RAMSES StegAware

 This work is funded by the Horizon 2020 RAMSES project.

1. Description

This forensic and steganalytic framework provides comprehensive analysis of image and video media and is part of the RAMSES platform. StegAware is a steganalysis tool to detect the presence of steganography over multimedia content (image and video). StegAware is capable of detecting a diverse range of steganographic tools and techniques. The image steganalysis component uses StegExpose for LSB steganalysis alongside a series of signature detection methods. The video steganalysis component uses signature steganalysis and generalised detection methods to identify steganography across a range of wellknown video steganography tools. In addition to this, we have added a feature for metadata forensics and have added functionality to integrate StegAware with the RAMSES platform.

Quick Start Guide

The following steps can be used for a quick setup of the tool. A more detailed breakdown of the prerequisites are discussed in 1.2:

1.0 $cd Desktop
1.1 $git clone github.com/UoK424/Ramses_StegAware
1.2 Rename folder to “Ramses”
1.3 $sudo apt install libimage-exiftool-perl
1.4 $sudo apt install mp4v2-utils
1.5 $sudo apt install openjdk-11-jre-headless (or latest version)
1.6 $sudo apt install libdigest-sha3-perl

1.2 Prerequisites

The tool is available as open-source on GitHub from the link below. StegAware can be either cloned or downloaded as a zip file and should be installed onto the user Desktop and renamed “Ramses”.

https://github.com/UoK424/RAMSES_StegAware

ExifTool – The metadata forensics feature of this tool makes use of ‘ExifTool’. This is a free, open-source program for reading, writing, and manipulating file metadata. The RAMSES steganalytic tool uses these features to extract relevant metadata from video and image files. This can be installed from the following command:

```
$sudo apt install libimage-exiftool-perl
```

Python – The reporting system is generated via python 3.X. This should be installed by default on Ubuntu distributions.

MP4Reader – Used in the OpenPuff detection script for mp4 metadata extraction

```
$sudo apt install mp4v2-utils
```

Java – The StegExpose feature of StegAware is a Java based tool. This installation will be required before Seek can be run. There are multiple versions that can be used. For example:

```
$sudo apt install openjdk-11-jre-headless
```

Sha3 – StegAware uses Sha3 hashes to assign a unique ID to each file that is uploaded to the RAMSES platform. The use of hashes also provide validation of the files authenticity to the user. This can be installed through the following command:

```
$sudo apt install libdigest-sha3-perl
```

File Paths – StegAware runs through a series of Bash and Python scripts. Because of this, we use relative paths to minimise setup time. The folder containing StegAware should always be placed on the user Desktop. If the user instead wishes to place the StegAware tool in a different directory/subdirectory, file paths can be updated across each script with the following command:

```
$ Find . -name "*.sh" -exec sed -I "s/OldWord/NewWord/g" '{}' \;
```

Dataset Path – StegAware processes all files in a given directory. By default this is set to:

```
/home/user/Desktop/Ramses/TestMedia/*Images* or *Video*
```

However, this can be changed in the `Ramses.sh` source script.

In addition, for any images to be processed by StegExpose, '/StegExpose/StegExpose.sh/' subscript. For example,

```
$java -jar StegExpose.jar /home/user/Desktop/TestImages >>
/home/user/Desktop/Ramses/Results/ImageAnalysis.txt
```

2. Usage

StegAware is a command line tool and is run from the Linux terminal. The user should navigate to the Ramses directory and run the source script from Python.

```
$ sudo python3 stegtool.py
```

The user will have access to the two main features of the tool. Either the user can run the steganalytic tests and push the results automatically to the RAMSES platform, or alternatively, the user can manage the results exclusively without running any steganalytic tests. This is shown in Figure 1.

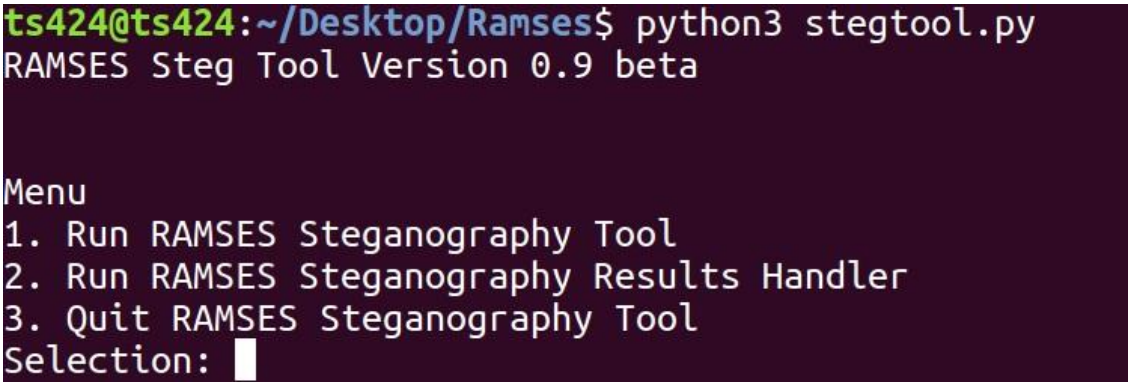

Figure 1- Welcome screen for StegAware

2.1 Option 1 – RAMSES Steganography Tool

The main function of the first option is to run steganalytic tests and manage any obtained results. From this, the user can decide to either push results to the RAMSES platform, or run the tests locally to upload at later date. This is shown in Figure 2.

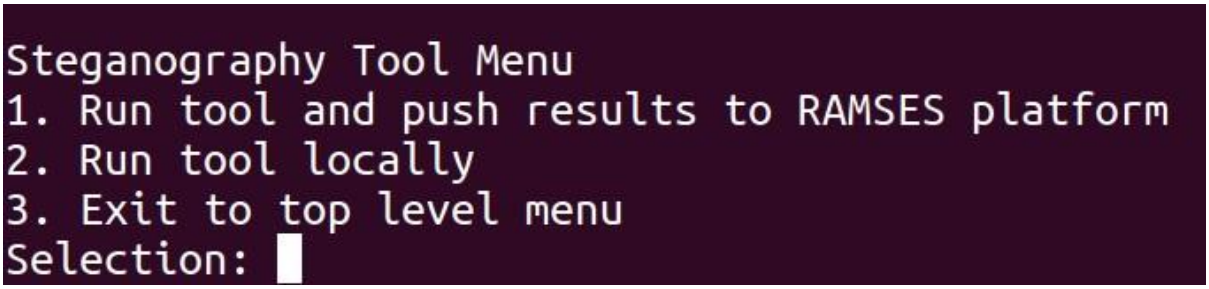

Figure 2 - Option for how to run tests

Before running any tests, you will need to give a name to your malware investigation and determine a privacy level for any results you generate. You can choose between **public, private, and agency**. This will determine who has access to any data that you upload.

Once you have gone through the initial setup, you can select a series of tests that best suits your investigation. You can run either video or image steganalysis tests, or perform forensic metadata analysis (as shown in Figure 3).

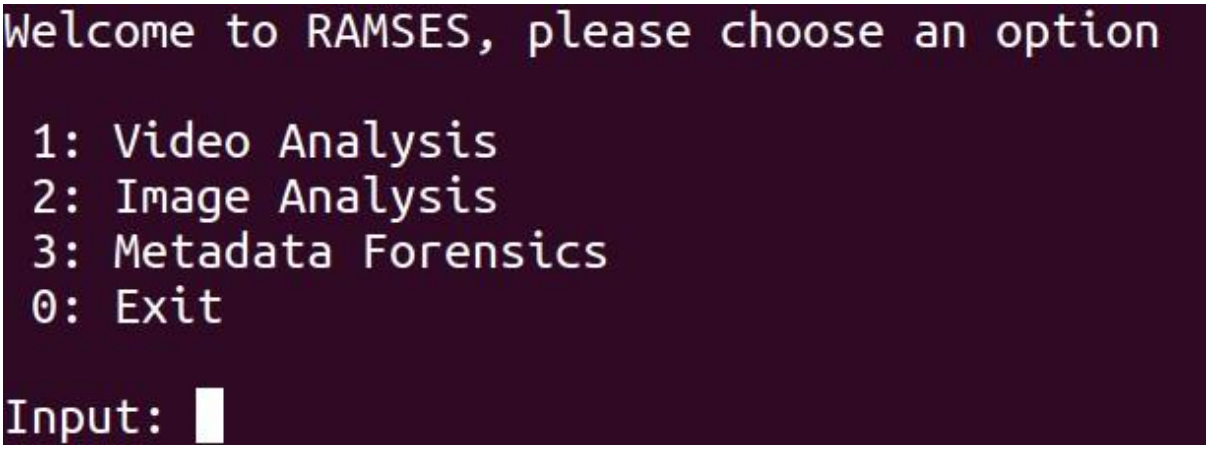

Figure 3 - Test options for StegAware

Upon selecting the first option (video analysis), you can either run each detection script individually (2-6) or alternatively, select the first option (1) to run all scripts automatically, as shown in Figure 4. Individual scripts are best used when the user knows that they are looking for a specific type of steganography/data embedding.

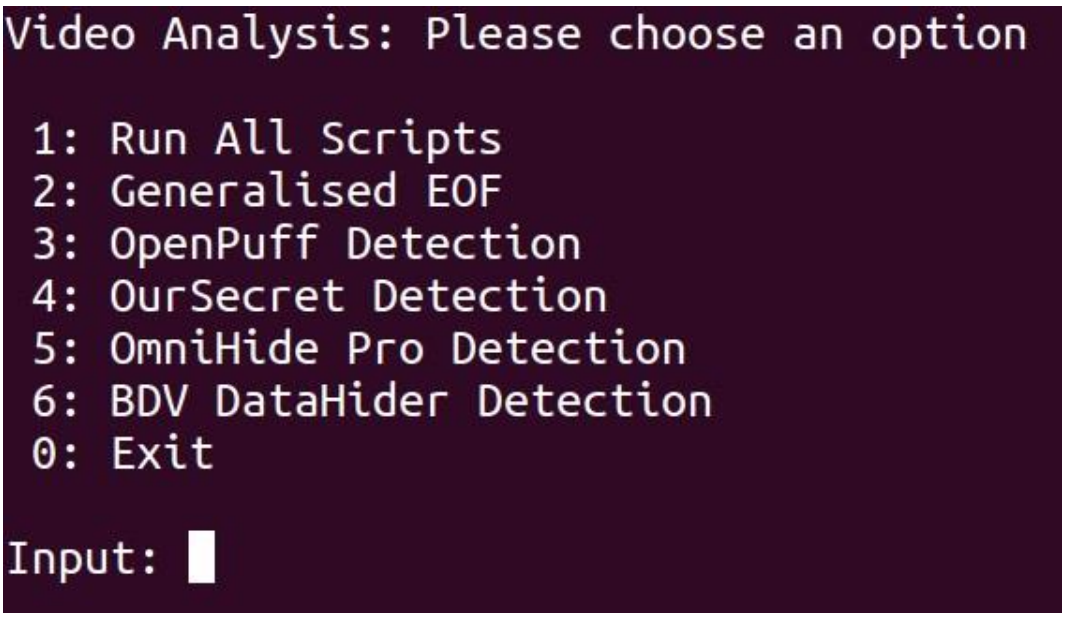

Figure 4 - Options for video analysis

If you need to perform steganalysis over image content as opposed to video, the 'Image Analysis' feature can provide multiple detection methods. The StegExpose option will provide a generalised method for LSB detection using statistical steganalysis via fusion steganalysis. Alternatively, you can perform signature detection based upon Pixelknot image steganography.

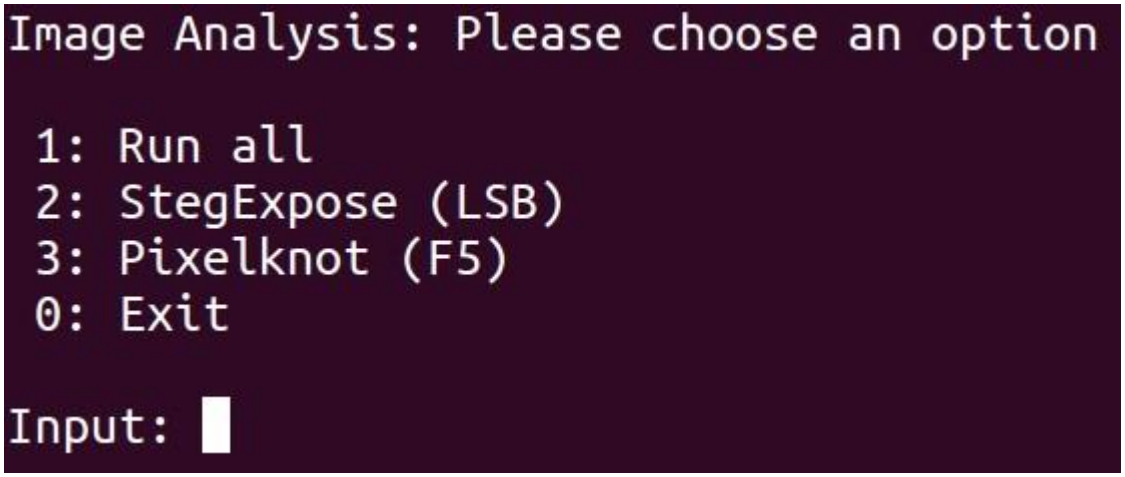

Figure 5 - Options for image analysis

Forensic metadata analysis will be performed automatically after both Image and Video analysis tests. However, it can also be run separately. After any series of tests has been run, results will either be automatically uploaded to the RAMSES platform or stored locally, depending on the user's preference.

2.2 Option 2 – Results Handler

Returning back to the main menu of StegAware (initially shown in Figure 1), the second option will allow the user to manage data and results without the need to interact with any steganalytic tools. This will allow the user to either import results from the RAMSES platform, submit their own results, or delete any results that they have ownership of. This option is shown in Figure 6.

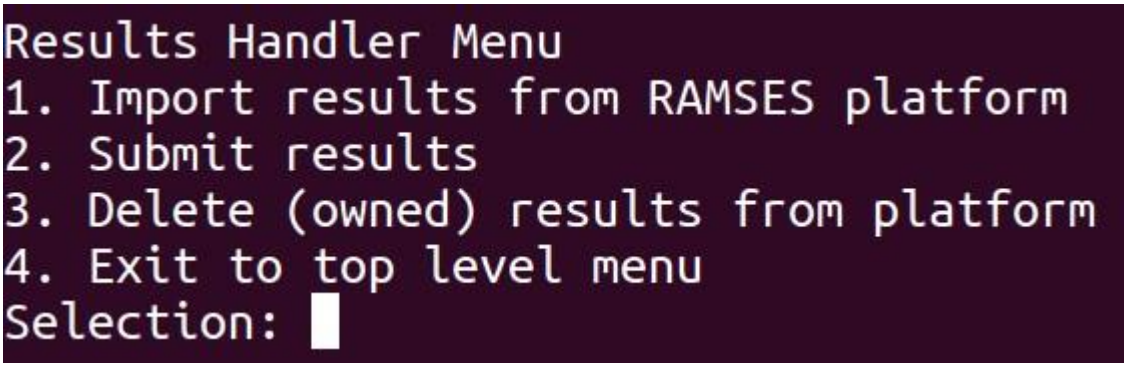

Figure 6 - Results handler options

3. Results

Any results obtained will be written to /Ramses/Results/ if they have been stored locally. In this subdirectory, a file is given that merges the results of both the steganalytic tests and the forensic metadata tests. The user will be given comprehensive details for each file scanned as shown in Figure 7

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	File Name	File Size	Image Size	File Type Extension	File Access Date	File Modify Date	Duration	SHA3 512 Hash	Date	UID	Steg_Algorithm	Steg_Present	Steg_Signature
2	MP4 (1).MP4	96 MB	1920x1080	mp4	2018:08:24 14:0	2015:07:17 05:45	0:00:31	4a3ddd07205e18823	2018:08:24 14:03:51	954084f7ba5295e9	None	None	None
3	MP4 (10).mp4	118 MB	320x240	mp4	2018:08:24 14:0	2014:01:22 13:06	0:28:21	f06a988c3cf2175e3d	2018:08:24 14:03:51	b6517412cfa3415a	None	None	None
4	MP4 (11).mp4	108 MB	320x240	mp4	2018:08:24 14:0	2014:01:27 11:52	0:26:41	57dd569e65d5d9e42	2018:08:24 14:03:51	a223b8dbc33a189	None	None	None
5	MP4 (12).mp4	108 MB	320x240	mp4	2018:08:24 14:0	2014:01:23 16:18	0:26:48	4eace6202db71345b	2018:08:24 14:03:51	b4225346a7a0cf0	None	None	None
6	MP4 (13).mp4	109 MB	320x240	mp4	2018:08:24 14:0	2014:01:23 11:37	0:27:02	bffb7383efdb3a5116	2018:08:24 14:03:51	09d7d877eba8269	None	None	None
7	MP4 (14).mp4	109 MB	320x240	mp4	2018:08:24 14:0	2014:01:22 13:16	0:27:02	954b9e55a99dc50a8	2018:08:24 14:03:51	3c5b1b1b47e35d5	None	None	None
8	MP4 (15).mp4	116 MB	320x240	mp4	2018:08:24 14:0	2014:01:22 15:16	0:28:47	da88450d264cf4223	2018:08:24 14:03:51	a06b8c44ab1d09e	None	None	None
9	MP4 (16).mp4	115 MB	320x240	mp4	2018:08:24 14:0	2014:01:23 16:33	0:28:31	7ea06045ffb0c1e8b3	2018:08:24 14:03:51	4145e7118ed6dbe	None	None	None
10	MP4 (17).mp4	118 MB	320x240	mp4	2018:08:24 14:0	2014:01:23 15:52	0:29:09	cacb59f238984bd54	2018:08:24 14:03:51	21cb19d31552da0	None	None	None
11	MP4 (18).mp4	54 MB	320x240	mp4	2018:08:24 14:0	2014:01:23 13:38	0:13:34	a3928936b79abb68c	2018:08:24 14:03:51	00f5222f97407562	None	None	None
12	MP4 (19).mp4	62 MB	320x240	mp4	2018:08:24 14:0	2014:01:23 13:54	0:15:29	88452da3ae14cc5ec	2018:08:24 14:03:51	31b66ed53ab01f7	None	None	None
13	MP4 (2).mp4	201 MB	320x240	mp4	2018:08:24 14:0	2014:01:23 13:51	0:49:46	7fb964e8b36248917	2018:08:24 14:03:51	3626bf6ba3f79df7	None	None	None
14	MP4 (20).mp4	47 MB	320x240	mp4	2018:08:24 14:0	2014:01:23 13:33	0:11:43	8a23d48193255f9b3a	2018:08:24 14:03:51	df2b03d0028391e	None	None	None
15	MP4 (21).mp4	181 MB	320x240	mp4	2018:08:24 14:0	2016:12:20 18:17	0:45:00	a49444108853f8b210	2018:08:24 14:03:51	b74e4e374020141	EOF Steganography	yes	EOF Injection
16	MP4 (22).mp4	92 MB	320x240	mp4	2018:08:24 14:0	2016:12:20 18:18	0:22:13	443bb57d5477c0e84	2018:08:24 14:03:51	293c692e5572d00	EOF Steganography	yes	EOF Injection
17	MP4 (23).mp4	235 MB	320x240	mp4	2018:08:24 14:0	2016:12:20 18:19	0:28:23	c44fc2cc894337dbe1	2018:08:24 14:03:51	611ecc037885253	EOF Steganography	yes	EOF Injection
18	MP4 (24).mp4	122 MB	320x240	mp4	2018:08:24 14:0	2016:12:20 18:20	0:29:16	a1f32bad0a9e44a70a	2018:08:24 14:03:51	033f427d8e41e8e4	EOF Steganography	yes	EOF Injection
19	MP4 (25).mp4	116 MB	320x240	mp4	2018:08:24 14:0	2016:12:20 18:20	0:28:49	4c3d8d8bb8eb9417b	2018:08:24 14:03:51	431ed372f04c296	EOF Steganography	yes	EOF Injection
20	MP4 (26).mp4	223 MB	320x240	mp4	2018:08:24 14:0	2016:12:20 18:22	0:27:39	c315a1863812bff88f	2018:08:24 14:03:51	79ab694e3b7d0cf	EOF Steganography	yes	EOF Injection

Figure 7 - Locally stored results

Alternatively, if the results have been uploaded to the RAMSES platform, they can be accessed by logging in to the platform through your browser. Here you will have various filtering options to easily identify the data that you are looking for (as shown in Figure 8).

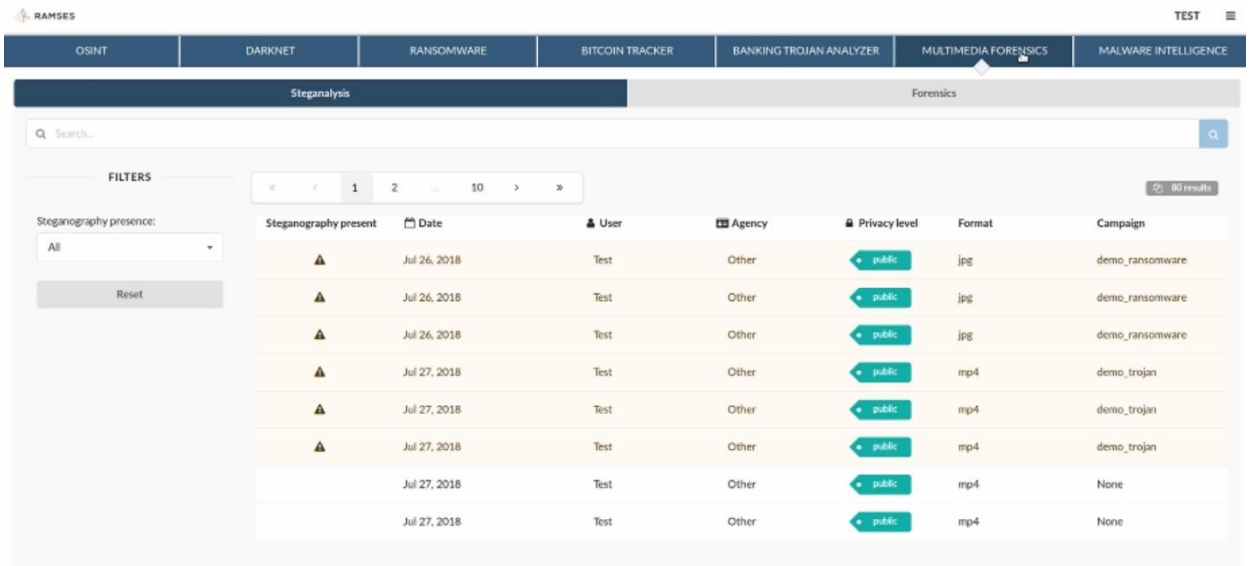

Figure 8 - Results uploaded to platform

4. Overview of Video Steganalysis Scripts

The video steganalysis feature of this tool is built upon a series of detection scripts. Each detection method performs signature analysis over MP4 files and for each case, we have identified and implemented highly accurate signatures to detect the presence of steganography and trace its use to a specific tool. This tracing feature can provide useful insights for forensic investigators and practitioners. The following video steganography tools and techniques can be detected through the video steganalysis features of StegAware.

4.1 OpenPuff Detection

OpenPuff steganography is performed in the metadata channel of a video file. Flags, a component of atoms that describe the structure of an MP4 file are typically null values. These values are modified by the OpenPuff embedding algorithm to hide part of the secret information [1], as shown in Figure 9.

Decoding Time to Sample Box	
Start offset	540 (0X0000021C)
Box size	24 (0X00000018)
Box type	stts (0X73747473)
Version	0 (0X00000000)
Flags	0 (0X00000000)

Decoding Time to Sample Box	
Start offset	540 (0X0000021C)
Box size	24 (0X00000018)
Box type	stts (0X73747473)
Version	0 (0X00000000)
Flags	139266 (0X00022002)

Figure 9 – Null flag (left) and modified flag (right)

Flags appear in many atoms throughout the MP4 file structure. However, the secret data embedded into these fields are encrypted and therefore, pseudorandom. Because of this, it is difficult to construct a consistent signature. Instead, a detection algorithm can be constructed by looking at flag fields and observing a ‘null’ or ‘modified’ value. A count can then be introduced to determine with high accuracy if a file has been modified by OpenPuff steganography. A result for this may appear as:

$ /filepath/filename - OpenPuff Steganography Detected! 30 5

The numbers in this example refer to the count of positive flags identified vs null flags. A higher weight of positive flag modifications will assume the presence of OpenPuff steganography with a higher rate of accuracy. The functionality of the OpenPuff program is described in Figure 10.

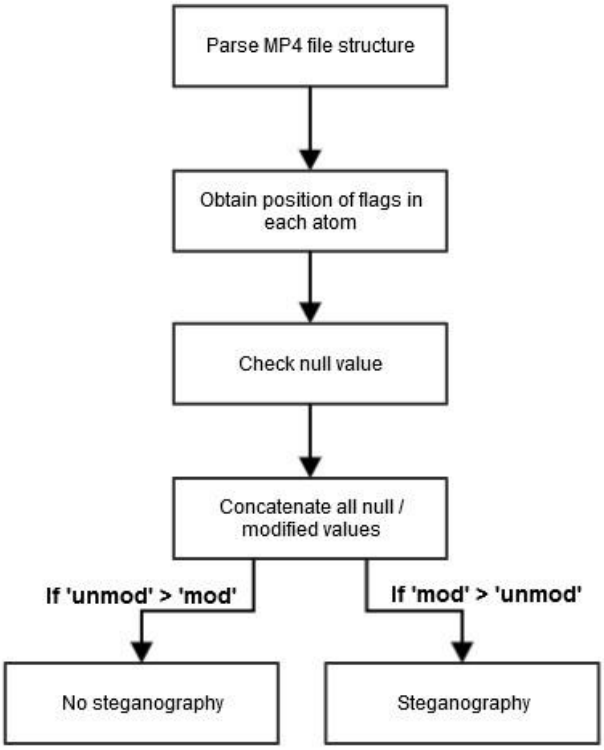

Figure 10 – Functionality of OpenPuff detection script

4.2 OurSecret, BDV DataHider, and OmniHide Pro

OurSecret, BDV Datahider, and OmniHide Pro all use EOF (end of file) steganography to hide a secret message. This is achieved by creating empty space at the end of a video file and filling it with secret message data. This method is often used because it is simple to implement and does not affect the image stream or audio stream of a video file. In each case, a unique signature can be identified among the embedded data that proves the existence of steganography and traces the embedded data to the tool that performed the steganographic function.

Detection is performed by simply looking for each known signature in any given video file. This is shown in Figure 11.

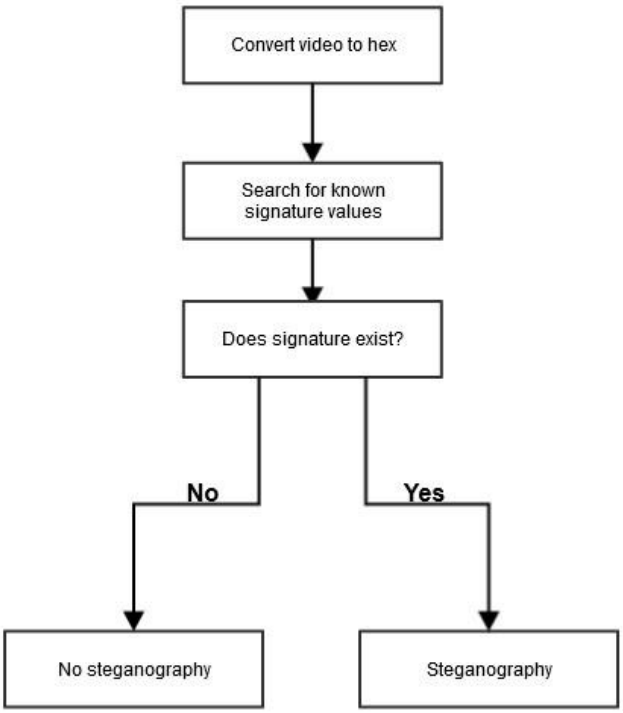

Figure 11 – Functionality of signature detection scripts

4.3 Steganosaurus Detection

Steganosaurus is video steganography tool that uses complex methods for hiding secret information. Data is embedded by motion vector (MV) steganography. This method takes the first ‘x’ or ‘y’ coordinate of an MV for a given macroblock and embeds within the LSB component of the target MV. A macroblock is an 8x8 block of pixels in any given frame. The functionality of the Steganosaurus detection script is shown through Figure 12.

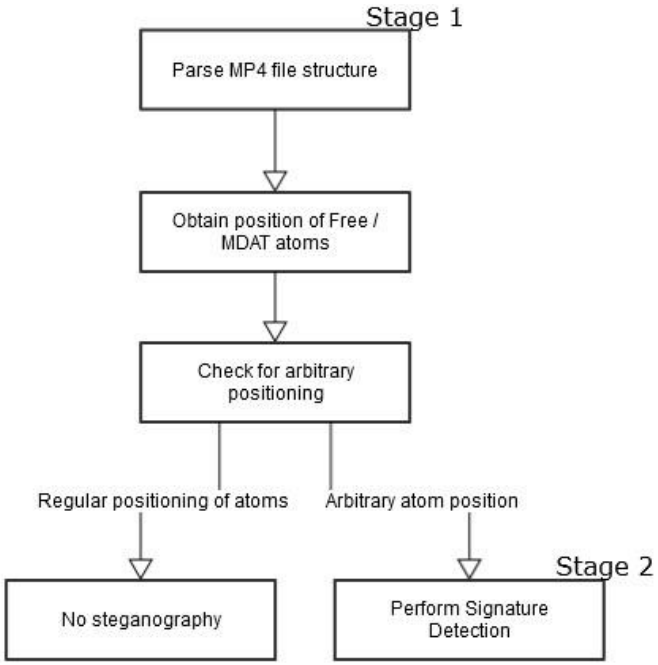

Figure 12 – Functionality of Steganosaurus detection script

However, the MV feature of Steganosaurus requires a cover-object to be modified by the compression algorithm to make necessary changes to a target MV. The compressor used by Steganosaurus makes predictable changes that can be identified in a stego-object and used for the purposes of steganalysis. Specifically, a consistent signature can be identified and extracted for use in a detection script.

4.4 Generalised EOF Detection

This feature of the SEEK framework provides multiple uses. Firstly, it can be a generalised method to quickly detect the known EOF steganography tools without giving any tool tracing. Secondly, it can be useful for detection new steganography tools that implement EOF injection that we may not have discovered. Finally, it is significantly useful when running all scripts automatically to reduce the sample size and reduce run time.
Video steganography tools that perform EOF injection modify the structure of MP4 files. The newly hidden data appears as an arbitrary atom that can be detected by error parsing as it won't match the typical features of an atom. The functionality of this script is shown in Figure 13.

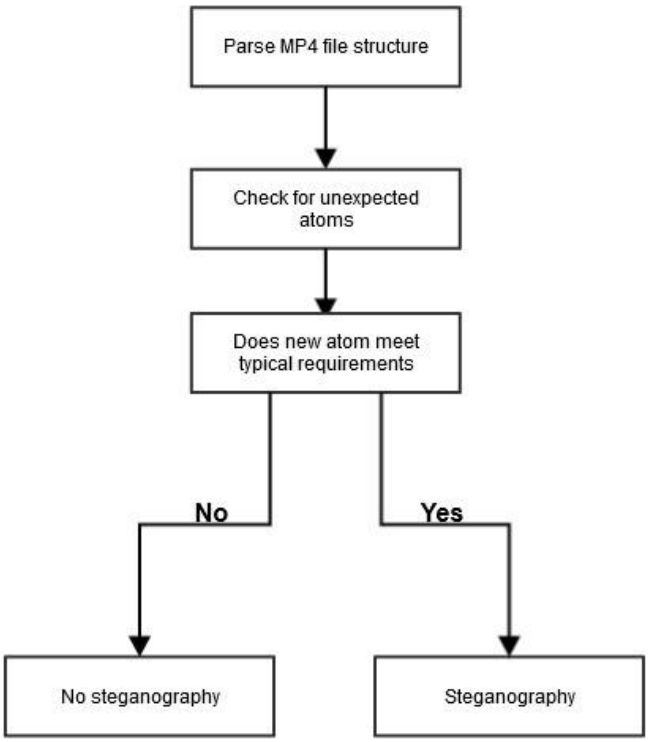

Figure 13 – Functionality of generalised EOF detection script

4.5 Executing Video Scripts

Each script for video steganography can be run individually if the user is looking to detect a particular steganography tool. Alternatively, the user can select to run all scripts automatically. This runs every video steganalysis script using a method designed to be efficient and highly accurate.

To begin, the two fastest scripts (OpenPuff and generalised EOF) will run to significantly reduce the sample size. Any detection from the generalised EOF will subsequently continue to run signature detection for each tool. Signature detection runs a full analysis of the file and will be more time-consuming, so it is ideal that this method runs on a smaller sample size reduced by EOF detection. Figures 14 & 15 show the sequence of the automated script and the results of each test.

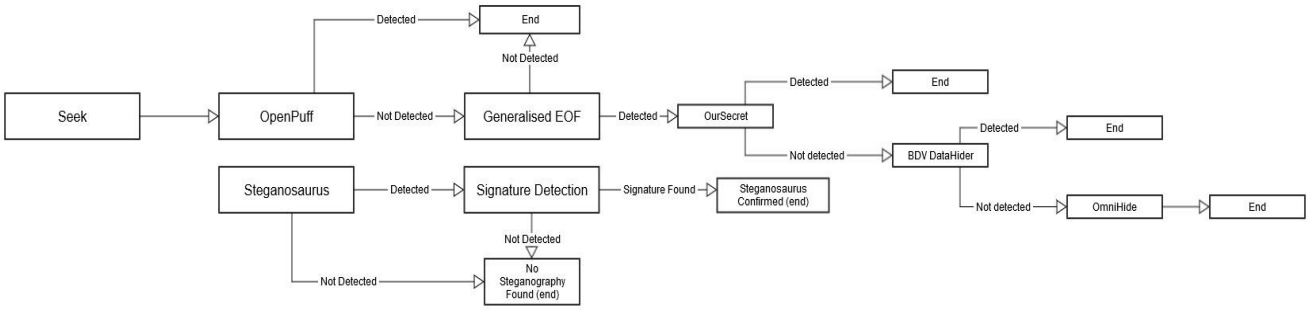

Figure 14 – Order sequence for running all scripts automatically

Steganalysis Scheme	Tests	False Positives	Average Runtime	Total Runtime
OpenPuff	1000	0%	0.056s	56.66
EOF	1000	0%	0.017s	17.08s
OurSecret	1000	0%	1.95s	32.5m
BDV DataHider	1000	0%	3.6s	60m
OmniHide Pro	1000	0%	2.35s	39.26m
Steganosaurus	1000	0%	0.034s	34.33s

Figure 15 – Results for each detection script

5. Overview of Image Steganalysis Scripts

The image steganalysis component of this framework has two unique detection methods. The first is StegExpose, the second is Pixelknot detection.

5.1 StegExpose

StegExpose [3] is a command line tool specialised in detecting LSB (Least Significant Bit) [4] steganography through a series of highly accurate detection methods, including: RS-Analysis (Regular and Sample Group) [5], SPA (Sample Pair Analysis) [6], Chi-Square [7], and Primary Sets. This allows for detection across a significantly large number of image steganography tools that use the LSB method and estimate given size of the embedded message. However, it will not trace the use of steganography to any particular tool. The performance for this tool is shown in Figure 16.

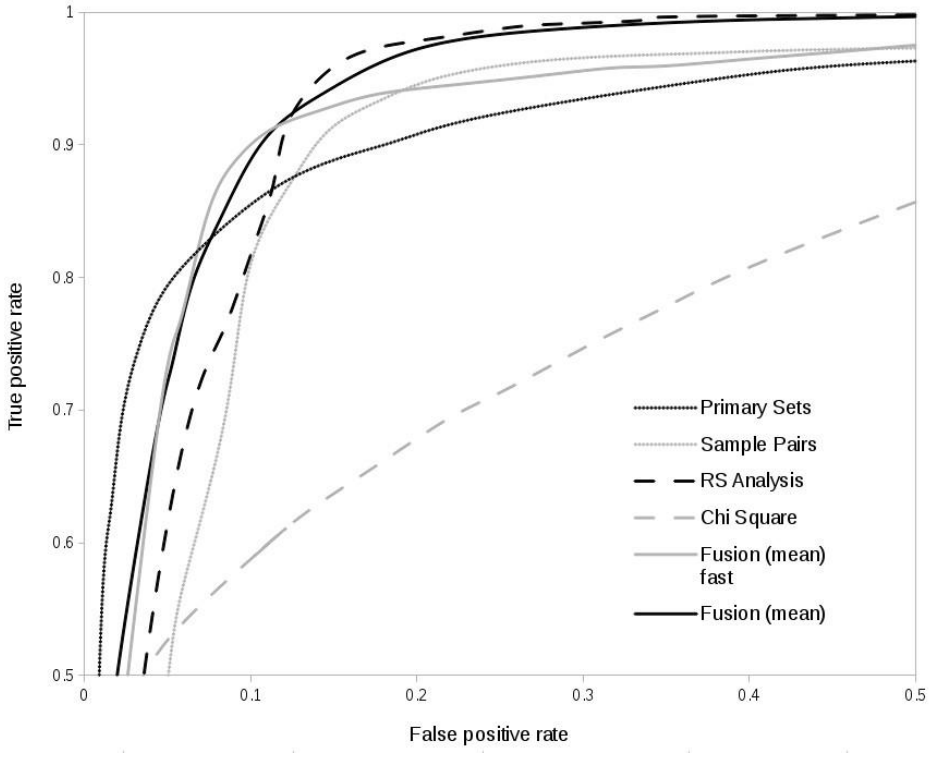

Figure 16 – Performance of StegExpose [3]

5.2 Pixelknot Detection

Pixelknot is as well-known and highly regarded image steganography tool for mobile devices. It uses F5 steganography to hide messages in DCT (Discrete Cosine Transform) coefficients and increase the embedding efficiency via matrix encoding. Our detection feature uses signature steganalysis to identify a consistent sequence of bytes within stego-objects and trace the use of steganography to the Pixelknot tool.

6. Forensic Metadata Analysis

Forensic metadata can be extracted using this tool. The RAMSES steganalytic framework makes use of ExifTool [9] to extract useful metadata from any given file (PNG, JPEG, MP4). Any useful information that can be taken from a file is compiled into a report alongside a SHA3-512 hash.
The options for this are user configurable, it is simple for the user to request more or less metadata type information from each file. This information is shown in Figure 17.

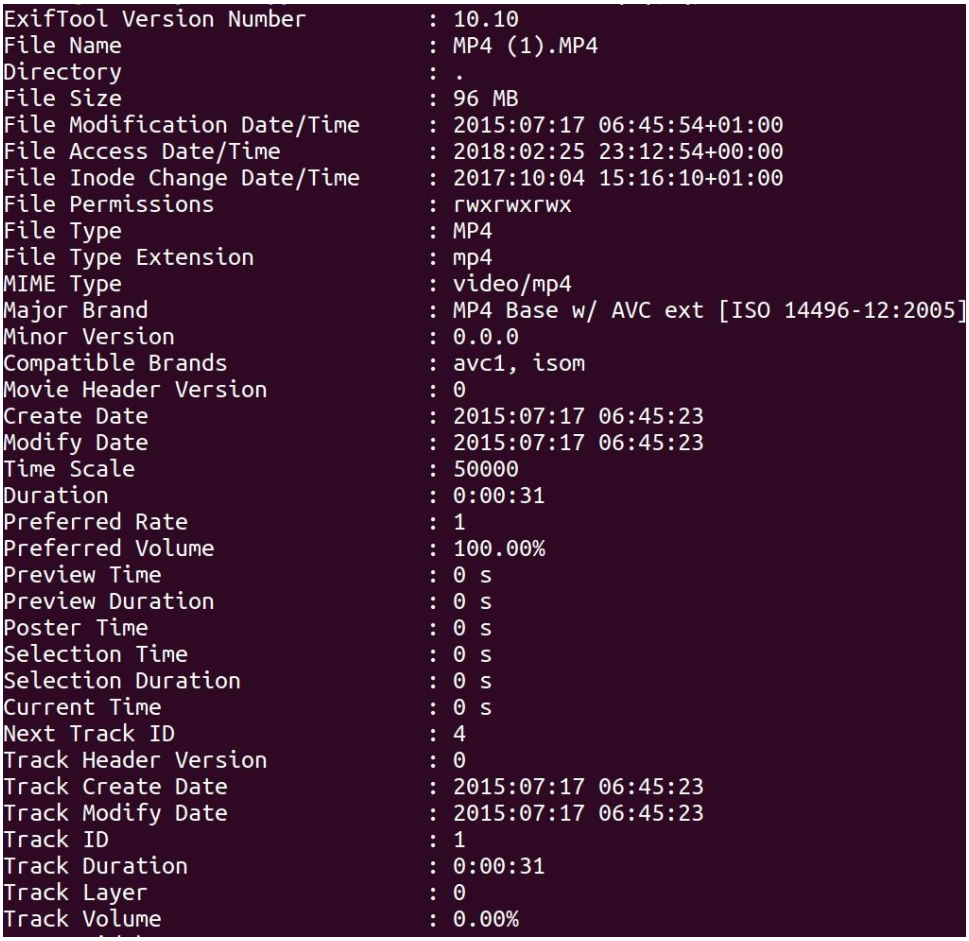

Figure 17 – Sample tag choices from Exiftool

Should you wish to discuss StegAware, please contact me by email – ts424@kent.ac.uk (Thomas Sloan)