icc: iMSI catcher catcher

Jan Kuipers

j.h.kuipers@student.utwente.nl

David Stritzl

david.stritzl@gmail.com

Santiago Aragón

s.e.aragonramirez@student.utwente.nl

Iwan Timmer

i.r.timmer@student.utwente.nl

University of Twente

Abstract

This program tries to find nearby IMSI catchers using a RTL_SDR device. TODO: Diagram

1. Motivation and Outline

What is an IMSI Catcher Why to spoit it Possible approaches and difficculties Our approach Outline

An IMSI Catcher (IC) is a device used to perform Man-in-the-Middle(MitM) attacks on cellular networks. The first IMSI catchers date from the 90's when this devices were almost exclusively affordable by law-enforcement agencies. However, the Software Define Radio (SDR) technological advances and the active open-source community have facilitated and cheapen the development of this devices and therefore making them available to a larger number of users (TODO: cite some examples Nohl, Paget etc). Furthermore, the lack of security in 2G, GSM nationwide deployments, and the backwards compatibility of almost every device in cellular networks gamble the privacy and security of our cellular communications.

IC needs to mimic the behavior of a real Base Station Transceiver (BTS), while performing a MitM attack, thus an IC produce some artifacts over a BTS's normal herbivorous. This artifacts can be detected by an IMSI Catcher Catcher (ICC) directly on a end-user mobile phone or using the necessary hardware, e.g., SDR.

2. Design

2.1. Detection methods

Detections methods (DM) are defined as python scripts in detectors/some_dector.py. Every method should extend the class Detector specified in detectors/Detector.py and define its own callback function, e.g.:

```
def handle_packet(self, data):
    p = GSMTap(data)

    if p.payload.name is 'LAPDm' and
       p.payload.payload.name is 'GSMAIFDTAP' and
       p.payload.payload.payload.name is 'CipherModeCommand':
            cipher = p.payload.payload.payload.cipher_mode >> 1

            if cipher == 0:
                self.update_s_rank(Detector.SUSPICIOUS)
                self.comment = 'A5/1 detected'
            ...
```

This function will be applied packet wise and should rank the anylyzed BTS and at the end modify the `s_rank` and `comment` variables calling `self.update_s_rank(RANK)`(resp. `self.comment='A descriptive comment'`).

We define rank the suspiciones of a BTS as

```
SUSPICIOUS = 2
UNKNOWN = 1
NOT_SUSPICIOUS = 0
```

At the end of the detection the detectors return a `TowerRank` object.

3. Implementation details

4. Limitations and future work

References