Which “type of documents” do you think these guys produce? ;-)**

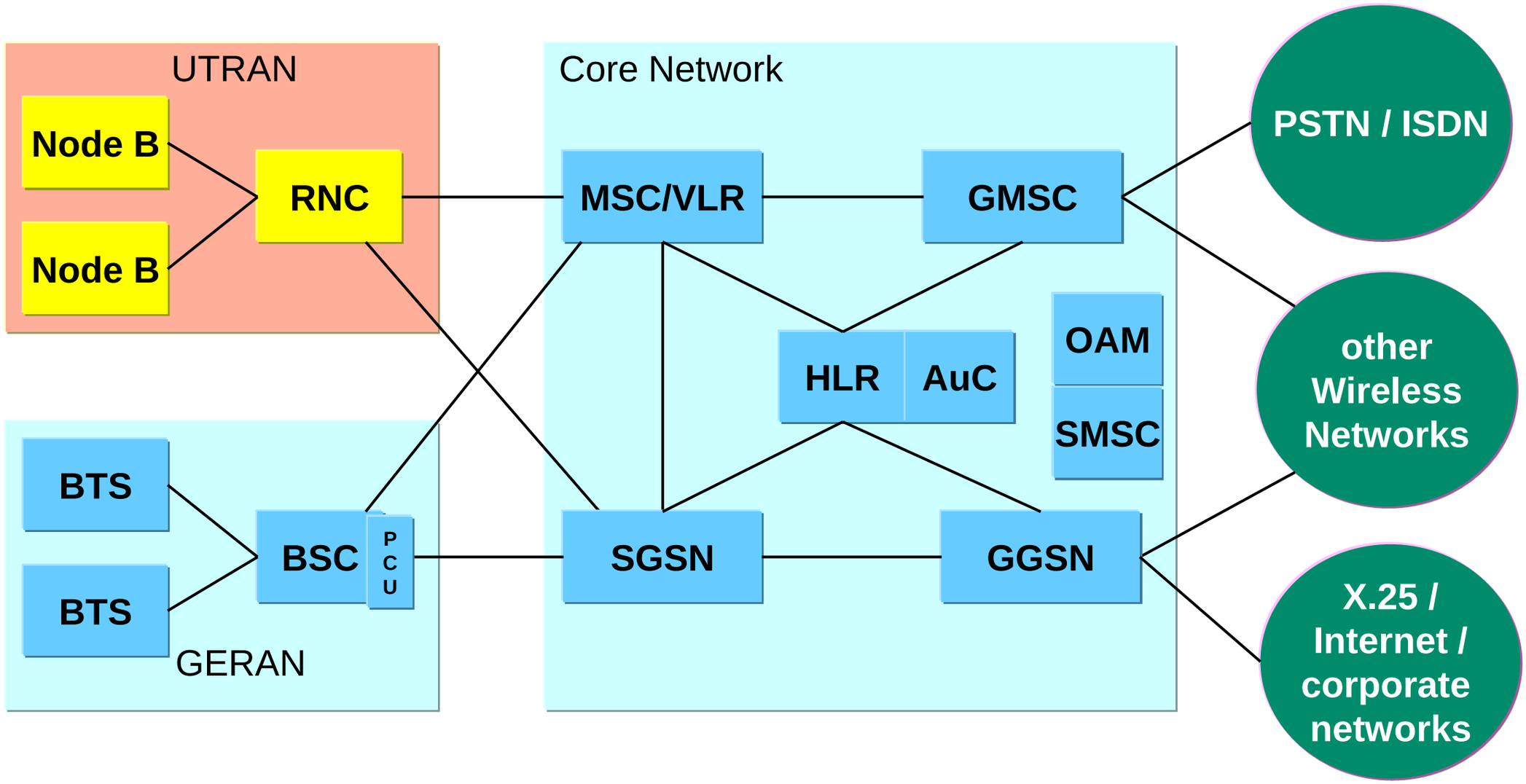
- **3GPP standards structured as/bundled in *releases***
  - **1992: *Phase 1* (GSM)**
  - **2000: *Release 99* incl. first specification of 3G UMTS**
  - **2008: Release 8 incl. first specification of LTE stuff**



- **At times, 3GPP standards are a bit… bulky ;-)**

# 2G/3G

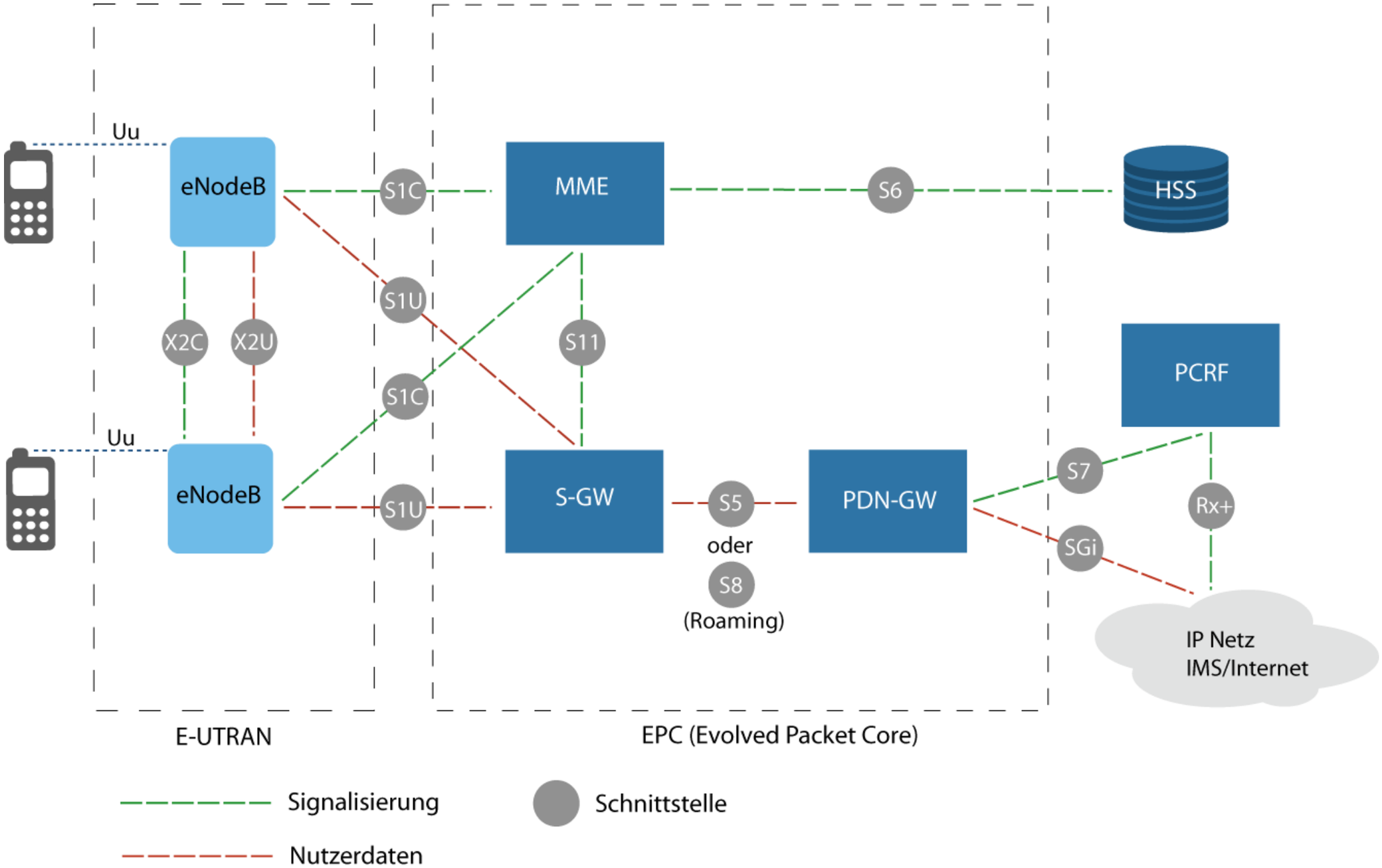
ERNW Living Security.




Source: 3GPP

RAN: Radio Access Network RNC: Radio Network Controller MSC: Mobile Switching Center AuC: Authentication Center
UTRAN: UMTS RAN BTS: Base Transceiver Station VLR: Visitor Location Register OAM: Operation Administration & Maintenance
GERAN: GSM Enhanced RAN BSC: Base Station Controller GMSC: Gateway MSC SMSC: Short Message Service Center
PCU: Paket Control Unit HLR: Home Location Register GSN: GPRS Support Node S/GGSN: Serving/Gateway GSN

# 4G

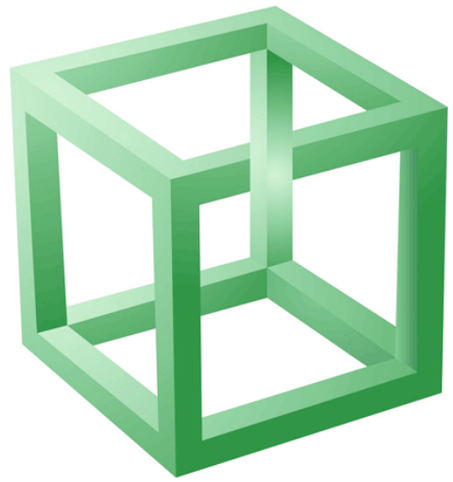

Uu
eNodeB
S1C
MME
S6
HSS
S1U
X2C
X2U
S11
S1C
PCRF
Uu
eNodeB
S1U
S-GW
S5
PDN-GW
S7
Rx+
oder
S8
(Roaming)
SGi
IP Netz
IMS/Internet
E-UTRAN
EPC (Evolved Packet Core)
Signalisierung
Schnittstelle
Nutzerdaten

ERNW
Living Security.

# Backhaul Networks

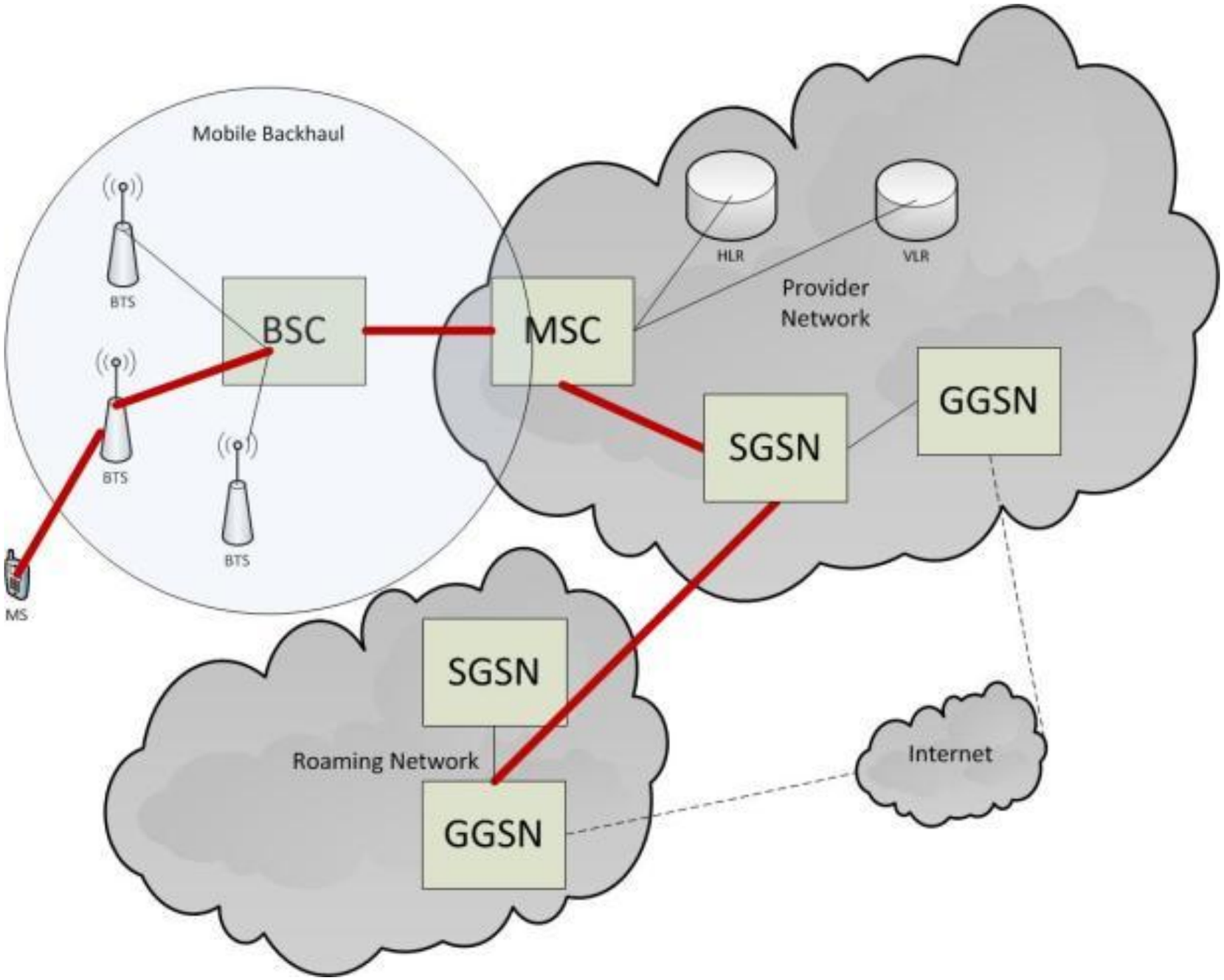
# Backhaul networks – Definition

- **In communication services**
  - **Used to transport information from one network node to another**
- **In mobile communication**
  - ***Mobile Backhaul***
  - **Carries data from the RAN to the management network and back.**
- **Three primary functions**
  - **Transport**
  - **Aggregation and grooming**
  - **Switching/routing**



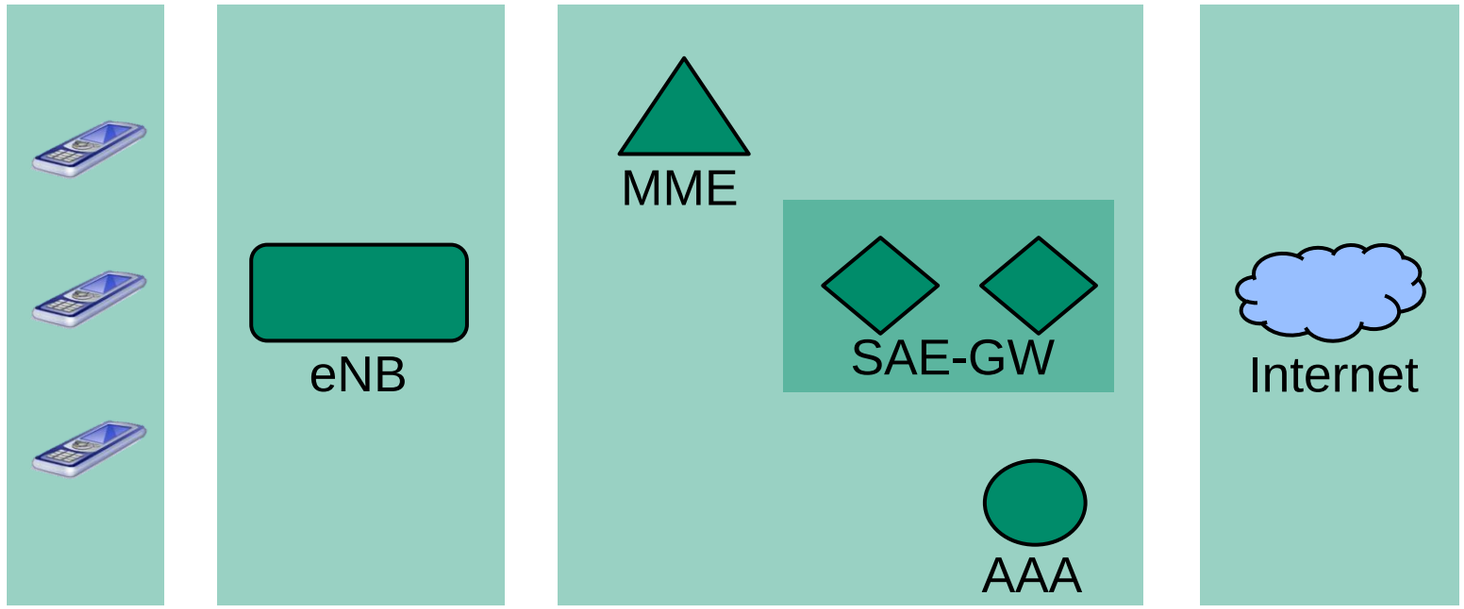
# Mobile Backhaul (3G)

ERNW
Living Security.



# Backhaul networks in 4G

- **4G specific requirements laid out by 3GPP**
- **Includes**
  - **eNodeB**
  - **MME**
  - **SGW**




- **Represents**
  - **The transport network between eNodeB and MME**
  - **The transport network between eNodeB and SGW**

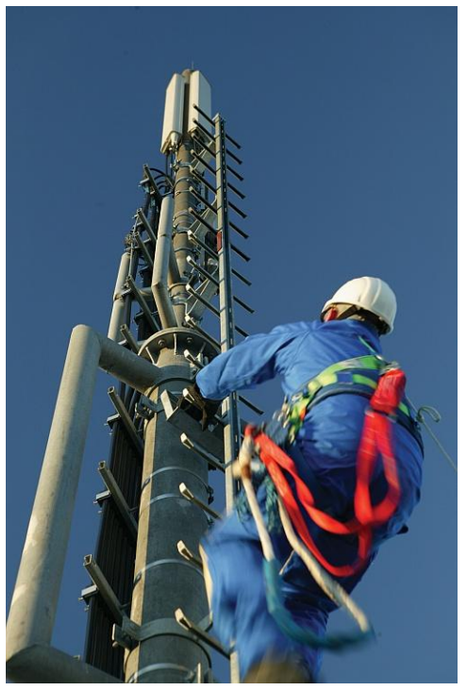
# Backhaul networks – Technologies

ERNW
Living Security.

- **Mostly ATM in the early years (GSM)**
- **PDH/SDH over Microwave, T1/E1**
- **IP/MPLS**
- **“Hybrid Approach“ with data offloading to DSL**
- **Carrier Ethernet**

# How to get into backhaul

- **Physical intrusion to some cage located “in the somewhere”**



- **Get access to “network segment”**
  - **Microwave**
  - **DSL**
  - **Carrier Ethernet**

- **4G aggregates “dumb” BTS and BSC/RNC functions on one device → eNB not “dumb” anymore.**

# Once you're in (a backhaul network)

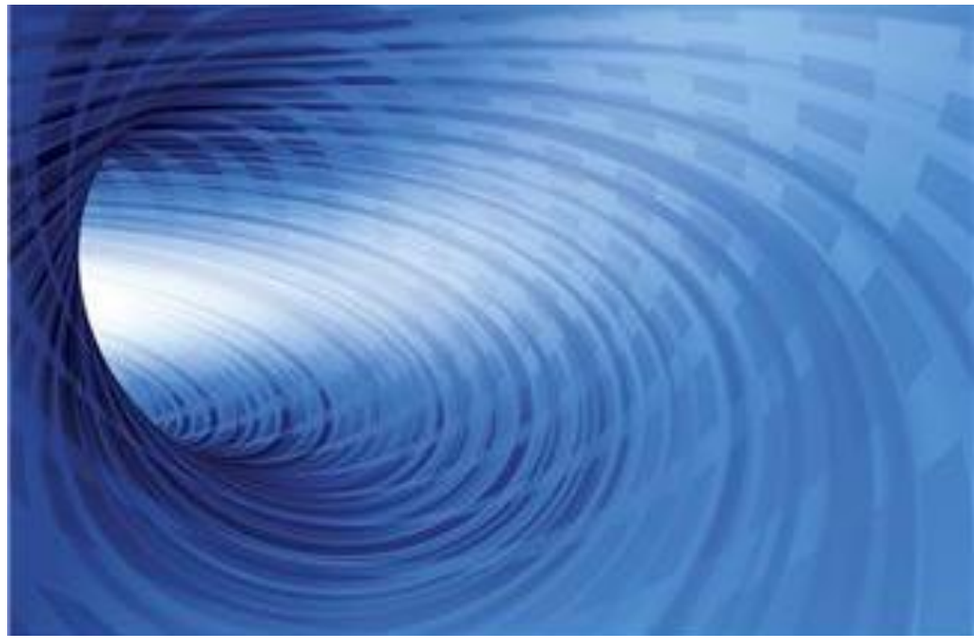
ERNW Living Security.

- **Attack components**
  - **3G: SGSN, RNC, NodeB**
  - **4G: MME, eNB, SAE-GW**
  - **Routers/Switches**

- **Eavesdropping**
  - **Will get you some key material**
    - **but what would you need this for? Pretty much everything is unencrypt. here anyway.**
  - **That's why 3GPP insists on using IPsec gateways.**
  - **Subsequent question: do (which) operators implement this?**
  - **In standard bodies $SOME_BIG_COUNTRY (hint: in Asia) strongly opposed this recommendation.**

# Protocols used in Backend

# GTP



- **GPRS Tunneling Protocol**
- **IP-based protocol initially used to carry GPRS within GSM and UMTS networks.**
  - **Plays major role in 4G networks as well.**
- **Three variants**
  - **GTP-C used for control plane (signaling)**
  - **GTP-U used for user data**
  - **GTP‘ used for charging data**

# GTP

- **GTP-C**
  - **Control section of the GTP standard**
  - **In 3G used for signaling between SGSN and GGSN**
  - **Activates and deactivates GTP sessions**
  - **In roaming scenarios this happens between different operators.**

- **GTP-U**
  - **Used for data transport between the RAN and the core network**
  - **Can tunnel packets in several formats: IPv4, IPv6, PPP etc. …**

- **GTP‘**
  - **Used in 3G for transmitting charging data from the CDF to the CGF.**

# GTP Header

- **The GTP Header**
  - **GTPv1**

| Bit 0-2 | 3 | 4 | 5 | 6 | 7 | 8-15 | 16-23 | 24-31 |
|---|---|---|---|---|---|---|---|---|
| Version | Protocol Type | Reserved | Extension Header Flag | Sequence Number Flag | N-PDU Number Flag | Message Type | Total length | |
| TEID | | | | | | | | |
| Sequence number | | | | | | N-PDU number | | Next extension header type |

  - **GTPv2**

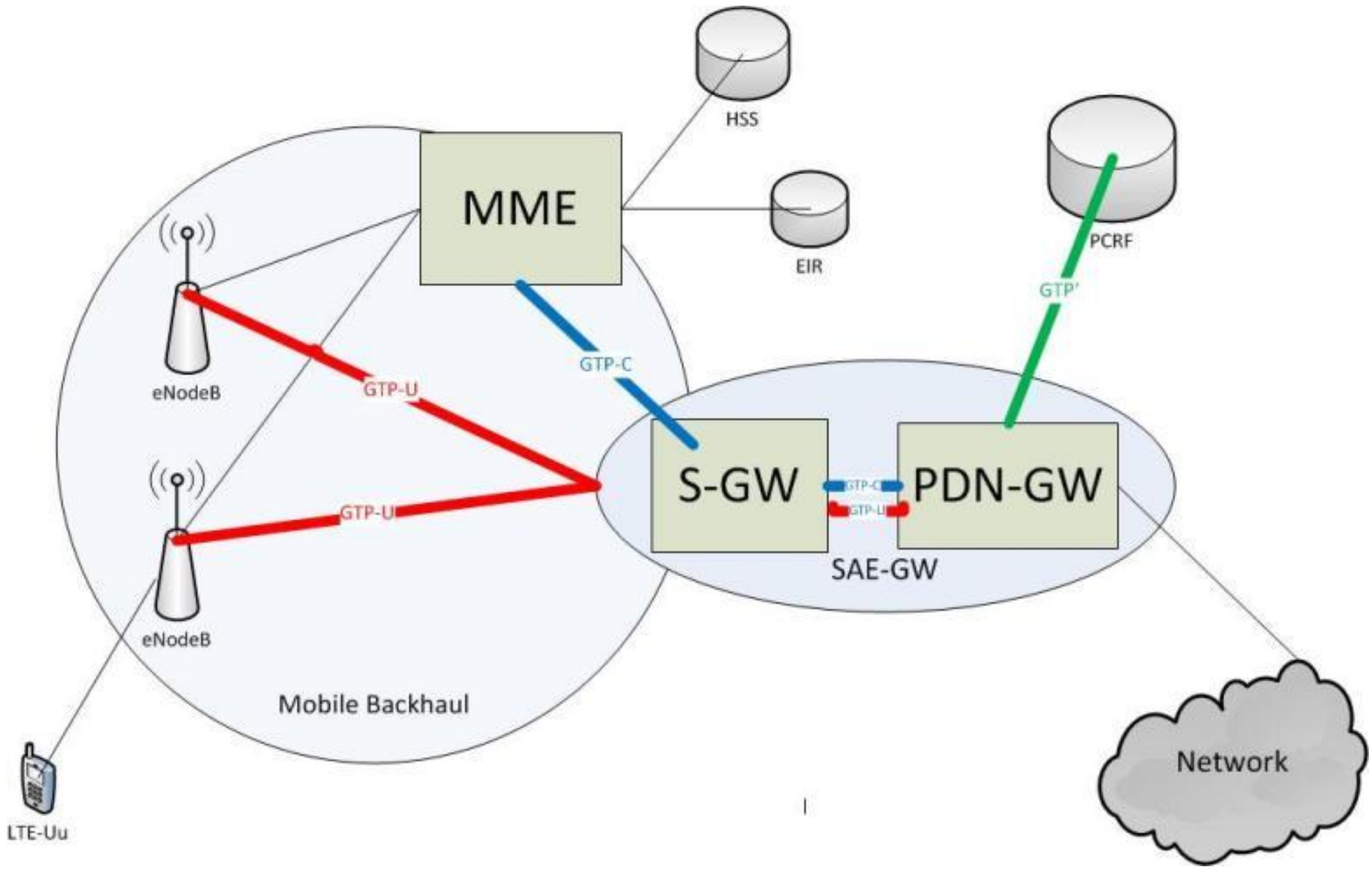
| Bit 0-2 | 3 | 4 | 5-7 | 8-15 | 16-23 | 24-31 |
|---|---|---|---|---|---|---|
| Version | Piggybacking flag (P) | TEID flag (T) | Spare | Message Type | Total length | |
| TEID (only present if T=1) | | | | | | |
| Sequence number | | | | Spare | | |

# Some GTP message types

- **GTP-C provides messages for**
  - **Echo**
  - **Create/Update/Delete/Initiate PDP Context**
  - **PDU Notification**
  - **Send Routing Information**
  - **Failure Report**
  - **Note MS/MS info**
  - **Identification**
  - **SGSN Context**
  - **Forward Relocation**
  - **Forward SRNS Context**
  - **RAN information**
  - **MBMS Notification/Context/(De-)Registration/Session**

# GTP in 4G

ERNW
Living Security.



# GTP-C

- **Control protocol for GTP session**

- **Very complex protocol**

- **A lots of different mandatory TLVs are defined for all the different Message types**

- **Even more optional TLVs are defined, plus vendor specific ‘secret’ TLVs**

# GTP-U

- **Tunneling protocol for ME-traffic.**
- **Static header length.**
- **Endpoint multiplexing done by 32bit TEID.**
  **(Tunnel Endpoint Identifier, more on that later)**

- **User data is transported in clear text**
- **No authentication mechanism in the protocol itself**

# GTP from a security perspective

- **Unauthenticated protocol**
- **No inherent security properties**
- **Trusted environment assumed**

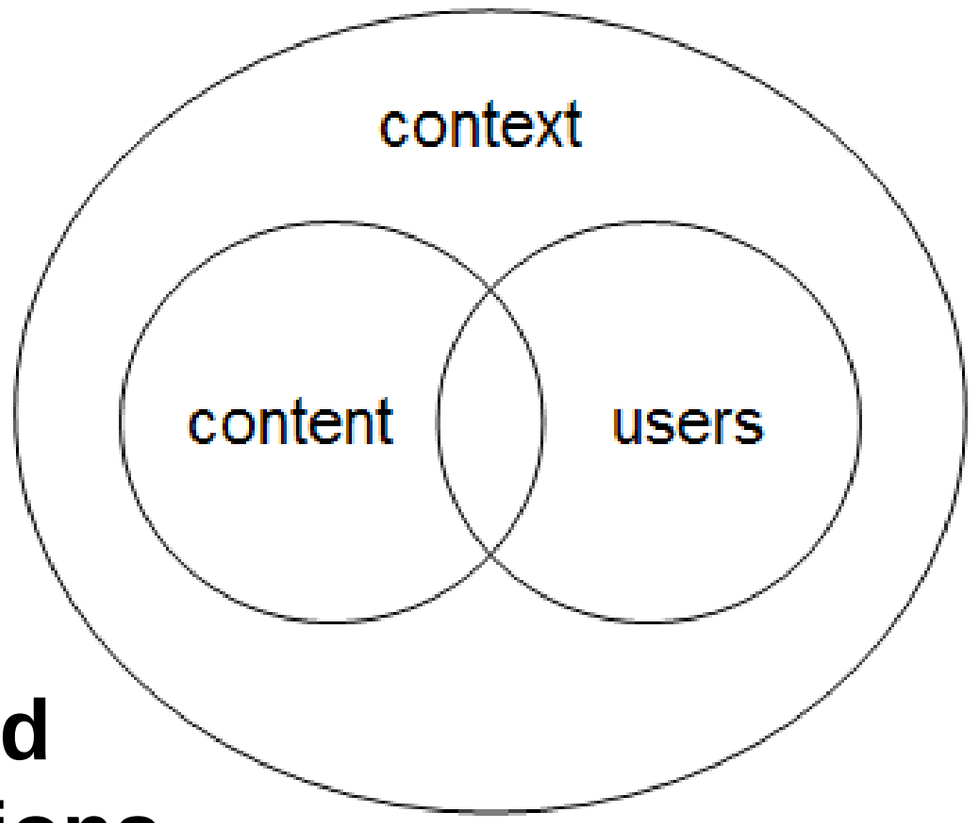
- **Is used to perform “quite some functions“**
  - **Session establishment (“activate PDP context“)**
  - **Forwarding of packets**
  - **Charging related stuff**

- **All these functions rely on certain protocol fields**
  - **Presumably only known to valid peers... which are isolated anyway...**

# The PDP-Context

- **Packet Data Protocol**

- **A PDP-Context is an established data connection from the Mobile station to the Network.**

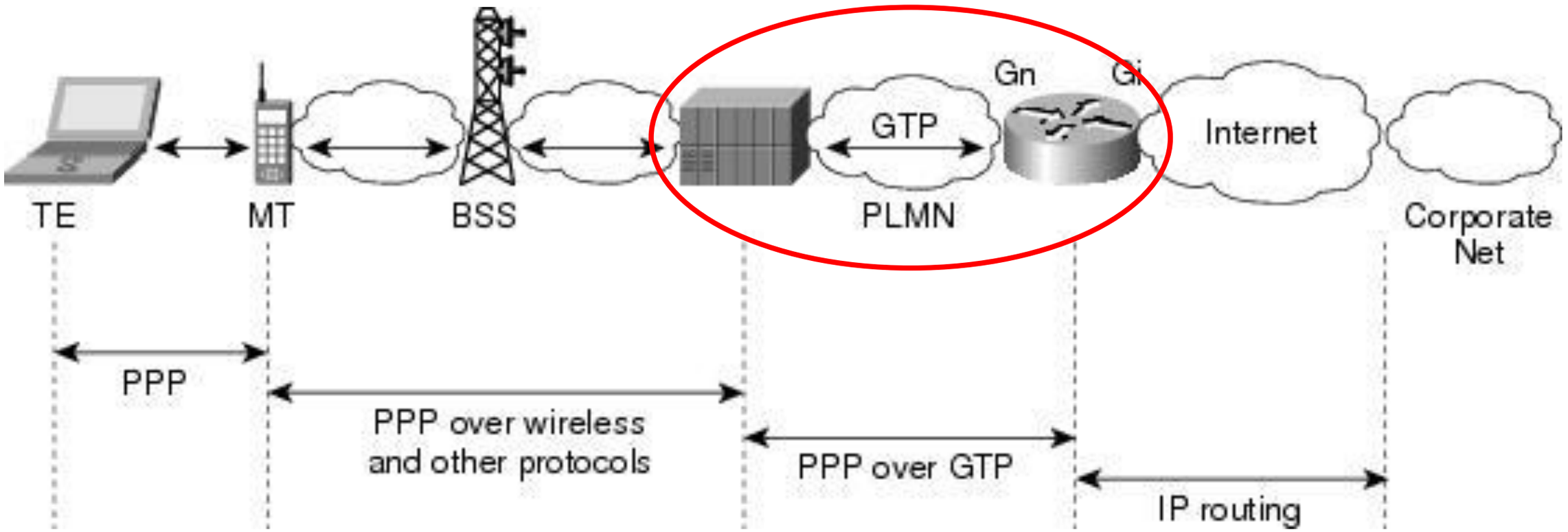



- **An Access Point Name (APN) is used to determine QoS and billing conditions.**

- **In 4G, also voice calls are data connections!**

# GTP session establishment

- **A GTP-PDPContext-request is sent via GTP-C, which includes a local TEID and an APN.**
- **If the APN is valid the request is answered with a GTP-PDPContext-response (including remote TEID).**




- **Afterwards GTP-U packets are processed.**

# TEID in detail

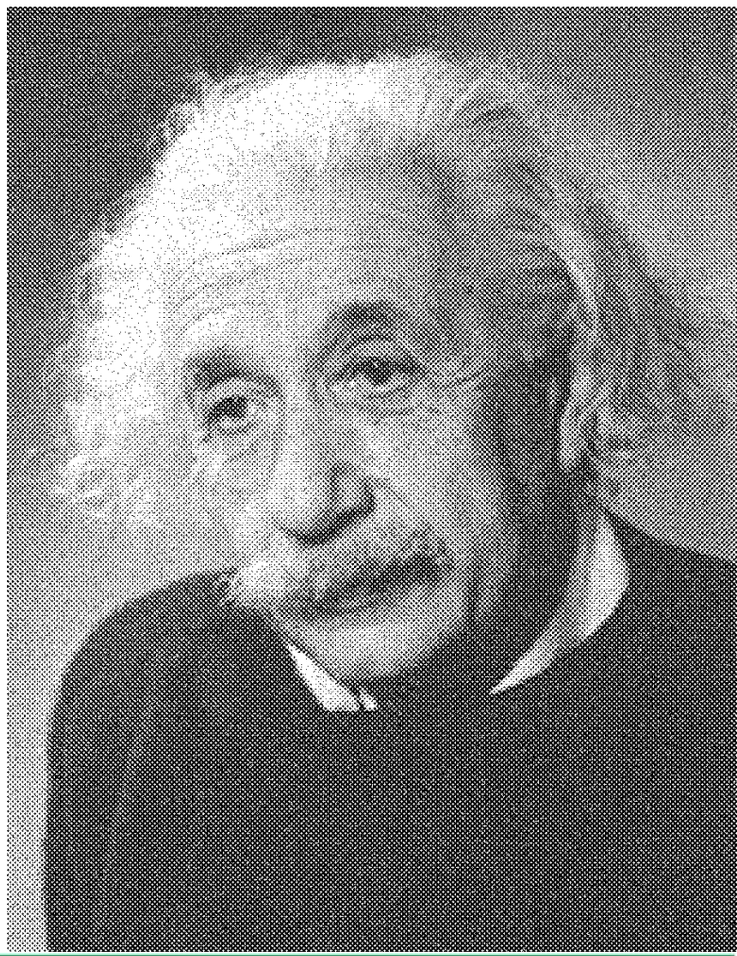
- ***Tunnel Endpoint Identifier***

- **Do I need to explain that it serves to *identify endpoints* of *tunnels*? ;-)**
  - **For each (user) data session.**

# TEID in detail

- **Apparently some discussion about it being random**
  - **For obvious (?) security reasons.**
  - **Although we were not able to find spec prescribing this.**

- **What we observed**
  - **0x00005c35**
  - **0x00005c4d**
  - **0x00005c65**
  - **0x00005c7d**
  - **0x00005c95**
  - **[…]**
  - **Does this look random to you ?**



# S1AP

- **S1 Application Protocol**

- **Used in 4G between eNodeB and MME (the S1 interface).**
- **Replaces GTP-C which is used in 3G on that interface.**
- **Uses SCTP for transport.**

- **Protocol is defined in ASN.1 only (!) in the 3GPP spec.**
- **Vendors implement proprietary extensions.**

- **What could possibly go wrong ;-)**

# S1AP – Details

ERNW Living Security.

- **We had the opportunity to test an eNodeB – MME pair, actively communicating over S1AP.**
- **Some things came to eyes early:**
  - **No authentication used whatsoever.**
  - **SCTP session is used to keep track of neighbor state.**
    **-> DoS via spoofed SCTP-ABORT packages.**


- **Others needed an fuzzing approach to come clear:**
  - **No good parsing of the (ASN.1 defined) protocol.**
  - **Fuzzing lead to major crash of the device.**

- **No tools or details released here, due to NDA.**

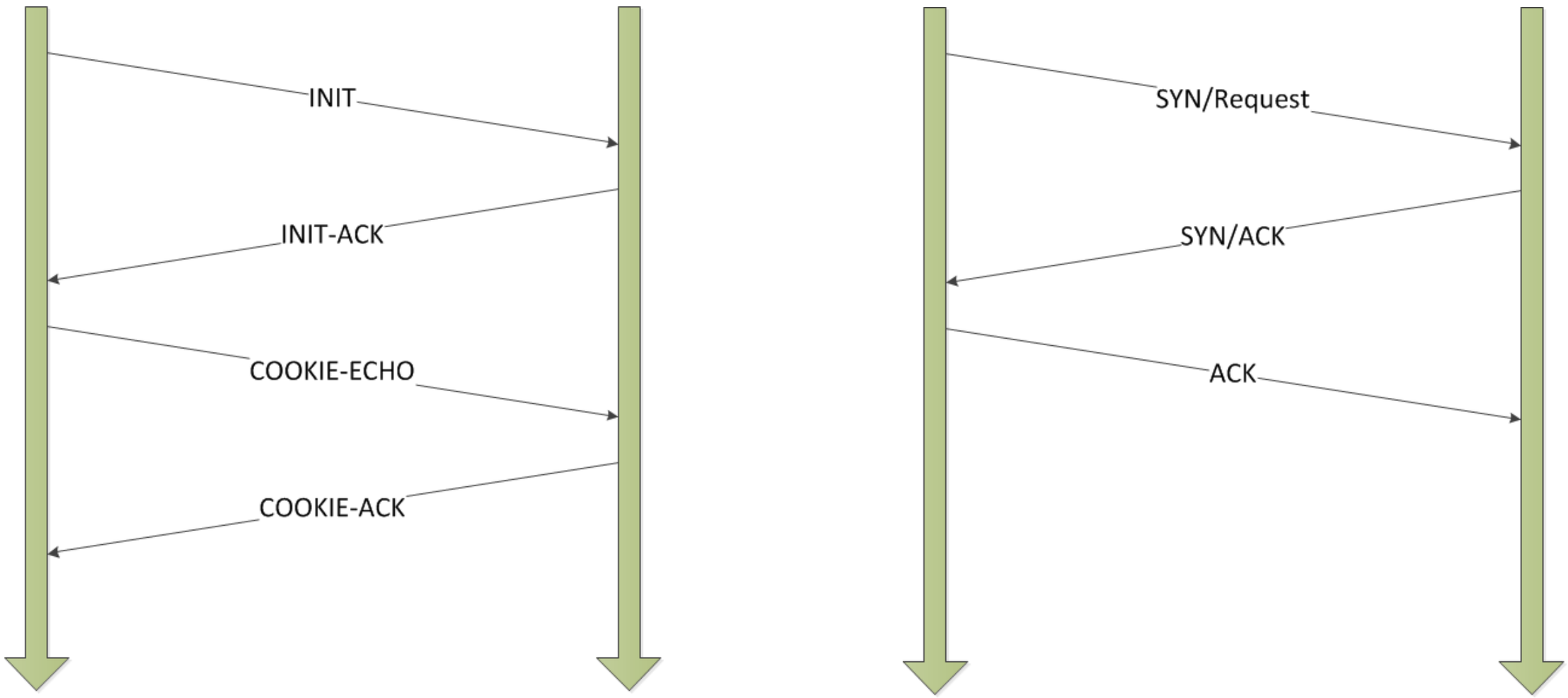
  **SORRY!**

# SCTP - Overview

- **SCTP**
  - **Stream Control Transmission Protocol**
  - **Specified by IETF, maintained IETF Transport Area (TSVWG) WG**
- **Specs:**
  - **RFC 3286 (Introduction)**
  - **RFC 2960 (2000)**
  - **RFC 3309**
  - **RFC 4960 (2007)**
  - **RFC 5062**

# SCTP – 4 way handshake

**SCTP** **vs.** **TCP**



# SCTP – Timeline

- **RFC 2960 (2000): initial spec**
- **RFC 4960 (2007): “major rewrite“**
- **RFC 5062 (2007) *Security Attacks Found Against the Stream Control Transmission Protocol (SCTP) and Current Countermeasures*”**

- **So, over time SCTP has changed a bit…**

# Tests in SCTP space – Practical problems

ERNW Living Security.

- **Current tools… do not work very well**
  - **Probably due to stack rewrites based on RFC 5206 and 4960**
- **nmap SCTP does not work “in a satisfactory manner”**
  - **-sZ does give results**
  - **-sY (“half-open handshake”) didn’t show anything useful**
    - **But we _knew_ the ports were there…**
- **Philippe Langlois‘ *SCTPscan* didn‘t work either.**

- **Daniel wrote quick+dirty “simple SCTP port scanner“.**

# SCTP hacked scanner ;)

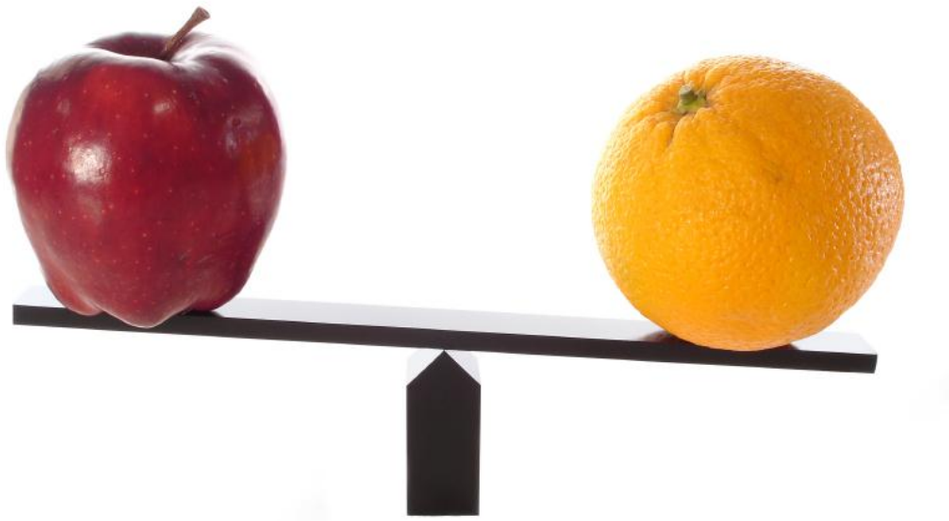
```
s = socket.socket(socket.AF_INET, socket.SOCK_SEQPACKET)
for i in ip:
    for j in xrange(sys.argv[2], sys.argv[3]):
        time.sleep(0.01)
        try:
            s.connect((j, i))
except Exception, e:
    print "Port %d closed on %s: %s" % (i, j, e)
else:
    print "Port %d open on %s" % (i, j)
    s.close()
```

**(this is more port-knocking no real port-scanning)**

# UDP vs. SCTP

ERNW Living Security.

- **UDP is ‘nice’ from an attackers point of view:**
  - **Easy to spoof**
  - **Fast to scan**



- **SCTP brings some effort to Man-in-the-Middle attacks**
  - **4-Way Handshake in performed**
  - **Security cookie is needed**

- **But, session termination by sending SCTP-ABORT packets no ‘hard thing’.**
- **In 4G, SCTP session state is used to track neighbor state**
  **-> DoS SGSN vs. GGSN**

ERNW
Living Security.

# The VMware

# The virtual machine

- **Minimal gentoo linux with**
  - **Username ‘root’**
  - **Password ‘toor’**
- **Tools and dependencies preinstalled**
  - **Tools in /root/tools**
  - **GTP dizz files in /root/tools/dizzes**
- **Wireshark on the host system is recommended**

# The virtual machine

- **Please make sure the virtual machine is running on your system.**
- **You will need it to follow the next part of the session.**

- **If you’re lacking Wireshark or VMware, both can be found in the local net:**

**http://10.0.0.1/**

ERNW
Living Security.

# The Lab

# GTP on 7200VXR

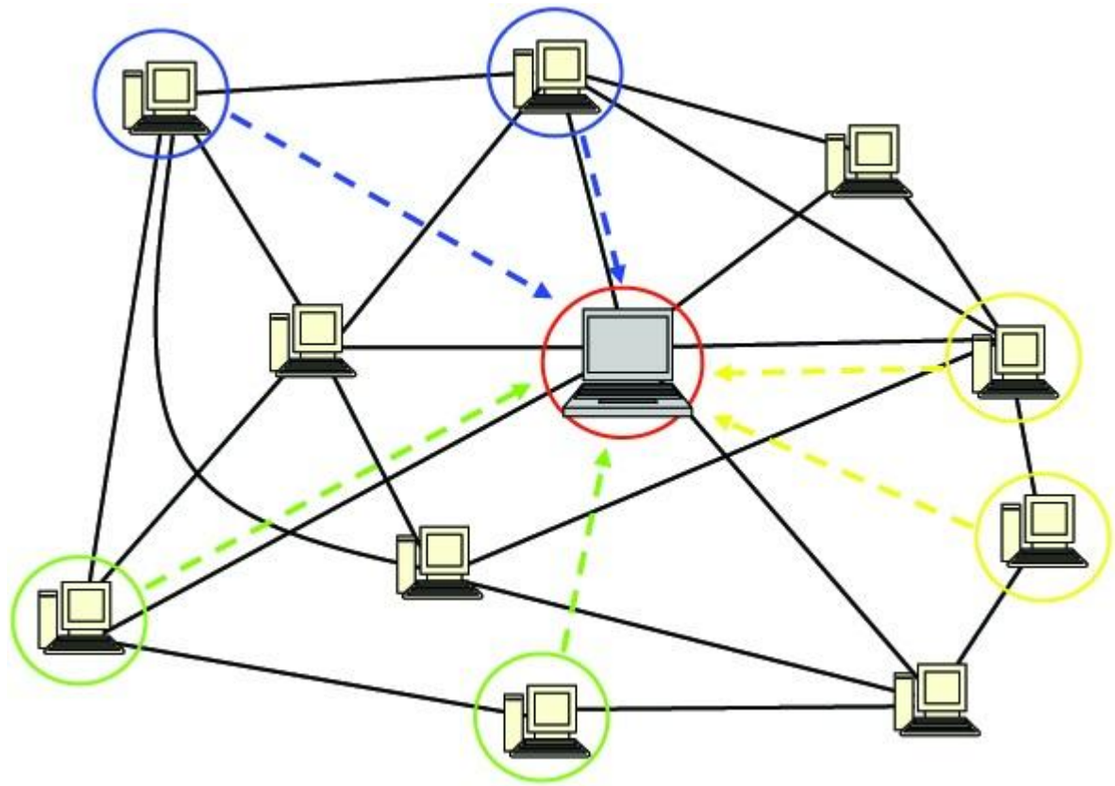
- **7200 is capable of serving as GGSN in a 3G net**
- **Special image needed**

- **`service gprs ggsn` config command**

- **Once activated, device opens up udp/2123 and udp/2152**
- **gtp-echo-requests (gtp-v1) are answered on both ports**
- **gtp-create-PDPcontext-requests (gtp-v1) are answered on udp/2123 (gtp-c) if a valid/configured APN is given in the request**

# Lab ranges

- **Local Network**
  - **DHCP enabled**
  - **10.0.0.0/24 gw 10.0.0.1**



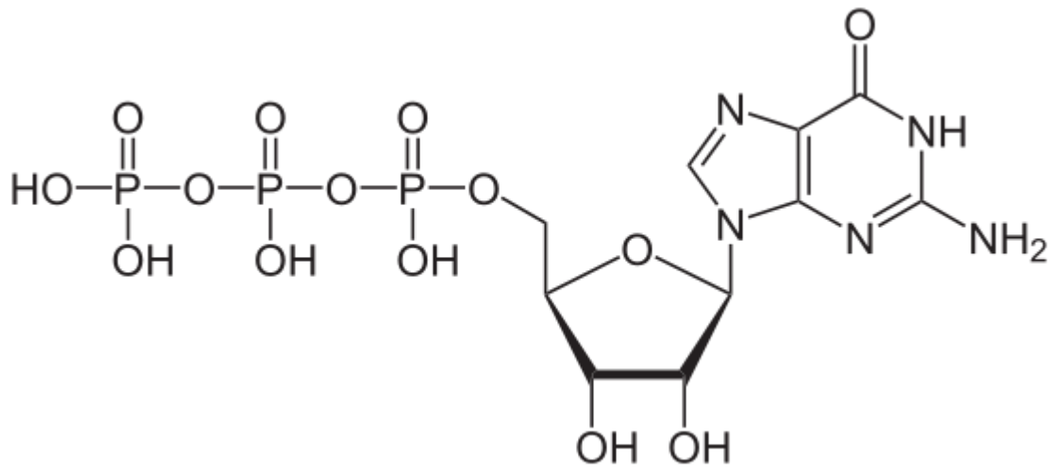
- **Target Network**
  - **172.25.1.0/24**

# The Tools

# gtp_scan

ERNW Living Security.

- **Scans a host to find gtp services on udp/sctp**
- **Python based**
- **Requires IPy**




- **Source:**
  - **http://c0decafe.de/tools/gtp_scan-0.7.tar.gz**

# gtp_scan – cmd

```
$ python gtp-scan.py --help
Usage: gtp-scan.py [options] address[/net]

Options:
  --version   show program's version number and
exit
 -h, --help   show this help message and exit
 -w SEC       Time to wait for cooldown
 -s           Use SCTP for transport, instead of
UDP
```

# gtp_scan – detail

- **GTP inbuilt ping mechanism is used to discover services.**
- **Scans for GTP-U, GTP-C and GTP'.**
- **Each port is tested with GTPv1 and GTPv2 `echo_requests`.**

- **Listening Services will send back a GTP `echo_response`, if no filtering is applied on the path.**

- **As hosts answer 'nicely' and UDP is used for transport, fast scanning of wide network ranges is possible.**

# Some statistics (GTP-C)

| | Version 1 | Version 2 |
|---|---|---|
| | | |
| AfriNIC | 26 (31) | 11 (26) |
| APNIC | 81 (131) | 97 (90) |
| ARIN | 52 (29) | 45 (51) |
| LACNIC | 22 (14) | 10 (18) |
| RIPE | 129 (97) | 94 (435) |
| | | |
| **UP** | **310 (302)** | **257 (620)** |

[Values in brackets are the results from our last scan, some months ago]

# apnbf

- **Script that brute forces the APN (Access Point Name) in GTPv1c.**
- **Python based**

- **Source:**
  - **http://c0decafe.de/tools/apnbf-0.1.tar.gz**

# apnbf – cmd

```
$ python apnbf.py --help
Usage: apnbf.py [options] address

Options:
  --version     show program's version number and
exit
 -h, --help    show this help message and exit
 -w WORDLIST   Wordlist to use
 -d SEC        BruteForce delay
 -v            Be verbose
```

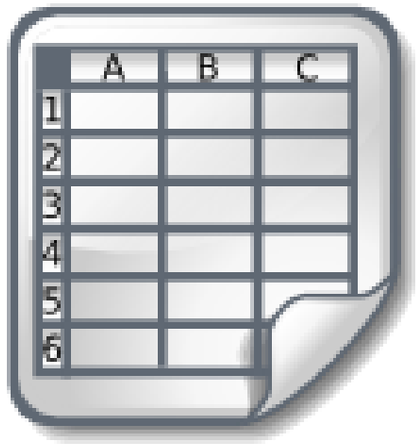
# apnbf – detail

- **Host are scanned for the possibility to establish a new PDP_context.**
  - **This requires a valid APN name.**
  - **If the establishment is possible, further attacks could be launched.**

- **Given list for APN names is brute forced.**
- **Returned error code gives a good impression of the hosts ‘shape’.**

# APNBF – results from the internetz

- **List of most used APNs in the Wild:**
  - **internet (12)**
  - **INTERNET (10)**
  - **Internet (10)**
  - **wap (5)**
  - **mms (5)**
  - **airtelnet.es (4)**
  - **online.telia.se (3)**
  - **cmnet (3)**




- **Some gtp speakers don’t care about the APN at all ;-)**

- **Python based fuzzing framework**
  - **Useful to fuzz GTP spreaker**

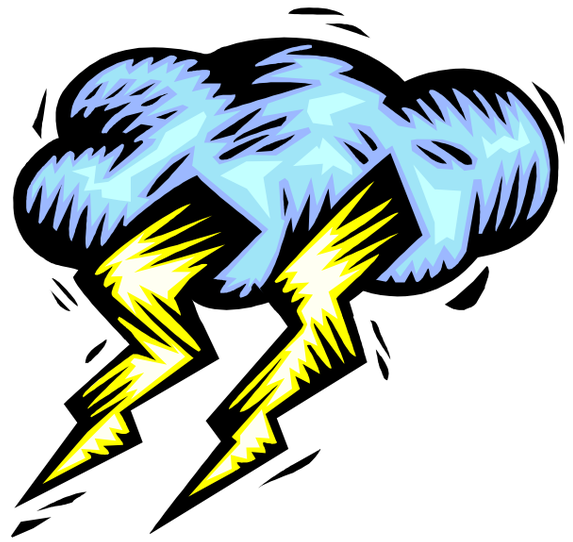
- **Requires pylibpcap and libdnet**

- **Source:**
  - **http://c0decafe.de/tools/dizzy-0.5.tar.gz**

# GTP on 7200VXR – DoS

ERNW Living Security.

- **Sending out _a_lot_ of gtp-echo-requests will stress the 7200er CPU to 100%, so that**
  - **No ICMP pings answered anymore.**
  - **No remote mgmt (ssh/telnet) possible (refuses connections on tcp/22).**
  - **No further GTP requests processed.**



- **Sending out _a_lot_ of gtp-create-PDPcontext-requests will also stress the device, so that only ~30% of all (valid and bogus) requests are answered.**
- **However a valid APN is needed**
  - **We'll get back to this ☺**

ERNW
Living Security.

# Exercises

# Scan for GTP

- **Scan the target range [172.25.1.0/24] for GTP* speaking devices.**

```
#cd /root/tools/gtp_scan-0.7/
```

```
#python gtp-scan.py 172.25.1.0/24
```

# Scan for GTP

```
gtp-scan v0.7        Copyright 2011 Daniel Mende <mail@c0decafe.de>
starting scan of 172.25.1.0/24
cooling down for 10 sec...
### 172.25.1.3 up, from udp/2123(gtp-c) sent 32030004000000000000000
     *** VALID LEN IN GTP:     version = 1 flags = XXX1 0010 type = 3
     *** VERSION NOT SUPPORTED
### 172.25.1.3 up, from udp/3386(gtp') sent 3e030000ff00000000000000
     version = 1 flags = XXX1 1110 type = 3 len = 0 data = ff0000
     *** VERSION NOT SUPPORTED
### 172.25.1.3 up, from udp/2123(gtp-c) sent
32020006000000000c3d00000e01
     *** VALID LEN IN GTP:     version = 1 flags = XXX1 0010 type = 2
     *** ECHO RESPONSE
done
```

# Find the right APN

- **Find at least one valid APN on the identified GTP-C speaking devices.**

```
#cd /root/tools/apnbf-0.1
#python apnbf.py -w apnlist 172.25.1.3
```

# Find the right APN

```
apnbf v0.1          Copyright 2011 Daniel Mende <mail@c0decafe.de>
starting scan of 172.25.1.3
trying internet.gprs.unifon.com.ar
     Missing or unknown APN
trying internet.unifon
     Missing or unknown APN
trying internet.ctimovil.com.ar
     Missing or unknown APN
trying internet
*** APN FOUND: internet

trying telstra.internet
     Missing or unknown APN
```

# Establish a valid PDP-Context

- **Find the `gtp_create_pdp_context_request.dizz` in the dizzes/gtp_v1/ folder on the virtual machine.**

- **Edit the APN_value field to match the discovered APN:**

```
{    '_name': 'APN_value',
     '_type': 'basic',
'bytelen': None,
'cur': '\x0bAPN_HERE',
'default': '\x0bAND_HERE',
'fuzz': 'none',
'length': None},
```

# Establish a valid PDP-Context

```
#nano /root/tools/dizzes/gtp_v1/gtp_create_pdp_context_request.dizz
```

- **press `CTRL+W` for find**
- **enter the search term `APN_value`**
- **replace the `ernwtel.com` with your APN**
- **press `CTRL+O` to save and `CTRL+X` to exit the editor**

# Establish a valid PDP-Context

- **Once the dizz file is prepared, start up dizzy and send the described packet once:**

```
#python dizzy.py -t -o udp -d 172.25.1.3 -e
2123:2123
../dizzes/gtp_v1/gtp_create_pdp_context_request.dizz
```

- **Look into Wireshark on your host system and examine the answer. What do you see?**

# Move on to real fuzzing

- **Establishing a valid PDP-context is nice and the first step for GTP state-full fuzzing, but we will stay with state-less fuzzing for this time, because:**
  - **This a 3G/4G lab session, no fuzzing training ;)**
  - **I don’t had the time (yet) to write state-full fuzzing scripts (although dizzy is usable as a state-full fuzzer)**
  - **We don’t want to kill the telco industry today :-D**

# Move on to real fuzzing

- **Edit the `gtp_create_pdp_context_request.dizz` file again and set every field you want to be fuzzed to:**

```
'fuzz': 'std',
```

- **Launch up the same dizzy command but remove the `-t` (testing) flag.**
- **Sit back and watch Wireshark ;-)**

- **BTW, what's the load on the target?**

# Conclusions

- **We expect to see a number of attacks in 3G and 4G mobile telco networks in the next years, for some reasons**
  - ***Walled (telco) gardens* are vanishing.**
  - **At the same time “terminals“ get more and more powerful.**
  - **In the future it‘s all IP in those networks.**
  - **There‘s a complex (IP based) protocol landscape.**

  **And potentially ppl_outside_telcos are able to understand these prots.**

  **As there are apparently people understanding Siemens PCS 7...**

- **Theory ≠ reality**

There’s never enough time…

ERNW
Living Security.

**THANK YOU…**


**...for yours!**