Remote Video Eavesdropping Using a Software Defined Radio Platform

Martin Marinov
St Edmund's College

UNIVERSITY OF CAMBRIDGE

A dissertation submitted to the University of Cambridge in partial fulfilment of the requirements for the degree of Master of Philosophy in Advanced Computer Science

University of Cambridge
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom

Email: mtm46@cam.ac.uk

May 17, 2014

Declaration

I Martin Marinov of St Edmund's College, being a candidate for the M.Phil in Advanced Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: ??,???

Signed:

Date:

Abstract

This dissertation presents a software toolkit for remotely eavesdropping video monitors using a Software Defined Radio (SDR) receiver. It exploits compromising emanations from cables carrying video signals. Analogue video is usually transmitted one line of pixels at a time encoded as a varying current. This generates a wideband electromagnetic wave that can be picked up by an SDR receiver. The presented software can map the received field strength of each pixel to a grayscale value in order to show a real-time false colour estimate of the original video signal.

The software significantly lowers the costs required for undertaking a practical attack compared to existing solutions. Furthermore, it allows for an additional digital post-processing which can aid in analysing and improving the results. It also provides mobility for a potential adversary, requiring only a commodity laptop and an USB SDR dongle. The attacker does not need to have any prior knowledge about the victim's video display. All parameters such as resolution and refresh rate can be estimated with the aid of the software.

The software comprises of a library written in C, a collection of plug-ins for various Software Define Radio (SDR) frontends and a Java based Graphical User Interface (GUI). It is designed to be a multi-platform application. All native libraries can be pre-compiled and packed into a single Java jar file which allows the toolkit to run on any supported operating system.

This report documents the digital processing techniques that have been employed in order to extract, detect and lock to a video signal. It also explains the architecture of the software system and the techniques used in order to achieve low latency and real-time interactivity. It demonstrates the usage of the system by performing a practical attack. It then gives some ideas about what could be improved further and some analysis of data that was collected during the development of the software.

Contents

1 Introduction **1**
1.1 Related Work 1
1.2 Availability 1

2 Background **3**
2.1 Electronic Emanations 3
2.2 IQ sampling 3

3 Methodology **5**
3.1 Sampling Rate 5

4 Practical attack **7**
4.1 The Setup 7
4.2 Preparation 8

5 Implementation **9**
5.1 Hardware 10
5.2 Architecture 10
5.2.1 Main Library 10
5.2.2 JavaGUI 10
5.3 Digital Signal Processing 10
5.3.1 Overview 10
5.3.2 Signal Reconstruction 10
5.3.3 Frame Synchronization 10
5.3.4 Resolution Detection 10
5.3.5 Extended Bandwidth 10
5.3.6 Testing and Benchmarking 10

6 Summary and Conclusions **11**

Chapter 1

Introduction

1.1 Related Work

[Related work and critique]

1.2 Availability

[Github repo url]

[1]

Chapter 2

Background

2.1 Electronic Emanations

2.2 IQ sampling

Chapter 3

Methodology

[Explain how the whole system works in high level]

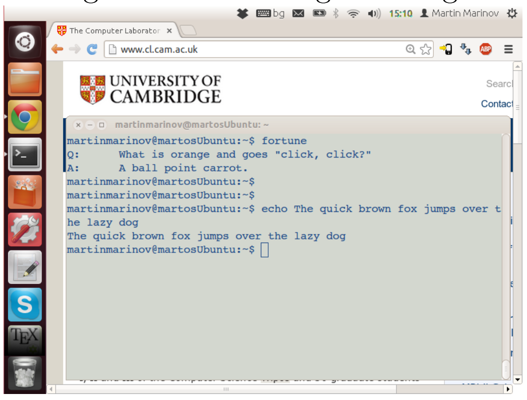
[TODO! Explain how resolution and framerate detection works]

3.1 Sampling Rate

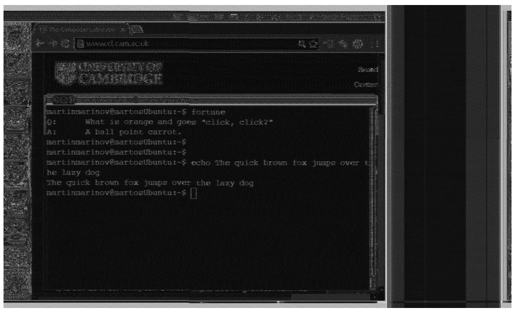
[TODO! Explain how bandwidth relates to resolution width * height * framerate = samplerate]

The sampling rate

Figure 3.1: The original image

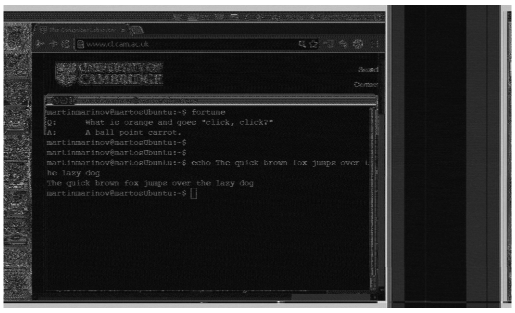


```
martinmarinov@martosUbuntu:~$ fortune
Q:      What is orange and goes "click, click?"
A:      A ball point carrot.
martinmarinov@martosUbuntu:~$
martinmarinov@martosUbuntu:~$
martinmarinov@martosUbuntu:~$ echo The quick brown fox jumps over t
he lazy dog
The quick brown fox jumps over the lazy dog
martinmarinov@martosUbuntu:~$ 
```



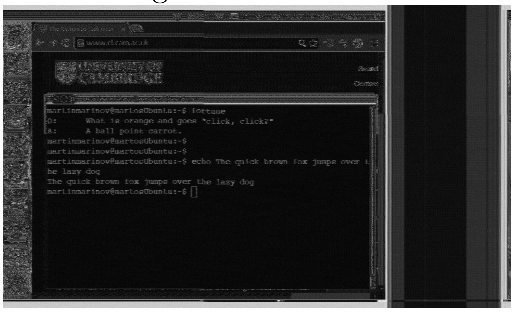
```
martinmarinov@martosUbuntu:~$ fortune
Q:      What is orange and goes "click, click?"
A:      A ball point carrot.
martinmarinov@martosUbuntu:~$
martinmarinov@martosUbuntu:~$
martinmarinov@martosUbuntu:~$ echo The quick brown fox jumps over t
he lazy dog
The quick brown fox jumps over the lazy dog
martinmarinov@martosUbuntu:~$ []
```

Figure 3.2: 50 MHz



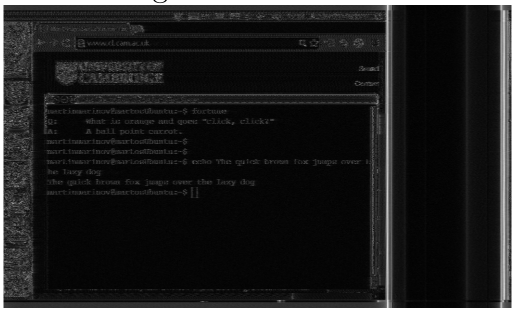
```
martinmarinov@martosUbuntu:~$ fortune
Q:      What is orange and goes "click, click?"
A:      A ball point carrot.
martinmarinov@martosUbuntu:~$
martinmarinov@martosUbuntu:~$
martinmarinov@martosUbuntu:~$ echo The quick brown fox jumps over t
he lazy dog
The quick brown fox jumps over the lazy dog
martinmarinov@martosUbuntu:~$ []
```

Figure 3.3: 40 MHz



```
martinmarinov@martosUbuntu:~$ fortune
Q:      What is orange and goes "click, click?"
A:      A ball point carrot.
martinmarinov@martosUbuntu:~$
martinmarinov@martosUbuntu:~$
martinmarinov@martosUbuntu:~$ echo The quick brown fox jumps over t
he lazy dog
The quick brown fox jumps over the lazy dog
martinmarinov@martosUbuntu:~$ []
```

Figure 3.4: 30 MHz



```
martinmarinov@martosUbuntu:~$ fortune
Q:      What is orange and goes "click, click?"
A:      A ball point carrot.
martinmarinov@martosUbuntu:~$
martinmarinov@martosUbuntu:~$
martinmarinov@martosUbuntu:~$ echo The quick brown fox jumps over t
he lazy dog
The quick brown fox jumps over the lazy dog
martinmarinov@martosUbuntu:~$ []
```

Figure 3.5: 20 MHz

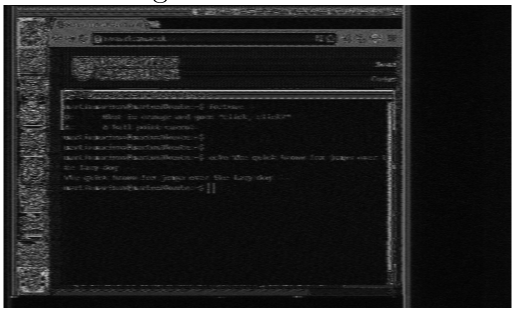


Figure 3.6: 10 MHz

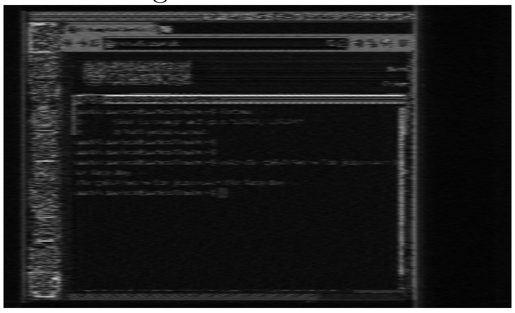


Figure 3.7: 5 MHz

Chapter 4

Practical attack

Before explaining how the software works internally, let's present a demonstration of a practical video eavesdropping attack. Its main aim is to show the ease with which such an attack could happen. In the meantime this will give an opportunity to explain the characteristics of the received signal and how they are exploited.

As in the real world, we will start with no knowledge of the victim's system. We will estimate the frequency at which the emission strength has the best Signal to Noise Ratio (SNR). We will then analyse the signal to detect the resolution and refresh rate of the screen. We will afterwards lock onto the signal and try to recover the original video. We will also discuss some techniques that could be utilized to improve the quality of the image.

4.1 The Setup

The choice for a SDR front-end for this demonstration is a USRP B200[1]. Depending on the particular requirements, an attacker might prefer mobility over accuracy and choose the smaller FlexiTV^{TM}MSi3101 SDR USB Dongle.

[1]Refer to 5.1 Hardware for discussion on currently available SDR devices.

However, for this demonstration we will attempt to obtain the highest possible resolution. Therefore we need an SDR radio that is capable of obtaining a wide bandwidth. This makes the 32 MHz of Bandwidth that the USRP provides a much better choice than the 8 MHz available from the Mirics dongle.

[Antenna]

[Location of victim]

[Victim's fonts]

4.2 Preparation

Chapter 5

Implementation

5.1 Hardware

5.2 Architecture

5.2.1 Main Library

5.2.2 JavaGUI

Multiplatform

Graphical Interface

Visualisation

5.3 Digital Signal Processing

5.3.1 Overview

5.3.2 Signal Reconstruction

Multithreading

Demodulation

Re-sampling

Chapter 6

Summary and Conclusions

[Summarize results]

[potential future work]

Bibliography

[1] Markus G Kuhn. Compromising emanations: eavesdropping risks of computer displays. *University of Cambridge Computer Laboratory, Technical Report, UCAM-CL-TR-577*, 2003.