A peer-reviewed version of this preprint was published in PeerJ on 27 May 2015.

View the peer-reviewed version (peerj.com/articles/cs-7), which is the preferred citable publication unless you specifically need to cite this preprint.

Sloan T, Hernandez-Castro J. 2015. Forensic analysis of video steganography tools. PeerJ Computer Science 1:e7 https://doi.org/10.7717/peerj-cs.7

Forensic Analysis of Video Steganography Tools

Thomas Sloan[1] **and Julio Hernandez-Castro**[2]

[1, 2]**University of Kent, School of Computing, Canterbury, Kent, CT2 7NF**

ABSTRACT

Steganography is the art and science of concealing information in such a way that only the sender and intended recipient of a message should be aware of its presence. Digital steganography has been used in the past on a variety of media including executable files, audio, text, games and, notably, images. Additionally, there is increasing research interest towards the use of video as a media for steganography, due to its pervasive nature and diverse embedding capabilities.

In this work, we examine the embedding algorithms and other security characteristics of several video steganography tools. We show how all feature basic and severe security weaknesses. This is potentially a very serious threat to the security, privacy and anonymity of their users. It is important to highlight that most steganography users have perfectly legal and ethical reasons to employ it. Some common scenarios would include citizens in oppressive regimes whose freedom of speech is compromised, people trying to avoid massive surveillance or censorship, political activists, whistle blowers, journalists, etc.
As a result of our findings, we strongly recommend to cease any use of these tools, and to remove any contents that may have been hidden, and any carriers stored, exchanged and/or uploaded online. For many of these tools, carrier files will be trivial to detect, potentially compromising any hidden data and the parties involved in the communication.
We finish this work by presenting our steganalytic results, that highlight a very poor current state of the art in practical video steganography tools. There is unfortunately a complete lack of secure and publicly available tools, and even commercial tools offer very poor security. We therefore encourage the steganography community to work towards the development of more secure and accessible video steganography tools, and make them available for the general public. The results presented in this work can also be seen as a useful resource for forensic examiners to determine the existence of any video steganography materials over the course of a computer forensic investigation.

Keywords: Steganography, steganalysis, EOF injection, signature, video, security, weakness, privacy, anonymity, censorship

INTRODUCTION

Steganography is the art and science of hiding information in plain sight. By ensuring that data is hidden from casual observers, a stego-system aims to reduce any suspicion that a third party may have over occurring communication. This can be a valuable resource where free speech is not guaranteed. In this and many other related contexts, steganography provides an ideal solution which makes it possible to avoid censorship (Krenn, 2004). An ideal stego-system should typically allow for highly sensitive information to be securely exchanged without the knowledge of others.

Recent research interest towards steganography focuses primarily on applications in the digital domain (Judge, 2001). Those provide a way for users to embed messages within files on a computer or a digital device. This process is typically carried out by identifying and manipulating redundant data in a file, so that any changes made are indiscernible through inspection. Modern digital stego-systems embed data through a variety of data hiding techniques such as image, audio and video steganography (Balgurgi and Jagtap, 2012). A rising interest in video steganography can be linked to the significant benefits it offers over alternative media formats. The pervasive nature of video, along with an increased embedding capacity, make it an ideal candidate. As a result, video steganography has benefited from rapid progress

in the academic domain (Balaji and Naveen, 2011). Unfortunately, publicly available tools do not reflect these academic advances, as we will show in the following. Most new and secure algorithms are only developed as a proof of concept tool in academic environments and do not lead to more secure tools that are accessible for the greater public.

The main contributions of this paper are:

- We examine a number of video steganography applications, both commercial and free. The list of analysed tools is comprehensive, as far as we know, with the exception of OpenPuff and MSU StegoVideo. For the tools discussed, we investigate and expose security-relevant characteristics.
- We show that all the tools discussed can be attacked, and in all cases it is possible to detect the presence of hidden contents. But even more powerful attacks are possible for certain applications: For example, in one particularly worrying case, it is feasible to extract the full hidden contents, without the need of the secret key.
- We show how simple scripts can be used to detect vulnerable embedding algorithms, and briefly discuss the impact of these weaknesses within the forensic domain.
- Furthermore, we provide recommendations on how these issues can be resolved.

There is currently a total lack of awareness on the insecurity of these tools, so it is therefore intended for our work to provide enough technical insights to justify our very strong recommendation against using these tools.

1 STEGANOGRAPHY

The purpose of steganography is to conceal the existence of communication. Modern steganography is typically carried out by manipulating data within carrier files. This concept can apply to a wide variety of file formats, as steganography is generally applicable to all data objects that contain some form of redundancy (Provos and Honeyman, 2003). However, media files are frequently ideal candidates as their pervasive nature will keep their true intent unnoticed by observers. Making use of the internet for file sharing, for instance, allows these carrier files to be sent and received with ease and without raising suspicions.

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1019v2 | CC-BY 4.0 Open Access | rec: 18 May 2015, publ: 18 May 2015

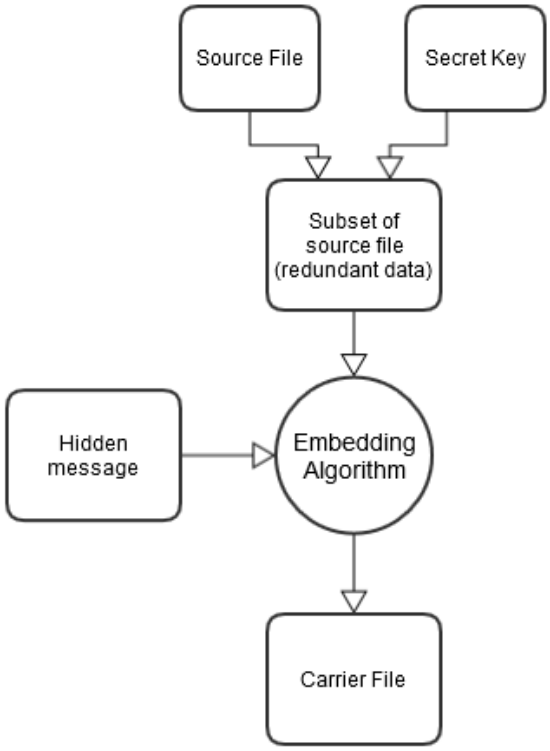

Figure 1. The general process of steganography

There are numerous embedding techniques that allow a user to hide data in a given object (Channalli and Jadhav, 2009). However, the general process of embedding data can be summarised in a few simple steps (as shown in Figure 1). Firstly, the communicating parties must agree on a stego-system and a shared secret key that will be used to determine the embedding algorithm. Once this has been achieved, the embedding algorithm will identify data within the file that is to be modified (these can be redundant bits). These bits are then replaced with the contents from the secret message that is to be embedded within the carrier file. As a result, analysis of a stego-file without knowledge of the key should not reveal the presence of steganography, therefore avoiding suspicion and keeping the hidden contents secure. In most cases, the hidden message will be encrypted prior to embedding.

1.1 Video Steganography & Steganalysis

Although still a relatively new research area, the field of video steganography has shown promising progress in recent years. A growing interest in the topic can be related to the many advantages that follow the use of a video file for hiding contents. Video steganographic techniques can provide the user with increased potential for capacity, transferability and most importantly imperceptibility (Dasgupta et al., 2013). Due to the ubiquity of video media throughout the internet, video steganography can prove to be a valuable resource to those who need to ensure their data is kept private. At present, only a small number of tools exist for public use in the form of both commercial and free software.

On the other hand, a very large number of steganography tools for other carriers are available. For example, the BackBone Steganography database lists fingerprints for over 1000 applications (BackBoneSecurity, 2014).

With so few video steganography tools available, the security that these applications offer must be thoroughly examined. With them potentially providing for a large number of users, it is critical to ensure

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1019v2 | CC-BY 4.0 Open Access | rec: 18 May 2015, publ: 18 May 2015

they are protecting data appropriately. If serious security vulnerabilities are found, as we argue in the rest of this paper, then awareness needs to be raised and their use discouraged to limit damage to its rightful users.

Video steganography can be a valuable resource for those who need to ensure that their communications remain private and secure. Most of the times, steganography will be employed for perfectly legitimate reasons. A good example are whistle-blowers, journalists, political activists, and basically anyone living in an oppressive regime where freedom of speech and individual rights are at stake. However, it is unfortunate that the advantages of steganography have been also of use to criminals and terrorists. In 2011, the German newspaper 'Die Zeit' (Gallagher, 2012) reported that an Al Qaeda agent had been intercepted with a memory card containing steganographic material documenting future plans within a video file. Research by certain criminal groups is leading to the development of new steganographic technology (Hosmer, 2012).

Current academic research in the field of video steganography explores techniques and embedding algorithms for securely hiding data within video files. The invasive nature of steganography typically leaves traces in a carrier file that can be detectable through careful analysis (Provos and Honeyman, 2003). Steganalysis follows this theme, to try and identify the presence of hidden content. State of the art techniques explore the temporal correlation between frames (Meghanathan and Nayak, 2010), however, this approach relies on tools that specifically embed data within the image frame. For a more in-depth presentation of the last advances in the matter, please check the Literature Review section.

Statistical analysis provides a way to examine the statistical properties of a carrier file for deviations that apply to a wide variety of embedding algorithms. Signature steganalysis is performed by identifying consistent, repetitive patterns embedded by a steganography tool (Mandal, 2012). Together, this approach provides an ideal method of attack against these tools, as it is typically applicable to a variety of embedding algorithms.

The steganalytic techniques used in this paper are typically in the remit of forensic steganalysis. This approach aims to discern the embedding algorithm of a stego-system, and to develop methods to distinguish between original and stego-content (Fridrich, 2009). Our method for examining stego-systems can also be defined as system steganalysis, as it exploits vulnerabilities within the tool implementation to detect the presence of steganography. The importance of these techniques is that they can drastically simplify the steganalytic process, and reveal crucial information surrounding a particular embedding algorithm.

2 LITERATURE REVIEW

A steganographic technique that we consider promising and has attracted quite a lot of academic interest over recent years is motion vector steganography. A motion vector is a key component of the motion estimation process. This embedding technique exploits the internal dynamics of video compression to hide messages (Cao et al., 2012). It benefits from high capacity for data hiding whilst maintaining excellent video quality (Pan et al., 2010). This steganographic approach has been subjected to quite heavy academic scrutiny, which has consequently lead to a number of interesting steganalysis results. Research carried out by Zhang et al. (2008) analysed statistical properties introduced by motion vector embedding to effectively detect motion vector based algorithms. Despite the current attacks, we still believe this approach offers promise, and will become a major embedding strategy in the future.

One of the many advantages of video steganography is that well-known embedding techniques for image and audio are also typically applicable to video. Research has been carried to bring the popular DCT steganographic technique to video. A system was proposed by Bodhak and Gunjal (2012) that embeds data through the manipulation of DCT coefficients. This method offers both imperceptibility against the original source and security of the embedded contents through encryption.

Although these proposed techniques highlight advancements in the field, they remain mainly theoretical and rarely make it to an implementation stage, particularly outside platforms such as MatLab, Maple, Mathematica, etc. This results in many of the tools available for the general public not being reflective of the latest academic developments in the area.

There was, however an interesting exception to this, though not a very successful one: An academic video

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1019v2 | CC-BY 4.0 Open Access | rec: 18 May 2015, publ: 18 May 2015

steganography tool was released by researchers at Moscow State University called MSU StegoVideo (Dmitriy Vatolin, 2007). This was one of the first video steganography tools subjected to academic steganalysis and has received significant interest. MSU StegoVideo is one of the first video steganography tools to embed data directly in the video stream, for which it is recognised as offering true video steganography. However, the tool is not open source (not good security-wise) and no details were provided by the author concerning the embedding algorithm (this is a terrible security practice, and a clear example of security-by-obscurity). Therefore, researchers interested in attacking the tool had to carry out some reverse engineering and frame-based analysis to properly understand the embedding mechanisms (Wu et al., 2010). After that, DCT-based steganalytic attacks have been used to successfully detect MSU StegoVideo Kancherla and Mukkamala (2009).

3 DISMANTLING TOOLS

Security vulnerabilities within any type of software are always a concern, and addressing them should be a matter of priority. This is even more evidently the case when dealing with vulnerabilities in security software.

Steganalysts attempt to detect, estimate and, if possible, extract hidden content within carrier files (Das et al., 2011). However, in certain cases where a tool shows weaknesses on a more fundamental level, direct attacks against the software can provide even more compromising results. This method of forensic and system steganalysis is our primary approach towards the analysis of the tools discussed throughout this paper.

We will focus on six video steganography tools (StegoStick, OurSecret, Masker, OmniHide Pro, BDV DataHider and Max File Encryption) that we prove offer a dangerously outdated and insecure embedding mechanism known as EOF data injection.

3.1 Embedding Algorithms

Across the nine video steganography tools we have identified, and that to the best of our knowledge covers all available tools for the general public, only three distinct embedding algorithms are in use, namely EOF Injection (by the six tools listed previously), metadata (by OpenPuff) and DCT (by MSU). The limitations of the DCT implementation in MSU StegoVideo have been shown in, notably, (Wu et al., 2010). We have presented similar powerful attacks (Sloan and Hernandez-Castro, 2015) against OpenPuff. In the rest of the paper, we will show the inherent vulnerability of the EOF Injection technique, thus completing a quite worrying view of the current security of video steganography tools.

3.2 Data Injection Tools

A total of six video steganography tools have been identified that employ EOF injection techniques (OurSecret, OmniHide Pro, Masker, StegoStick, BDV DataHider, and Max File Encryption). Many of their embedding approaches have similarities on both obvious and subtle levels. Most importantly, all six tools just embed data at the end of the file. This allows for the hidden content to be easily identified and it becomes possible to use this data to examine and reverse engineer the functionality of a given tool. In many cases, certain strings may be embedded into a carrier file that appear consistently throughout multiple hidden files. The consistency and the length of these strings provide a valuable means in which a carrier file alerts to the presence of steganography, and also links the carrier file to the respective tool. These are known as *signatures*, and are considered to be a very significant weakness in steganography tools.

We can perform this analysis despite these tools being proprietary and not including any kind of source code for analysis.

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1019v2 | CC-BY 4.0 Open Access | rec: 18 May 2015, publ: 18 May 2015

Toolkit	Price / Downloads	Resource location
OurSecret v2.5.5 (12/06/2012)	Free / 230,000+	http://www.securekit.net/oursecret.htm
OmniHide Pro v1.0 (16/08/2011)	$8 / 18,000+	http://omnihide.com/
Masker v7.5	€59 / 16,155+	http://www.softpuls.com/masker/
StegoStick Beta (16/06/2008)	Free / 6302+	http://stegostick.sourceforge.net/
Max File Encryption v2.0 (19/07/2013)	Free / 3606+	http://www.softeza.com/fileencryption/
BDV Data Hider v3.2 (01/06/2010)	$14.99 / N/A	http://www.bdvnotepad.com/products/bdv-datahider/

Table 1. Summary of EOF injection tools

Table 1 provides a basic insight into the popularity and accessibility of the six tools examined throughout this paper. Our count for the number of downloads was taken as of 07/04/2015[1]. Through an observation of the downloads across multiple download sites, we noticed that many of these tools are still downloaded each week, showing they are currently in use.

3.2.1 OurSecret

Formerly simply called 'Steganography', OurSecret provides the capability for video steganography on a wide variety of video formats. Currently on version 2.5.5, this tool offers an option for password based file protection, upon which users can only access embedded content by presenting the correct password. As freeware, the OurSecret tool imposes no restrictions or limitations on functionality. Research carried out by Adonis in 2007 identified a weakness within previous versions of this tool (affecting v. 1.7.1 and 1.8) that allowed an attacker to replace the user password with their own to extract embedded contents. However, this vulnerability as discussed seemed to be limited to JPEG images only (Adonis, 2007).

Similar to the vulnerability identified by Adonis in 2007, it is possible to extract the full embedded contents from a video file modified by OurSecret. In the absence of a password, the embedding algorithm will use a fixed 16-byte string to reflect this. This is a critical vulnerability. As this 16-byte hexadecimal string is used in place of a user-provided password, an attacker can simply copy this value into any OurSecret encoded files using a hex editor and bypass the user provided password with this null-password string. This simple modification will deceive the tool into recognising a carrier file as having no password requirements. As a result, the attacker is then be able to access and extract the full hidden contents. We denote this as a Null-Password Vulnerability.[2]

Figure 2 illustrates the Null-Password exploit in practice. These are the final 26 bytes of a file modified by OurSecret. The highlighted bytes are the Null-Password replacing the original 16 bytes of a file that were injected by OurSecret. This simple exchange grants the attacker full access to the hidden contents with no need for the secret key.

```
00 00 10 1A 00 00 6C 3C 39 6C 30 6B 6C 31 30 6E
38 38 6A 3A 38 3C 00 9B 12 00
```

Figure 2. Null-password replacing any previous user password

In addition, it is possible to observe a consistent 40-byte string, as shown in Table 2. This is a sequence of hexadecimal characters that occurs within all OurSecret modified video files, across all formats and encoding options. We can therefore classify this as a valid signature to identify the existence of any contents hidden with OurSecret. This signature appears directly after the final byte of an unmodified file. This is a fundamental vulnerability with the embedding algorithm, that compromises the security of

[1] Many of the tools listed in Table 1 do not show the number of downloads on their host site or any other centralised form, so we gathered these figures from a number of downloading sites and added them up. We can safely assume that the actual number of downloads is significantly higher that the figures shown here.

[2] This vulnerability is an improvement to the JPEG exploit as referenced on CVE-2007-0163, where this weakness was not mentioned.

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1019v2 | CC-BY 4.0 Open Access | rec: 18 May 2015, publ: 18 May 2015

a users data by informing attackers not only that steganography has been used on a video file, but also which steganography tool has been employed.

9E 97 BA 2A 00 80 88 C9 A3 70 97 5B A2 E4 99 B8 C1 78 72 0F
88 DD DC 34 2B 4E 7D 31 7F B5 E8 70 39 A8 B8 42 75 68 71 91

Table 2. OurSecret signature

In the light of the vulnerabilities we have identified within OurSecret, it is apparent that this tool is unsuitable for use in any circumstances. The identification of a unique signature will allow attackers to detect the presence of OurSecret steganography. Furthermore, by exploiting the null-password vulnerability, an attacker is capable of recovering all hidden information.

3.2.2 OmniHide Pro

OmniHide Pro is a commercial product capable of embedding data within image, audio, video and other media. Released in 2010 for the Windows operating system, OmniHide offers a Trial and a Pro version, although the former provides only limited functionality. OmniHide Pro costs $8, and will provide the user with all the tools' features. In the following, we will only examine the Pro version.

To study the embedding algorithm, we apply a similar testing framework to the one employed for OurSecret. Figure 3 shows a carrier file through an hexadecimal editor. It reveals a similar embedding algorithm to OurSecret, also employing EOF data injection. Reviewing the offset for where the unmodified file would have ended, it is apparent that the OmniHide Pro tool also suffers from several vulnerabilities that can compromise the security of a users' data. OmniHide embeds information within videos by appending the data directly to the end of the file. The first sequence of bytes are followed by a string of white space characters. These are reserved bytes to be replaced by longer file names that are embedded into the carrier.

Figure 3. First 11 bytes of embedded content

Figure 4 shows how we could identify the name of the text document embedded within the video. For the purposes of these tests we embedded a file called 'ATest.txt'. This converted ASCII is the first string of identifiable information. Using a hex viewer, this data is subsequent to the offset in which the unmodified file would have ended. The presence of a file name that is so easily identifiable within a carrier file, raises many security concerns. For example, if an attacker has access to the users' computer, it may be possible to identify the initial file still present in a plain text format. It is best practice for steganography tools to fully encrypt a file before embedding. This provides an additional layer of security and will make it more challenging for any attackers to identify the existence and location of hidden contents, and to recover them. OmniHide Pro clearly lacks this fundamental feature.

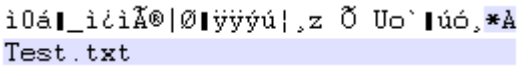

Figure 4. Hex string of embedded data, converted to ASCII

Further examination of OmniHide Pro revealed other characteristics of the embedding algorithm that can be considered as security weaknesses from a steganalytic perspective. As the file size of the carrier

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1019v2 | CC-BY 4.0 Open Access | rec: 18 May 2015, publ: 18 May 2015

increases proportionally to the embedded content, it is apparent that no compression is taking place during the embedding process. This is again against recommended best practice, as file compression should be used to reduce the size of embedded contents. Figure 5 captures the end of file characteristics of data injected by OmniHide Pro (approximately 19MB). This reflects the size of the carrier file prior to data injection. A subtraction between these two values will allow any attacker to quickly and accurately calculate the size of the injected hidden contents.

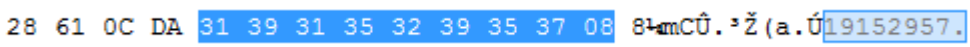

Figure 5. File size of the unmodified carrier

The identification of the file size in a carrier can be used as a pseudo-signature for Omnihide Pro. To examine the effectiveness of this pseudo-signature and its potential to correctly detect carrier files, two scripts (see Appendices A & B) were written and tested across 600 video files, using 100 of each of the following video formats: FLV, MP4, MPEGII, WMV, MOV and AVI. The results revealed that the effectiveness of the pseudo-signature is partially influenced by the file format used. The tool provided an accuracy of 97.17% across the 600 video files, with a 2.83% of false positives. The only format that generated false positives was WMV.

3.2.3 Masker

Masker is a commercial application that claims to offer secure video steganography. The latest stable format, version 7.5 was released in 2009 and can be accessed as either an evaluation version for a limited period of time, or alternatively, can be purchased for €59. Masker provides the option of several different encryption algorithms to secure user data, this includes: Blowfish, CASTS, DES, Serpent-256, AES-256, TripleDES and TwoFish. Masker does not work over MP4 files, but is capable of processing the following file formats: AVI, MOV, MPG, MPEG, ASF, MPA and MPE. We have been able to identify several weaknesses when examining the functionality of this tool. It is apparent that Masker shares many vulnerabilities with the previously discussed tools. Analysing an AVI file with a small amount of embedded content, it is apparent that Masker also employs EOF data injection.

Figure 6 illustrates the EOF injection technique adopted by Masker. The first 10 bytes in the image are the final part of the source file. The highlighted bytes reflect the start of Masker's embedding. Using a hex viewer, these EOF injection algorithms are trivial to detect and exploit.

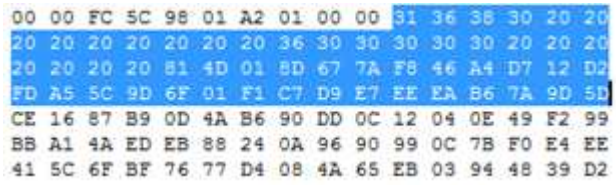

Figure 6. Masker embedding

Similarly to OurSecret, it is possible to identify a distinctive sequence of bytes that can uniquely characterise the embedding algorithm. This is consistent across the varying embedding options, irrespective of the encryption algorithm used or the choice of secret key. We can consider this string to be a signature for the Masker steganography tool and, due to its unusual length, it will ensure a high detection rate.

9E 8D DB 50 73 C2 BF 65 B8 1E 03 AE D5 62 C6 CC 71 9A B9 B9 48 49 D2 64 EE 49 B7 63 7B EF EB A1 01 04 40 7B F0 ED 2D 86 47 E1 5F FC 7C 41 C9 8B C9 02 CA 1C 07 21 EE 8D 32 66 47 7C 8F E1 4E E8 AE 66 AB 32 E3 D2 F9 A1 0E

Table 3. 73-byte Masker signature

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1019v2 | CC-BY 4.0 Open Access | rec: 18 May 2015, publ: 18 May 2015

The signature captured in Table 3 shows a 73-byte string that appeared throughout every test we carried out on Masker. This consistency of injected data opens the Masker tool to exploitation, essentially breaking the tool from a steganalytic perspective. Similar to a vulnerability within OmniHide Pro we were able to identify bytes that reflect the size of embedded contents.

Having identified the existence of a signature it is possible to create a simple script to detect the presence of steganography. An automated script would be easily capable of detecting a signature with a length of 73 bytes to a high degree of accuracy. From an academic perspective, this type of vulnerability complete breaks the tools capability for secure steganography. In Section 3.3 we further discuss our approach towards signature steganalysis based on the strings that have been captured.

3.2.4 Max File Encryption

Max File Encryption is a cryptographic tool with video steganography capabilities. The current version is 2.0, released on 07/2013. Analysis of Max File Encryption using the aforementioned test framework has provided similar discouraging results and illustrate many of the same mistakes. By employing EOF injection, it displays a concerning number of weaknesses. Upon inspection on a hex editor, it is apparent that the carrier file contains key components of the program, such as the libraries and public key tokens used, and other artifacts. Not only this can lead to a large number of signatures, it is also possible to identify key parts of the program source code through this.

Figure 7 identifies an embedding characteristic identical to that of OmniHide Pro. This tool injects information illustrating the file size of the unmodified carrier. This provides an attacker with the opportunity to estimate the size of embedded content. Provided an attacker has access to the source computer, it may be possible to identify the hidden file, generally the only one matching that exact size.

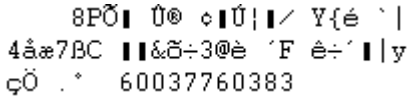

Figure 7. File size of embedded contents

Max File Encryption appears to be another generic EOF injection tool containing a number of already familiar vulnerabilities, that turn it into a very insecure stego system.

3.2.5 StegoStick

StegoStick is a freeware steganography tool that claims to have the capability to hide any file type into any other file type. This concept, although appealing, is significantly less tempting when realising it also employs EOF data injection. The current release is a beta version from 08/2013. At present, our examination of the tool has not revealed any consistent string that can be used for the detection of StegoStick steganography. The only recoverable information that could be extracted from the tool identifies the files types embedded into the video.

Figure 8 shows an ASCII representation of hex data captured from a StegoStick carrier file. We embedded varying file types such as .pdf and .txt into video files, these were identifiable during our analysis of the tool. The ability to extract embedded file types can provide small insights such as what file types to look for on a suspect computer. Although this tool uses EOF injection, which is again a very poor form of steganography, StegoStick does apparently not have the same number of security issues found in the previously analysed tools.

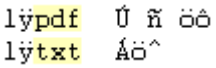

Figure 8. StegoStick embeds the file type of hidden data

A stronger steganalytic attack against StegoStick is briefly presented in section 3.4, where we describe a general approach that works against all examined EOF tools.

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1019v2 | CC-BY 4.0 Open Access | rec: 18 May 2015, publ: 18 May 2015

3.2.6 BDV DataHider

BDV DataHider is a commercial steganography application developed by Bedavlad Software. The most recent release is 3.2, available from 06/2010. This tool can be purchased for $14. Similar to alternative video steganography tools, BDV DataHider employs EOF injection techniques. Although the developers express high confidence in the quality of the tool and state "There is nothing more powerful on the market when you want to transfer unnoticeably (sic) some files over Internet or on a flash drive" (Software, 2010). However, it is apparent that the embedding algorithm features very similar weaknesses to those previously exposed. A major identifiable vulnerability is provided by the tools' interface.

As illustrated through Figure 9, BDV DataHider will inform any user if a file contains the presence of BDV DataHider steganography. Upon adding a carrier file to the GUI, the message "This file contain Secret Storage" will be shown. This demonstrates that there is the potential to acquire a signature from the tool. At the time of writing this paper, a consistent signature has yet to be identified.

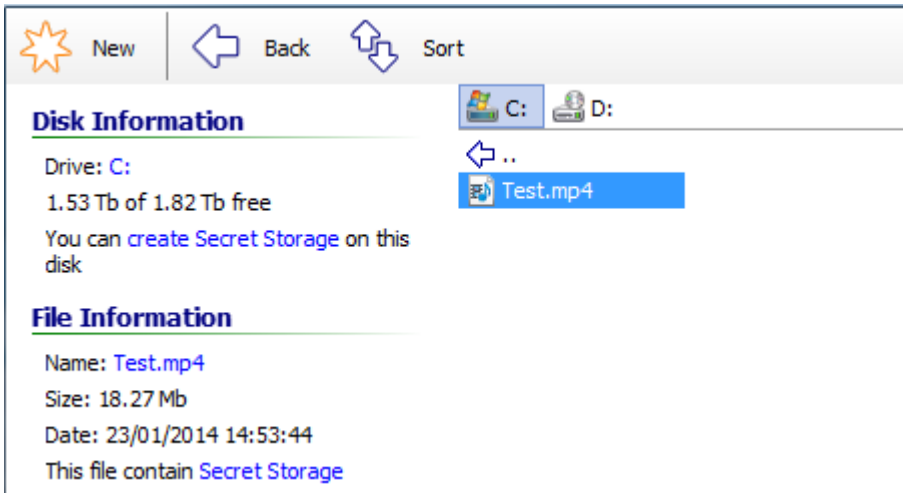

Figure 9. Data Hider provides its own detection system.

Again, we have a more powerful and general attacks against BDV DataHider, that will be briefly discussed in section 3.4

3.3 Signature Steganalysis

The identification of signatures for several of the video steganography tools analysed should be a cause for concern. We were able to develop simple but efficient and accurate scripts to detect these signatures within carrier files. We show this in Appendix C, this is tailored primarily towards the detection of OurSecret by exploiting the identified signature. However, it can be easily modified to detect similar tools by changing the signature that it looks for within the script. We found that these signatures provided 100% accuracy when attempting to detect the presence of video steganography. These tools should be considered completely broken from a steganalytic perspective, and users should stop using them immediately. We can not, unfortunately, offer a secure alternative because both OpenPuff and MSU StegoVideo are also vulnerable to steganalysis, and the EOF Injection approach is flawed beyond repair.

3.4 Generalising EOF Injection Attacks

Steganography performed through EOF data injection is totally insecure. Not only is detection trivial, but the resulting deviation in a files' statistical properties can leave a carrier file susceptible to a number of other, more powerful attacks.

If an attacker can automate a script to examine data injected at the end of a video file, it is possible to design a general attack against all of these tools. We tested this idea and, in each case, found it was possible to detect with perfect accuracy any EOF injection technique used the six tools previously discussed in this paper.

This provides a quick and general method for detecting this type of steganography, and emphasises how poor these algorithms actually are. The only shortcoming to this attack, however, is that when

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1019v2 | CC-BY 4.0 Open Access | rec: 18 May 2015, publ: 18 May 2015

compared with signature steganalysis it will not be able to link the carrier file to the tool used. On the other hand, this generalised attack (shown in Appendix D) can be used reliably against BDV DataHider and StegoStick, where we were unable to find a consistent signature.

4 SUMMARY OF FINDINGS

Throughout this paper, a total of six video steganography tools have been examined and evaluated. Multiple weaknesses have been identified across them, capable of seriously compromising users' data. In this section, we provide a brief summary of our findings in Table 4, highlighting the vulnerabilities discovered.

Toolkit	Embedding Algorithm	Vulnerabilities Identified
OurSecret	Data injection - EOF	40-byte signature, full retrievable contents with null password
OmniHide Pro	Data injection - EOF	Recover hidden filename and size, found pseudo-signature
Masker	Data injection - EOF	73-byte signature, hidden file size can be determined
StegoStick	Data injection - EOF	Hidden file type recoverable
Max File Encryption	Data injection - EOF	Multiple signatures
BDV Data Hider	Data injection - EOF	Auto detection within the tool

Table 4. Summary of the vulnerabilities found. It is important to note that all tools were also vulnerable to our generalised EOF detection algorithm.

We discovered that in all cases it is possible to isolate the embedding algorithm and develop detection methods against these tools. In this paper, we demonstrate how a warden can detect the presence of steganography within a video file. Following the identification of consistent signatures throughout several of the tools, scripts can be developed for the detection of hidden contents. The generalised attack against EOF injection was able to successfully identify the presence of video steganography across each of the tools discussed in this paper. Usage of any of these tools must stop immediately.

5 CONCLUSIONS AND FUTURE WORK

The findings in this paper emphasise the poor state of existing video steganography tools, highlighting how the use of simple signature-recognition scripts is enough to break a number of these applications. We have identified severe security vulnerabilities within each of the video steganography tools examined. We believe the described findings, together with the scripts accompanying this paper, will be useful for forensic practitioners searching for video steganography in their investigation. With only a handful of video steganography tools currently available, our findings are likely to affect a large number of video steganography users. We believe it is crucial to raise awareness into the inadequate security of these tools.

To further evaluate the merits of this approach, but also to investigate the prevalence of video steganography usage on the internet, we are currently working on a general framework, the Steganalytic wEbrEsearch frameworK (SEEK)[3]. We also plan to address other forms of video steganography, and to add DCT and motion vector detection to the framework. We also hope that showing the current poor state of affairs in publicly available video steganography will contribute to the future development of new and more secure tools. The case of MSU StegoVideo is a good example of academic progression for usable tools. Future tools should also pay more attention to include academic developments, and implement tried and tested techniques instead of ad-hoc algorithms.

[3] See http://thomassloan.com/my-research/

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1019v2 | CC-BY 4.0 Open Access | rec: 18 May 2015, publ: 18 May 2015

Appendices

A OMNIHIDE PRO SCRIPT 1

```
#!/bin/bash
IFS=$'\n'
echo "" > /home/user/Desktop/OmniHide/dumps/OHtmp1

for filename in /home/user/Desktop/Videos/*
do

      xxd -p $filename > /home/user/Desktop/OmniHide/dumps/OHtmp2
      tail -c 18 < /home/user/Desktop/OmniHide/dumps/OHtmp2 | tr -d '\n' >
          /home/user/Desktop/OmniHide/dumps/OHtmp3
      cat /home/user/Desktop/OmniHide/dumps/OHtmp3 >
          /home/user/Desktop/OmniHide/dumps/OHtmp1
      cat /home/user/Desktop/OmniHide/dumps/OHtmp3 | tr -d '\n'
          >>/home/user/Desktop/tail.txt | tr -d '\n'

sed -i '/^$/d' /home/user/Desktop/OmniHide/dumps/OHtmp1
sed -i '/^$/d' /home/user/Desktop/tail.txt
source /home/user/Desktop/OmniHide/detect.sh

done
```

B OMNIHIDE PRO SCRIPT 2

```
#!/bin/bin

IFS=$'\n'
for line in $(< /home/user/Desktop/OmniHide/dumps/OHtmp1); do

      result=$(echo $line | grep -Eo [[:digit:]] | wc -l)

      if [[ $result -gt 16 ]]; then
      echo -e \ "Content hidden with OmniHide Pro found on" $filename
          >>/home/user/Desktop/results
      else
      echo -e \ "No content hidden with Omnihide Pro on" $filename
          >>/home/user/Desktop/results
fi
done
```


PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1019v2 | CC-BY 4.0 Open Access | rec: 18 May 2015, publ: 18 May 2015

C SIGNATURE DETECTION SCRIPT FOR OURSECRET

```
#!/bin/bash

for filename in /home/user/Desktop/Videos/*
do
xxd -p $filename > /home/user/Desktop/tmp1
      tr -d '\n' < /home/user/Desktop/tmp1 > /home/user/Desktop/tmp2
      if grep -c -q
          -"9e97ba2a008088c9a370975ba2e499b8c178720f88dddc342b4e7d317fb5e
   87039a8b84275687191" /home/ts424/Desktop/tmp2; then
      echo -e \ "Content hidden with OurSecret found in" $filename >>
          /home/user/Desktop/results
      else
      echo -e \ "No OurSecret content in" $filename >>
          /home/user/Desktop/results
fi
done
```

D GENERALISED EOF DETECTION SCRIPT

```
#!/bin/bash
catch="invalid atom size, extends outside parent atom"
printf '%s\n' $file | tr -d '\n' >> /home/user/Desktop/EOF/EOFResults.txt
for file in /home/user/Desktop/Videos/*
do
mp4file --dump $file > /home/user/Desktop/EOF/EOFDump

test=$(grep -o "invalid atom size, extends outside parent atom"
    /home/user/Desktop/EOF/EOFDump)
echo $test >/home/user/Desktop/EOF/EOFtmp1

if [[ $test == $catch ]]
then
echo -e \ "EOF injection found" >>/home/user/Desktop/EOF/EOFResults.txt
else
echo -e \ "No EOF steganography" >>/home/user/Desktop/EOF/EOFResults.txt
fi

done
```


PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1019v2 | CC-BY 4.0 Open Access | rec: 18 May 2015, publ: 18 May 2015

REFERENCES

Adonis (2007). Cve details - securekit. http://www.cvedetails.com/cve/CVE-2007-0163.

BackBoneSecurity (2014). *Backbone's Steganography Database Exceeds 1,000 Applications*.

Balaji, R. and Naveen, G. (2011). Secure data transmission using video steganography. In *Electro/Information Technology (EIT), 2011 IEEE International Conference on*, pages 1–5. IEEE.

Balgurgi, P. P. and Jagtap, S. K. (2012). Audio steganography used for secure data transmission. *Proceedings of International Conference on Advances in Computing*, pages 699–706.

Bodhak, P. V. and Gunjal, B. L. (2012). Improved protection in video steganography using dct & lsb. *International journal of engineering and innovative technology (IJEIT)*, 1(4).

Cao, Y., Zhao, X., and Feng, D. (2012). Video steganalysis exploiting motion vector reversion-based features. *Signal Processing Letters, IEEE*, 19(1):35–38.

Channalli, S. and Jadhav, A. (2009). Steganography an art of hiding data. *arXiv preprint arXiv:0912.2319*.

Das, S., Das, S., Bandyopadhyay, B., and Sanyal, S. (2011). Steganography and steganalysis: different approaches. *arXiv preprint arXiv:1111.3758*.

Dasgupta, K., Mondal, J. K., and Dutta, P. (2013). Optimized video steganography using genetic algorithm (ga). *Procedia Technology*, 10:131–137.

Dmitriy Vatolin, O. P. (2007). Msu stegovideo. *http://goo.gl/XqiWi1*.

Fridrich, J. (2009). *Steganography in Digital Media*. Cambridge University Press.

Gallagher, S. (2012). Steganography: How al-qaeda hid secret documents in a porn video. *http://arstechnica.com/business/2012/05/steganography-how-al-qaeda-hid-secret-documents-in-a-porn-video*.

Hosmer, C. (2012). Data hiding and steganography annual report 2012. *WetStone Technologies*.

Judge, J. (2001). Steganography: past, present, future. sans institute publication.

Kancherla, K. and Mukkamala, S. (2009). Video steganalysis using motion estimation. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pages 1510–1515. IEEE.

Krenn, R. (2004). Steganography and steganalysis. *Retrieved September*, 8:2007.

Mandal, P. C. (2012). An extensive review of current trends in steganalysis. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 1(7):pp–215.

Meghanathan, N. and Nayak, L. (2010). Steganalysis algorithms for detecting the hidden information in image, audio and video cover media. *international journal of Network Security & Its application (IJNSA)*, 2(1):43–55.

Pan, F., Xiang, L., Yang, X.-Y., and Guo, Y. (2010). Video steganography using motion vector and linear block codes. In *Software Engineering and Service Sciences (ICSESS), 2010 IEEE International Conference on*, pages 592–595. IEEE.

Provos, N. and Honeyman, P. (2003). Hide and seek: An introduction to steganography. *Security & Privacy, IEEE*, 1(3):32–44.

Sloan, T. and Hernandez-Castro, J. (2015). Steganalysis of openpuff through atomic concatenation of mp4 flags. *Digital Investigation*, 13:15–21.

Software, B. (2010). Bdv datahider. *http://www.bdvnotepad.com/products/bdv-datahider/*.

Wu, J., Zhang, R., Chen, M., and Niu, X. (2010). Steganalysis of msu stego video based on discontinuous coefficient. In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, volume 2, pages V2–96. IEEE.

Zhang, C., Su, Y., and Zhang, C. (2008). A new video steganalysis algorithm against motion vector steganography. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM'08. 4th International Conference on*, pages 1–4. IEEE.

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1019v2 | CC-BY 4.0 Open Access | rec: 18 May 2015, publ: 18 May 2015