ERNW
providing security.

How to build a Stingray

Hendrik Schmidt <hschmidt@ernw.de> / @hendrks_

Brian Butterly <bbutterly@ernw.de> / @BadgeWizard

ERNW
providing security.

Cellular Networks

- Connecting $mobile_devices which each other
  - Internet of Things (EGSM, LTM-M)
  - Automotive Systems
  - Industry 4.0

- Using Services as
  - Voice
  - Data
  - Messaging
  - OTA Updates

ERNW
providing security.

The Goal?

- Simulating a real world environment / a provider
- Interception of mobile data

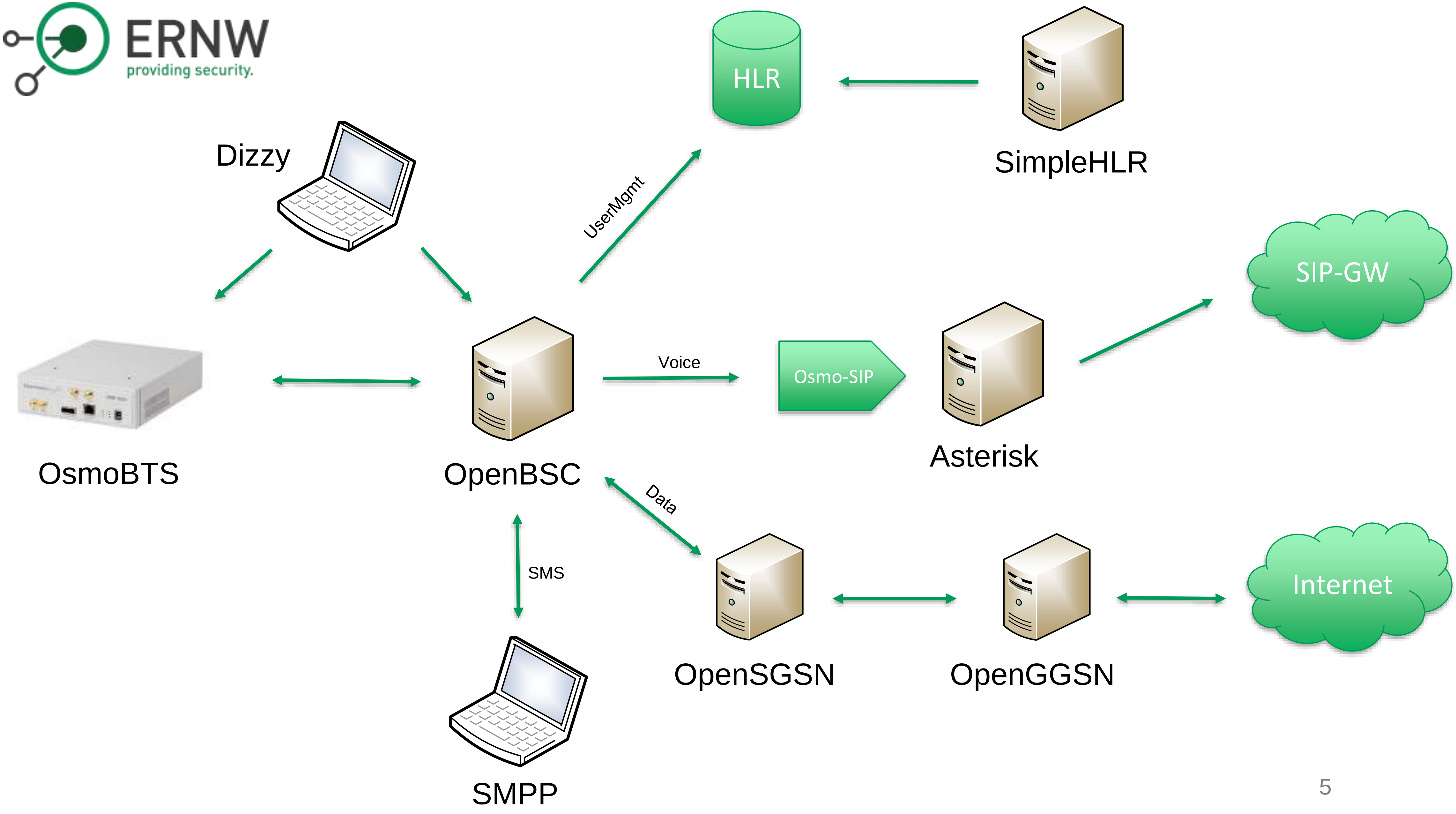
- Raw Data Access
  → Open Source?
- Portable
- Monitoring Capabilities
  - Wireshark?

→ What, we are building our own Stingray?

Tools

- GSM
  - phone: osmocomBB
  - network: openBSC, osmoBTS, openBTS, gr-gsm
- UMTS
  - phone: xgoldmon
  - network: openBTS-UMTS
- LTE
  - phone: Samsung Kalmia, SnoopSnitch
  - network: Amarisoft, openLTE, srsLTE, OpenAirInterface

ERNW
providing security.



Why is this Working?

- Mobile Connection depends on
  - MCC / MNC (Roaming SIM?)
  - Encryption Keys

    → Can be ignored when using A5/0
  - APN
  - SMSC-Number

- Limitations
  - GPRS/EDGE (UMTS available soon)
  - Restricted APNs

(Brief) Cell Selection

1. Build Cell Selection Table
2. Read Last Cell from SIM
3. Select Home Network (best/loudest)
4. Select Roaming Network (best/loudest)

Challenges:

- Cell Fixation
- Higher privileged networks (LTE)

→ Jamming

Data Interception
(eliminating the magic)

- GPRS Data Access
- “Common” Pentest Methodology
  - Identification of running services
  - Eavesdropping & Encryption Tests
  - Man-in-the-Middle of Communication

- Ever used a M2M SIM for free Internet?

Voice Interception

- Intercepting Calls like a Full-MitM-IMSI-Catcher
  - Testing implemented Security Measures (Authentication/Encryption)
  - Emergency Calls
- SIP based Uplink to PSTN
- Ever used a M2M SIM for free calls?

ERNW
providing security.

Short Messaging Service

- SMS PDU Attacks
- SMS UDH Attacks
- Application access via SMS

- OTA Updates via (8-bit) binary Data
  - Depends on PID/DCS
- Data Forward to SIM

- Ever used a M2M SIM for free SMS?

ERNW
providing security.

The Python Code

```
def send_message(destaddr, dcs, pid, message):
        print 'Sending SMS "%s" to %s' % (string,dest)
        pdu = client.send_message(
                source_addr_ton=smpplib.consts.SMPP_TON_INTL,
                source_addr_npi=smpplib.consts.SMPP_NPI_ISDN,
                source_addr='1001',
                dest_addr_ton=smpplib.consts.SMPP_TON_INTL,
                dest_addr_npi=smpplib.consts.SMPP_NPI_ISDN,
                destination_addr=destaddr,
                data_coding=dcs,
                protocol_id=pid,
                esm_class=smpplib.consts.SMPP_GSMFEAT_UDHI,
                short_message=message,
                registered_delivery=False,
        )
        print(pdu.sequence)
```

- TP-DCS:
  - GSM 7-Bit
  - 8-Bit Data
  - UCS-2
  - Message Class
- TP-PID
  - Forward SM
  - Data Download (125)
  - U(SIM) Data Download (127)
  - … and more
- Furthermore
  - UDHI
  - Status-Reports
  - Tracing

Radio Layer Access

- Testing Security of BaseBand & Configuration
  - Authentication/Encryption
- RAW PDU Access enables us to do targeted Fuzzing

Demo!

ERNW
providing security.

Conclusion

- Build an Full-MitM IMSI Catcher with (mostly) Open Source Tools
  - Whitehat vs. Blackhat
- Code Access and Interfaces to send Raw Data

- What can you do to avoid connecting to my fake cell?
  - Quite nothing, watch out for signs on your phone. It’s up to the standards and their implementations