GSMem

Data Exfiltration from Air-Gapped Computers over GSM Frequencies

Mordechai Guri, Assaf Kachlon, Ofer Hasson, Gabi Kedma, Yisroel Mirsky, Yuval Elovici

Ben-Gurion University of the Negev, Israel
Background
BRIDGING THE AIR GAP
Background
Air Gapped Networks

Definition: A cyber security measure that secures computer network by physically isolating it from unsecured networks, such as the public Internet or another unsecured local area network.

Examples of air gapped networks:
• Military defense system
• Critical infrastructure command and control centers
• Computerized medical equipment
• Finance
• And more…
Background
Air Gapped Networks

The Scenario:
• An attacker has succeeded in infecting the network
  • USB, insider, etc...
• The Attacker now wants to retrieve data from that network (over the air gap).
## Previous Work

<table>
<thead>
<tr>
<th>Method</th>
<th>Transmitter</th>
<th>Receiver</th>
<th>Distance (m)</th>
<th>Rate (bit/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AirHopper [23]</td>
<td>Display cable</td>
<td>Cellular FM receiver</td>
<td>7</td>
<td>104-480</td>
</tr>
<tr>
<td>Ultrasonic [21] [24]</td>
<td>Speaker</td>
<td>Microphone</td>
<td>19.7</td>
<td>20</td>
</tr>
<tr>
<td>SAVAT [22]</td>
<td>CPU/memory (laptops)</td>
<td>Dedicated equipment</td>
<td>1.0</td>
<td>N/A</td>
</tr>
<tr>
<td>BitWhisper [25]</td>
<td>Computer CPU/GPU</td>
<td>Computer Heat Sensors</td>
<td>0.4</td>
<td>8 bit/hour</td>
</tr>
</tbody>
</table>
## Background

### Previous Work

<table>
<thead>
<tr>
<th>Method</th>
<th>Transmitter</th>
<th>Receiver</th>
<th>Distance (m)</th>
<th>Rate (bit/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AirHopper [23] (78MHz - 108MHz)</td>
<td>Display cable</td>
<td>Cellular FM receiver</td>
<td>7</td>
<td>104-480</td>
</tr>
<tr>
<td>Ultrasonic [21] [24]</td>
<td>Speaker</td>
<td>Microphone</td>
<td>19.7</td>
<td>20</td>
</tr>
<tr>
<td>SAVAT [22] (~80KHz)</td>
<td>CPU/memory (laptops)</td>
<td>Dedicated equipment</td>
<td>1.0</td>
<td>N/A</td>
</tr>
<tr>
<td>BitWhisper [25]</td>
<td>Computer CPU/GPU</td>
<td>Computer Heat Sensors</td>
<td>0.4</td>
<td>8 bit/hour</td>
</tr>
</tbody>
</table>
## Background

### Previous Work

<table>
<thead>
<tr>
<th>Method</th>
<th>Transmitter</th>
<th>Receiver</th>
<th>Distance (m)</th>
<th>Rate (bit/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AirHopper [23]</td>
<td>Display cable</td>
<td>Cellular FM receiver</td>
<td>7</td>
<td>104-480</td>
</tr>
<tr>
<td>[78MHz - 108MHz]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ultrasonic</td>
<td>Speaker</td>
<td>Microphone</td>
<td>19.7</td>
<td>20</td>
</tr>
<tr>
<td>[21] [24]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAVAT [22]</td>
<td>CPU/memory (laptops)</td>
<td>Dedicated equipment</td>
<td>1.0</td>
<td>N/A</td>
</tr>
<tr>
<td>(~80KHz)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BitWhisper [25]</td>
<td>Computer CPU/GPU</td>
<td>Computer Heat Sensors</td>
<td>0.4</td>
<td>8 bit/hour</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An ordinary desktop PC is converted into a small transmitting cellular antenna!
Demonstration Video
https://www.youtube.com/watch?v=RChj7Mg3rC4
Transmitter
GSMEM
Transmitter
CPU-Memory BUS Emissions

How do we convert a computer’s CPU-RAM configuration into a **radio antenna**?

How do antennas work?

- Antennas emit radio waves (EMR) by **oscillating** current through their terminals.
- Radio waves are characterized by their **frequency** (oscillation in Hz) and **amplitude** (strength in dBm).
Transmitter
CPU-Memory BUS Emissions

How do we get this “antenna” to emit EMR on a cellular band (range of frequencies)?

- **Observation 1:** A large CPU-RAM transfer builds up oscillating current in the configuration.
- **Observation 2:** The BUS transfers bits at the FSB speed, emitting the energy around that frequency (e.g. 800 MHz)

![Graph showing emissions at different frequencies](image)
Transmitter

CPU-Memory BUS Emissions

Algorithm 1 transmit32 (data)

1: buffer ← ALIGNED_ALLOCATE(16,4096)
2: tx_time ← 500000
3: for bit_index ← 0 to 32 do
4:     if (data[bit_index] = 1) then
5:         start_time ← CURRENT_TIME()
6:     while (tx_time > CURRENT_TIME() - start_time) do
7:         buffer_ptr ← buffer
8:         for i ← 0 to buffer_size do
9:             SIMDNTMov(buffer_ptr, 128bit_register)
10:            buffer_ptr ← buffer_ptr + 16
11:        end for
12:     end while
13:     else
14:         SLEEP(tx_time)
15:     end if
16: end for
Transmitter
Sending a Bit (Modulation)

To send a bit, we use a variant of B-ASK:

Send( “0” ): Do nothing for $T$ seconds
Send( “1” ): Raise amplitude for $T$ seconds
Transmitter
Sending Lots of Bits (Framing)

To send a sequence of bits (some data payload) we perform framing.

*This is for the benefit of the receiver to perform:*
1. Transmission detection
2. Synchronization
3. B-ASK threshold selection (what amplitude is “0”?)
   - Dynamically updated (change in distance...)

<table>
<thead>
<tr>
<th>Preamble</th>
<th>Payload</th>
<th>Preamble</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>1010</td>
<td>12 bits</td>
<td>1010</td>
<td>12 bits</td>
</tr>
</tbody>
</table>
Transmitter
Properties & Characteristics of the Transmitter

• Only has a 4KB memory footprint
• No root/admin required
• No APIs are used

• Affects Intel and AMD architectures...
• Works on Windows/Linux...
Receiver
About Modifying Phones...

Baseband processor:

• The connection with the cellular network is managed by a dedicated chip, called the “baseband”.
• Completely separated from the main OS (e.g., Android).
• Firmware of all common brands is closed-source

This will not deter highly motivated, and resourceful threats …as we’ve seen in the past.
Then how did we modify the firmware?

**OsmocomBB**: An open source GSM baseband software implementation (2010)

- For our experiments, used the **OsmocomBB** compatible Motorola C123 GSM phone.

We note that GSMMem can even work on a nine-year old, low-end mobile phone... modern technology can go even further.
Receivers
Getting the bits

A Very Simplistic Approach:

1. Listen on “best” frequency
2. Search for the ‘1010’ preamble (each bit $T$ seconds long)
   • Threshold based (dynamically changed)
3. Extract 12 bit payload if preamble is found

Frequency Domain

Time Domain

www.tmatlantic.com
Receivers
Getting the bits

Algorithm 2 ReceiverHandler

1: \( dBm \leftarrow \text{MEASURE}(f_c) \)
2: \( \text{filtered\_signal} \leftarrow \text{UPDATE\_MOVING\_AVERAGE}(dBm) \)
3: \( \text{if} \ (\text{state} = \text{SCAN}) \ \text{then} \)
4: \( f_c \leftarrow \text{SCAN\_FREQ()} \)
5: \( \text{setState}(\text{PREAMBLE}) \)
6: \( \text{end if} \)
7: \( \text{if} \ (\text{state} = \text{PREAMBLE}) \ \text{then} \)
8: \( \text{if} \ (\text{IDENTIFY\_PREAMBLE}(\text{filtered\_signal}) = \text{true}) \ \text{then} \)
9: \( \text{setState}(\text{RECEIVE}) \)
10: \( \text{end if} \)
11: \( \text{end if} \)
12: \( \text{if} \ (\text{state} = \text{RECEIVE}) \ \text{then} \)
13: \( b \leftarrow \text{DEMODULATE\_BIT}(\text{filtered\_signal}) \)
14: \( \text{bitSequence}.\text{add}(b) \)
15: \( \text{if} \ (\text{bitSequence}.\text{size}\%16 = 0 \text{ or } \text{SIGNAL\_LOST}(\text{filtered\_signal})) \ \text{then} \)
16: \( \text{setState}(\text{PREAMBLE}) \)
17: \( \text{end if} \)
18: \( \text{end if} \)
Evaluation

GSMEM
## Evaluation
### Experiment Setup

#### Transmitters

<table>
<thead>
<tr>
<th></th>
<th>WS1</th>
<th>WS2</th>
<th>WS3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OS</strong></td>
<td>Linux Fedora 20</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Chassis (metal)</strong></td>
<td>infinity chassis</td>
<td>GIGABYTE Setto 1020 GZ-AX2CBS</td>
<td>Silverstone RL04B</td>
</tr>
<tr>
<td><strong>CPU</strong></td>
<td>Intel i7-4790</td>
<td>Intel i7-3770</td>
<td>Intel i7-5820K</td>
</tr>
<tr>
<td><strong>Motherboard</strong></td>
<td>GIGABYTE GA-h87M-D3H</td>
<td>GIGABYTE H77-D3H</td>
<td>GIGABYTE GA-X99-UD4</td>
</tr>
<tr>
<td><strong>RAM Type</strong></td>
<td>2 x 4GB 1600MHz</td>
<td></td>
<td>4 x 4GB 2133MHz</td>
</tr>
<tr>
<td><strong>RAM Frequencies Tested</strong></td>
<td>1333/1600 MHz</td>
<td></td>
<td>1833/2133 MHz</td>
</tr>
<tr>
<td><strong>RAM Operation Modes Tested</strong></td>
<td>Single / Dual</td>
<td></td>
<td>Dual / Quad</td>
</tr>
</tbody>
</table>

#### Receivers

- USRP B210
- Motorola C123
Evaluation
Reception Distance

Amplitude ‘0’ vs ‘1’
Evaluation
Reception Distance

Delta between ‘0’ & ‘1’

[Graph showing the impact of frequency on reception distance]

XKCD

[Cartoon with a person holding a pineapple]
Evaluation
Reception Distance

Amplitude ‘0’ vs ‘1’
Evaluation
Signal to Noise Ratio (SNR)

SNR from the back of WS1 & WS2

Distance at which SNR = 0.5 dB
Evaluation
Bit Rates

<table>
<thead>
<tr>
<th>Data</th>
<th>Length(bit)</th>
<th>Rx Time</th>
<th>Rx Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC Address</td>
<td>48</td>
<td>30 sec</td>
<td>48 ms</td>
</tr>
<tr>
<td>Plain Password</td>
<td>64</td>
<td>40 sec</td>
<td>64 ms</td>
</tr>
<tr>
<td>MD5</td>
<td>128</td>
<td>1.3 sec</td>
<td>128 ms</td>
</tr>
<tr>
<td>GPS Coordinate</td>
<td>128</td>
<td>1.3 sec</td>
<td>128 ms</td>
</tr>
<tr>
<td>SHA1 Hash</td>
<td>160</td>
<td>1.6 min</td>
<td>160 ms</td>
</tr>
<tr>
<td>Disk Encryption Key</td>
<td>256</td>
<td>2.6 min</td>
<td>256 ms</td>
</tr>
<tr>
<td>RSA Private Key</td>
<td>2048</td>
<td>21.3 min</td>
<td>2.04 sec</td>
</tr>
<tr>
<td>Fingerprint Template</td>
<td>2800</td>
<td>29.1 min</td>
<td>2.8 sec</td>
</tr>
</tbody>
</table>

Bit Error Rate (BER)

Filters, FEC and other well known methodologies can improve the BER further!
Conclusion

Summary
• It’s feasible to get data out of an “Air-Gapped” network
• EMR from memory-bus can be exploited to transmit information
• Mobile devices can receive this information

Note:
• Some corporations allow simple GSM phones into restricted areas...
• Issue applies to: GSM, LTE,... bands
• GSMem is relevant to other scenarios as well
Thank you for listening!

Questions?