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 19, 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

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

3.1 Anatomy of a Video Signal

Almost all contemporary video monitors are raster based. The image is transferred from the video controller to the display in scanlines that occur at a specific rate. Each of these scanlines contains a number of pixels which are continuously encoded as a time varying signal. This signal is generated internally in the video controller by an oscillator which runs at the pixel clock rate. The signal is thereafter multiplied by the pixel intensities at the specific time.

Let's assume that the signal started transmitting at time $t = 0$, and each video frame contains h scanlines, each of which contains w pixels. The frequency with which frames are being generated is f frames per second. Therefore the duration of transmission of each individual pixel is

$$T_\Delta = \frac{1}{whf}$$

Furthermore at time t frame number n_t started transmission where

$$n_t = \lfloor tf \rfloor$$

Therefore if we assume that the top left corner of a frame has coordinates $(0, 0)$, a pixel at position (x, y) in frame n_t will start to be transmitted at time

$$T_{(x,y)} = (yw + x + n_t wh)T_\Delta$$

and will finish transmission before time

$$T_{(x+1,y)} = T_{(x,y)} + T_\Delta = (yw + x + 1 + n_t wh)T_\Delta$$

at which the next pixel will start transmitting.

In practice w and h are determined by the screen resolution and f is simply the screen refresh rate. For a typical screen resolution of $width \times height$, it is true that $w \geq width$ and $h \geq height$. The reason is that video signals tend to have additional synchronisation regions. This means more pixels are transmitted than what is in the active video region. This gives opportunity for the receiving monitor to synchronise its internal clock, calibrate its colour levels or in case of CRT, allow enough time for the electron beam to return to the beginning of the next line on the screen. The synchronisation timings for personal computers have been standardised by Video Electronics Standards Association (VESA) [2].

In order to decode an individual pixel, the receiving monitor has its own internal oscillator. It locks it to the pixel rate of the incoming signal either via an external clock source or using the synchronization regions. Once it receives the signal for an individual pixel, the monitor will usually average it over the time from $T_{x,y}$ to $T_{x,y} + T_\Delta$. The amplitude of this average relates to pixel intensity. This allows the monitor to display the video in real time. If multiple colours are desired, the intensity of each of them (or some other combination) will be transmitted separately on different wires.

3.2 Eavesdropping

As we already know, a changing signal in a wire produces an electromagnetic field. Therefore the changing intensities of the pixels in the wires will produce electromagnetic waves. Therefore we can model the signal as a band limited signal with a maximum frequency component of $\nu_{max} = \frac{1}{T_\Delta}$. This signal will contain the full information that is required to reconstruct the transmitted image. In practice, the signal contains higher frequencies. This is due to the fact that the signal for each pixel is not smooth [1]. However, the extra frequencies could be eliminated with a low pass filter which can produce the ideal band limited signal. Therefore for the rest of the discussion, I will assume that the video signal is indeed limited to.

[Explain how the whole system works in high level]

[TODO! Explain how resolution and framerate detection works]

3.3 Sampling Rate

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

The sampling rate

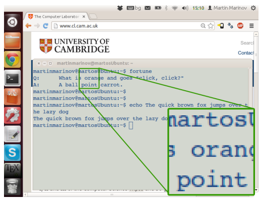
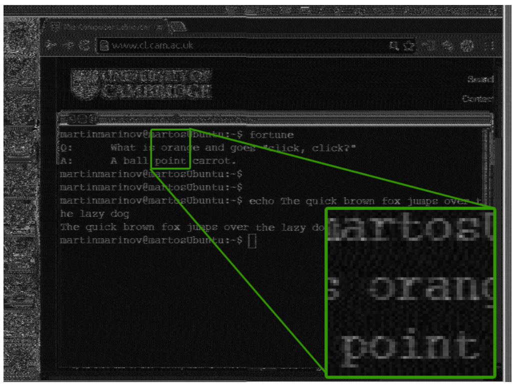
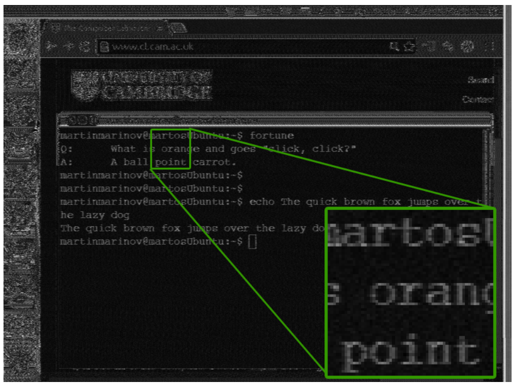


Figure 3.1: Transmitted image



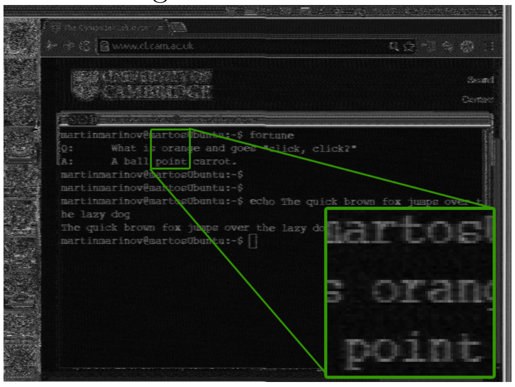
```
martinmarinov@martosUbuntu:~$ fortune
Q:      What is orange and goes "click, click?"
A:      A ball point carrot.
martinmarinov@martosUbuntu:~$
martinmarinov@martosUbuntu:~$
martinmarinov@martosUbuntu:~$ echo The quick brown fox jumps over t
he lazy dog
The quick brown fox jumps over the lazy do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

Figure 3.2: 50 MHz



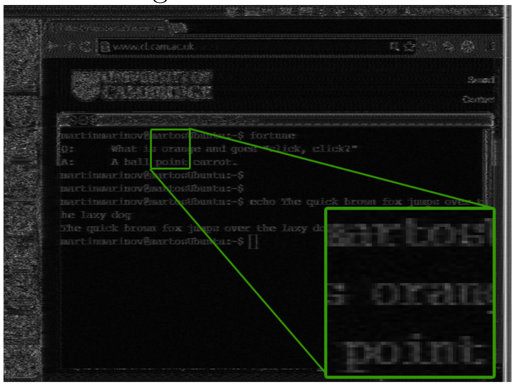
```
martinmarinov@martosUbuntu:~$ fortune
Q:      What is orange and goes "click, click?"
A:      A ball point carrot.
martinmarinov@martosUbuntu:~$
martinmarinov@martosUbuntu:~$
martinmarinov@martosUbuntu:~$ echo The quick brown fox jumps over
he lazy dog
The quick brown fox jumps over the lazy do
martinmarinov@martosUbuntu:~$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

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
he lazy dog
The quick brown fox jumps over the lazy do
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
he lazy dog
The quick brown fox jumps over the lazy do
martinmarinov@martosUbuntu:~$
```

Figure 3.5: 20 MHz

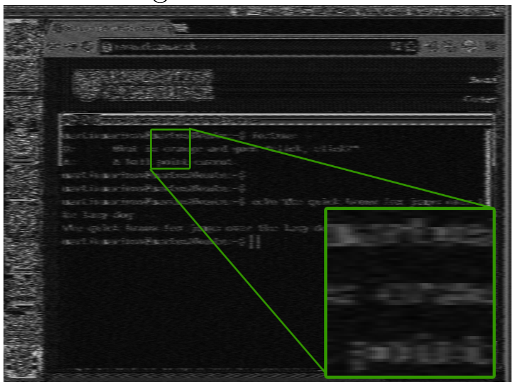


Figure 3.6: 10 MHz

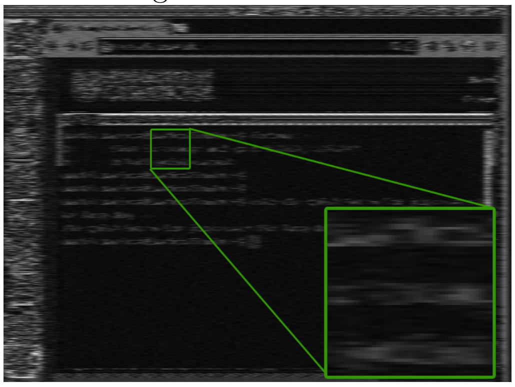


Figure 3.7: 5 MHz

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 Mirics FlexiTV$^{\text{TM}}$MSi3101 SDR USB

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

Dongle. 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

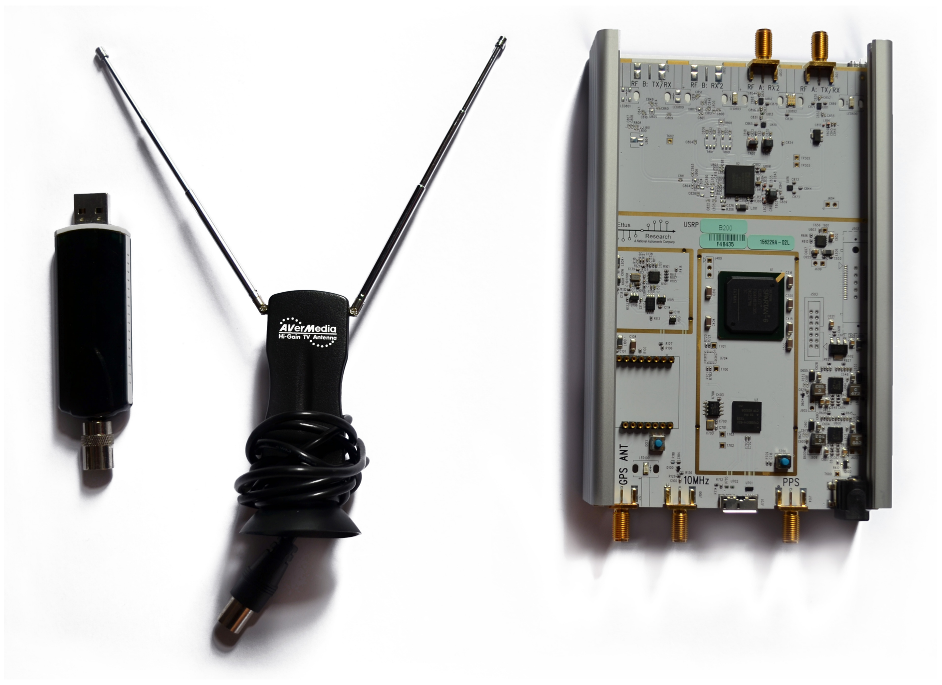


Figure 5.1: From left to right: MSi3101, AverMedia antenna and USRP B200

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

Auto Gain

Low-pass

5.3.3 Frame Synchronization

5.3.4 Resolution Detection

5.3.5 Extended Bandwidth

5.3.6 Testing and Benchmarking

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.

[2] Monitor Timing Specifications. Version 1.0. *Revision 0.8, Video Electronics Standards Association (VESA), San Jose, California*, 1998.