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 Sloan¹ and Julio Hernandez-Castro²

¹, ²University 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. There is increasing research interest towards the use of video as a media for steganography. This is, at least in part, due to its pervasive nature and good embedding capabilities.

In this article, we examine the embedding algorithms and security characteristics of several video steganography tools. We conclude that many feature basic and severe weaknesses. This constitutes a serious threat to those using these applications, some of which have perfectly legal or ethical reasons to do so, such as those whose freedom of speech is superseded by oppressive regimes, whistle-blowers, journalists, etc.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, exchanged and/or uploaded online. For many of these tools, carrier files will be trivial to detect, potentially compromising any hidden data. We finish this work by presenting our steganalytic results, that highlight a very poor state of the art in practical video steganography tools. As there is a complete lack of secure and publicly available tools, both free and commercial, we therefore encourage the steganography community to work towards the development of more secure and accessible video steganography tools 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

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). Through this, it provides 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 never lead to more secure tools that are accessible for the greater public.

The main contributions of this paper are:
- We analyse the steganographic algorithms for video used by both commercial and free publicly available tools.
- We show that all the tools can be attacked, by detecting the presence of hidden contents. But even more powerful attacks are possible: For example, in one case it is even feasible to extract the full original contents without the need of the secret key.
- We show how simple scripts can be used to detect vulnerable embedding algorithms, and discuss the impact of these trivial weaknesses within the forensic domain. Furthermore we provide recommendations on how these issues can be resolved.

There is currently a lack of technical information related to the functionality of these tools, so it is therefore intended for this paper to additionally provide technical insights to justify our 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.

![Figure 1. The general process of steganography](https://dx.doi.org/10.7287/peerj.preprints.1019v1)
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). To begin, the communicating parties must first agree on a shared stego-system and a shared secret key that will be used to determine the embedding algorithm. Once this has been acknowledged, 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 content 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.

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. 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, a number tools exist for public use in the form of both commercial and free applications. A large number of varying steganography tools are available, the Backbone Steganography database lists fingerprints for over 1000 applications (BackBoneSecurity, 2014) but only a small handful are capable of performing video based steganography. With so few video steganography tools available, the security that these applications offer must be thoroughly examined. With these tools potentially providing for a large number of steganography users, it is critical to ensure 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 is 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 detectable traces in a carrier file that may be able to detectable through careful analysis (Provos and Honeyman, 2003). Steganalysis follows this theme, using novel research 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 in a video file. For a more in-depth presentation of the last advances in the matter, please check the Literature Review section in the following. Regarding steganalysis, 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). This type of steganalysis 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 formal methods to distinguish between original and stego-content (Fridrich, 2009). Our method of examining stego-systems can also be defined as system steganalysis, this method exploits vulnerabilities within the implementation to detect the presence of steganography such as that used in signature steganalysis. 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). This embedding technique benefits from high capacity for data hiding whilst maintaining excellent video quality (Pan et al., 2010). This steganographic method has been subjected to quite heavy academic scrutiny, which has consequently lead to a number of interesting steganalytic techniques against motion vector steganography. Research carried out by Zhang et al. (2008) analysed statistical properties introduced by motion vector embedding to effectively detect motion vector 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 that brings the popular DCT steganographic technique to video. A system was proposed by Bodhak and Gunjal (2012) that embedded data through the manipulation of DCT coefficients in the transform domain. This proposed method offers 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 not being reflective of the latest academic developments in the area.

There was, however an exception to this, but it was not a very successful one: An academic video steganography tool was released by 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 terrible security wise, as a clear example of security-by-obscurity). Therefore, researchers had to carry out some reverse engineering and frame-based analysis to understand the embedding mechanisms (Wu et al., 2010). Consequently, DCT-based steganalytic attacks have been used to successfully detect MSU StegoVideo inside video streams Kancherla and Mukkamala (2009).

3 DISMANTLING TOOLS

Vulnerabilities within any type of software are always a concerns, and addressing them should always be a matter of priority. This is more evidently the case when dealing with vulnerabilities discovered in security software. There are many ways in which tools can be examined for vulnerabilities, most commonly through reverse engineering. Techniques for the detection of embedded content inside files fall within the field of steganalysis. Steganalysts attempt to detect, estimate and sometimes extract hidden content within carrier files (Das et al., 2011). However, in certain cases where a tool shows weaknesses of a more fundamental level, direct attacks against the software can provide significantly better results. This method of forensic and system steganalysis is our primary approach towards the analysis of the tools discussed throughout this paper.

In this paper, we 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 totally 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. This are known as signatures, and are considered to be a 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.

| 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.bdvnopad.com/products/bdv-datahider/ |

Table 1. Summary of EOF injection tools

3.2.1 OurSecret

Formerly 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 (as shown in Figure 2).

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 total 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.
In addition, it is possible to observe a consistent 40 byte string. This is a sequence of hexadecimal characters that occur within all OurSecret modified video files, across all formats and encoding options. We can therefore categorise 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 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.

![Figure 2. Null-password replacing any previous user password](image)

In the light of the vulnerabilities we have identified within OurSecret, it is apparent that this tool is unsuitable for use in all 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 comprises a Trial and a Pro version, although the former offers 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 OurSecret. A cursory analysis of a modified file reveals a similar embedding algorithm to that uses 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. As shown in Figure 3, 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](image)

We could immediately identify the name of the text document embedded within the video. For the purposes of these tests the file was called "ATest.txt" and within the video, this is the first string of identifiable ASCII, followed directly after the offset in which the unmodified file would have ended a video stream (as shown in Figure 4). The presence of a file name that is so easily identifiable within a carrier file, raises security concerns. 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 generally recommended as 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. OmniHide Pro clearly lacks this fundamental feature.

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

Further examining the OmniHide Pro embedding algorithm, it is possible to acquire other important details concerning the tools’ functionality. As the file size of the carrier increases proportionally to the embedded content, it is apparent that no compression is taking place during the embedding process. This is also against recommended best practice. In addition, the final bytes of a modified video file represent the initial video size before any data encoding takes place (as shown in Figure 5).

Figure 5. File size of the unmodified carrier

A consistent pseudo-signature has been found for files with contents hidden by OmniHide Pro. The identification of the initial file size before the embedding provides valuable information to the steganalyst; with this they will be capable of determining the exact size of the embedded contents. Using this information, as well as obtaining the name of the file that has been encoded within the carrier, attackers will be far more likely to succeed at recovering sensitive information from files modified by OmniHide. Specifically, in investigations where access to the full contents of a computer may be possible, obtaining the file name and size before any steganography has taken place can be of great help.

To examine the effectiveness of the pseudo-signature and its potential to correctly identify steganography within carrier files, two scripts were tested. The function of each script is to cut the final bytes of a file that identifies the carriers unmodified file size and examine the string. If the analysed string contains only integers, it then infers the presence of OmniHide Pro steganography. Essentially the function of this bash script is to detect the absence of alphabetic characters, whilst emphasising purely integer based strings.

Tests were carried out 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 directly limited by the file format used. The tool provided an accuracy rate of 97.17% across the 600 video files, with 2.83% of occurrences flagged as false positives. All occurring false positives were revealed to be only from the WMV video format. Furthermore, it is apparent that using this pseudo-signature on the MPEGII video format will provide 100% accuracy, as the final bytes of an MPEGII file consist of entirely alphabetic characters rather than integers.

3.2.3 Masker

Masker is a commercial steganography product that has capability for video embedding. 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, CAST5, DES, Serpent-256, AES-256, TripleDES and TwoFish. However, for embedding data within video files, Masker is limited to the following file formats: AVI, MOV, MPG, MPEG, ASF, MPA and MPE.

Our analysis of this tool focused primarily on AVI as the over MP4 due to compatibility issues. We have been able to identify several key features when examining the functionality of this tool and its embedding algorithm. It is apparent that Masker shares important weaknesses with the previously discussed Our-Secret and OmniHide Pro tools. Reviewing an AVI file with a small amount of content embedded, it is straightforward to see that Masker also employs EOF data injection. As shown in Figure 6, the highlighted data reflects the first bytes subsequent to the video.
Similarly to OurSecret, it is possible to identify a unique sequence of bytes that can uniquely characterise the embedding algorithm. This is consistent across the varying embedding options. We can consider this string to be a signature for the Masker steganography tool and due to the length of the signature it can ensure a high rate of accuracy for detection. Figure 9 shows the signature identified for the Masker embedding algorithm.

Table 3. 73-Byte Masker signature

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 and with a high degree of accuracy. From an academic perspective, this type of vulnerability complete breaks the tools’ capability for secure steganography. In addition, it is possible to observe the file size of content embedded within Masker carrier files (Figure 7). Provided an attacker has full access to the source computer, it may be possible to identify the original information in a plain text format based on an observation of the file size.

Figure 7. Masker encodes the file size of embedded data

3.2.4 Max File Encryption

Max File Encryption is a cryptographic tool with video steganography support and capabilities. The current version of this tool is 2.0 released 07/2013. Analysis of Max File Encryption using the aforementioned framework of tests has provided similar results to other tools and illustrates that Max File Encryption makes the same errors as many other steganography tools. Furthermore, it has shown that this is another application that gives little consideration to stego-security beyond obscurity and basic data hiding. Employing EOF data injection as the chosen form of steganography, this tool displays a significant number of observable weaknesses.

It is apparent that in many parts of the modified file are key components of the program output. Not only is this consistently and highly repeated throughout, leading to a large number of detectable signatures, it is also possible to identify key parts of the program source code through this. This type of weakness allows an attacker to pick from a large reserve of potential signatures to scan for this tool specifically.
Similar to OmniHide Pro, this tool also provides information concerning the file size of the unmodified carrier (Figure 8). This provides an attacker with the opportunity to estimate the size of embedded content with a simple deduction. Provided an attacker has access to the victim’s computer, it may be possible to identify a selection of files by file size that match the properties of the embedded content as plain text.

Max File Encryption is another generic EOF injection tool that alike many others, contains a host of vulnerabilities that not only lead to the identification of video steganography, but also link the carrier file to its respective tool.

**StegoStick**

StegoStick is a freeware steganography tool that has the capability to hide any file type into any other file type. The concept of this, although appealing, is significantly diminished by the execution of EOF data injection as the chosen form of steganography. The current release of this tool is a beta version from 08/2013. At present, signature steganalysis has not revealed any consistent string that can be used for the detection of StegoStick steganography. The only purely obtainable information that could be extracted from the tool identifies the file types embedded into the video (as shown in Figure 10).

The ability to identify embedded file types can provide small insight into the hidden content, however this does not equal the security concerns that are present within similar data injection tools.

**BDV DataHider**

BDV DataHider is a commercial steganography application developed by Bedavlad Software. The most current release is 3.2 (06/2010), this tool is purchasable 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 some files over Internet or on a flash drive” (Software, 2010). However, it is apparent that embedding algorithm for BDV DataHider features similar weaknesses alike many other EOF injection tools. One of the first identifiable weaknesses is provided by the tool’s interface.
Figure 11. Data Hider provides its own detection system

As illustrated through Figure 11, BDV DataHider will inform any user if a file contains the presence of BDV DataHider steganography. This demonstrates that there is the potential to acquire a signature from the tool that can be used to detect the presence of BDV DataHider steganography. At the time of writing this paper, a consistent signature has yet to be identified.

3.3 Signature Steganalysis
The identification of a number of consistent signatures for several of the video steganography tools discussed in this paper should be a major security concern. We were able to develop simple but efficient and accurate scripts to detect these signatures within stego-modified carrier files. By exploiting the signatures extracted from OurSecret, Masker and Max File Encryption, we were successful in identifying the presence of video steganography with perfect accuracy. Our scripts provided 100% accuracy in detecting each tool with the corresponding signature. For the rest of the tools, we offered slightly different but similarly powerful attacks. 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 for the moment because both OpenPuff and MSU StegoVideo are also vulnerable to steganalysis and the above tools are flawed beyond repair.

3.4 Generalising EOF Injection Attacks
Steganography performed through EOF data injection is insecure. Not only is detection trivial, but the resulting deviation in a files’ statistical properties without decorrelation can leave a carrier file susceptible to a number of other, more powerful attacks. We have identified six video steganography tools that employ EOF data injection and consider to provide inadequate security of user data.

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 and in each case found it was possible to detect EOF injection in the six tools discussed in this paper. This provides a quick and general method for detecting this type of steganography, and emphasises how poor these embedding algorithms actually are. The weakness to this attack however is that when compared with signature steganalysis it will not be able to link the carrier file to the concrete tool that embedded the content respectively. On the other hand, this generalised attack 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 these tools, that are capable of seriously compromising
users’ data. In this section, we provide a summary of findings that illustrate all the vulnerabilities discovered.

| Toolkit         | Embedding Algorithm | Vulnerabilities Identified                           |
|-----------------|---------------------|-----------------------------------------------------|
| OurSecret       | Data injection - EOF| 40 byte signature, retrievable contents with null password |
| OmniHide Pro    | Data injection - EOF| Can recover hidden filename, and size, plus pseudo-signature |
| Masker          | Data injection - EOF| 73 byte signature, file size can be determined       |
| StegoStick      | Data injection - EOF| file type                                            |
| 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 notice all tools were also vulnerable to the generalised EOF detection algorithm we proposed.

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 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. This emphasises the poor state of the art regarding accessible video steganography tools. For those concerned about the privacy and security of their data, action should be taken immediately and the usage of these tools must be stopped.

5 CONCLUSION

When using a tool to perform video steganography, a user is placing trust in the tool capabilities to securely embed data, these expectations should be met especially if the tool is paid for. This concealed information should not only be indiscernible, but the embedding algorithm should ensure it protects the encoded contents effectively. Simply concealing the data from sight is clearly not enough. In addition, our results further prove the need of security tools to be open source, avoiding the outdated principle of security through obscurity.

We have identified severe vulnerabilities within each of the video steganography tools examined. Furthermore, signatures have been found for several applications that can be used to help attackers track the presence of these tools within video files. These signatures can be quickly incorporated into detection systems and video scrapping tools to reliably detect the presence of steganography and even to recover the hidden contents, completely defeating the primary purpose of steganography. Additional analysis revealed the existence of a null-password vulnerability in OurSecret. This would allow an attacker to extract password protected data at will. With only a small handful of video steganography tools currently available, these discoveries are likely to affect a large number of video steganography users. It is crucial to raise awareness into the inadequate security of these EOF injection tools.

6 FUTURE WORK

The findings in this paper emphasise the state of existing video steganography tools and how we can use highly accurate signature steganalytic scripts to break many of these tools. To fully evaluate this approach, we are working on a web framework to detect the presence of video steganography across a wide variety of tools and launch an web-based search for videos with hidden data. To achieve this we have started development on the Steganalytic wEbrEsearch frameworK (SEEK)³.

The purpose of this framework is to research into the usage of video steganography throughout the internet. We have included a variety of detection algorithms (including those discussed in this paper), alongside others for different video steganography tools such as OpenPuff. We plan to address other forms of video steganography and to add DCT and motion vector detection to the framework.

³See http://thomassloan.com/seek
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. 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 to follow, except for not disclosing the embedding algorithm. Future tools should also pay more attention to academic developments, and reflect tried and tested techniques instead of ad-hoc algorithms.

Appendices

A SIGNATURE DETECTION SCRIPT FOR OURSECRET

```bash
#!/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 "9e97ba2a008088c9a370975ba2e499b8c178720f88ddc342b4e7d317fb5e87039a8b84275687191" /home/ts424/Desktop/tmp2; then
            echo -e "OurSecret hidden content found in" $filename >>/home/user/Desktop/results
        else
            echo -e "There is no OurSecret content hidden within" $filename >>/home/user/Desktop/results
        fi
    done
```

B GENERALISED EOF DETECTION SCRIPT

```bash
#!/bin/bash

catch="invalid atom size, extends outside parent atom"

printf '%s
' $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 detected" >>/home/user/Desktop/EOF/EOFResults.txt
        else
            echo -e "No steganography detected"
                >>/home/user/Desktop/EOF/EOFResults.txt
        fi
    done
```

C OMNIHIDE PRO SCRIPT 1
#!/bin/bash

IFS=\$'\n'

for filename in /home/user/Desktop/Videos/*
do
    xxd -p $filename > /home/user/Desktop/OmniHide/dumps/OHtmp2
    tail -c 18 < /home/user/Desktop/OmniHide/dumps/OHtmp2 | tr -d \'\n' >
    /home/user/Desktop/OmniHide/dumps/OHtmp3
    cat /home/user/Desktop/OmniHide/dumps/OHtmp3 | tr -d \'\n'
    >>/home/user/Desktop/OmniHide/dumps/OHtmp1
    cat /home/user/Desktop/OmniHide/dumps/OHtmp3 | tr -d \'\n'
    >>/home/user/Desktop/OmniHide/dumps/OHtmp1
    sed -i '/^$/d' /home/user/Desktop/OmniHide/dumps/OHtmp1
    sed -i '/^$/d' /home/user/Desktop/OmniHide/dumps/OHtmp2
    sed -i '/^$/d' /home/user/Desktop/OmniHide/dumps/OHtmp3
    cat /home/user/Desktop/OmniHide/dumps/OHtmp2 >
    /home/user/Desktop/OmniHide/dumps/OHtmp3
    cat /home/user/Desktop/OmniHide/dumps/OHtmp3 | tr -d \'\n'
    >>/home/user/Desktop/OmniHide/dumps/OHtmp2
    cat /home/user/Desktop/OmniHide/dumps/OHtmp3 | tr -d \'\n'
    >>/home/user/Desktop/OmniHide/dumps/OHtmp3

done

D OMNIHIDE PRO SCRIPT 2

#!/bin/bash

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 "OmniHide Pro content may be present within" $filename
        >>/home/user/Results/results
    else
        echo -e "OmniHide Pro has no presence within" $filename
        >>/home/user/Results/results
    fi

done
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., Bandopadhyay, 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.