# HTML5 ZERO CONFIGURATION COVERT CHANNELS: SECURITY RISKS AND CHALLENGES

Jason Farina Mark Scanlon Stephen Kohlmann
Nhien-An Le-Khac Tahar Kechadi

School of Computer Science & Informatics,
University College Dublin, Ireland.
{jason.farina, stephen.kohlmann}@ucdconnect.ie, {mark.scanlon, an.lekhac, tahar.kechadi}@ucd.ie

**ABSTRACT**

In recent months there has been an increase in the popularity and public awareness of secure, cloudless file transfer systems. The aim of these services is to facilitate the secure transfer of files in a peer-to-peer (P2P) fashion over the Internet without the need for centralized authentication or storage. These services can take the form of client installed applications or entirely web browser based interfaces. Due to their P2P nature, there is generally no limit to the file sizes involved or to the volume of data transmitted – and where these limitations do exist they will be purely reliant on the capacities of the systems at either end of the transfer. By default, many of these services provide seamless, end-to-end encryption to their users. The cybersecurity and cyberforensic consequences of the potential criminal use of such services are significant. The ability to easily transfer encrypted data over the Internet opens up a range of opportunities for illegal use to cybercriminals requiring minimal technical know-how. This paper explores a number of these services and provides an analysis of the risks they pose to corporate and governmental security. A number of methods for the forensic investigation of such transfers are discussed.

**Keywords**: Covert Transfers, Encrypted Data Transmission, Counter-forensics

## 1. INTRODUCTION

For the typical home user, sending anything larger than a single image or document electronically is still a cumbersome task when reliant on popular online communication methods. Most email providers will limit the file size of attachments to something in the order of megabytes and many will additionally restrict file types such as executables or password protected archives based on internal security policies. Sending larger files usually requires users to upload the content to third party storage providers, e.g., Dropbox, OneDrive, Box.net, etc., and provide a link to the content to their intended recipients. From a security standpoint, this leaves user vulnerable to their communication being intercepted or duplicated and their data being downloaded by others. Regarding the security of their data stored on this third-party provider, users must blindly trust this third-party to not access or share their data with any unauthorized party.

While the requirement to send larger volumes of information over the Internet is ever increasing, the potential for third-party interception, or illicit access to, this data has become a common story in the general media. Recent leaks from whistle-blowers regarding the degree of surveillance conducted by large government funded spying agencies on everyday citizens has pushed the topic of cybersecurity into the public realm. Increasingly, Internet users are becoming conscious of their personal responsibility in the pro-

 

tection of their digital information. This results in many users being discontent with their personal data stored on these third party servers – likely stored in another jurisdiction with privacy requirements much lower than those of their own locale.

To respond to this demand a number of file exchange/transfer services have grown in popularity in recent months facilitating the secure transfer of files in a peer-to-peer (P2P) fashion from point A to point B. Most of these services afford the user encrypted end-to-end file transfer and add an additional level of anonymity compared to regular file transfer services, e.g., email attachments, FTP or instant message file-sharing. The more security concerned users will opt for the cloudless versions of these services. This independent control over personal information has advantages and disadvantages for the end user. The advantage is that the user has precise knowledge over who has initial access to his/her information and what country the data is stored in. The downside comes in terms of reliability. The data stored or transferred using these services is only available if at least one host storing the file is online.

As with most security or privacy enhancing Internet services, these services are open to abuse by cybercriminals. In effect, the additional level of anonymity and security provided by these services provides cybercriminals with "off-the-shelf" counter-forensic capabilities for information exchange. Cybercriminal activities such as data exfiltration, the distribution of illicit images of children, piracy, industrial espionage, malicious software distribution, and can all potentially be facilitated by the use of these services. The ad hoc nature and mainstream protocol features of some of these services make secure detection and prevention problematic without causing disruption to normal Internet usage. This also causes forensic event reconstruction difficulties as any traces have a high degree of volatility when compared to more traditional file transfer options.

### 1.1 Contribution of this work

For many, the topic of covert channels immediately brings to mind some form of steganography likely in combination with an Internet anonymizing service, such as Tor and I2P. While some work has been conducted on the reverse engineering/evidence gathering of these anonymizing P2P proxy services, little work has been done in the area of online services providing end users with the ability to securely and covertly transfer information from peer to peer in an largely undetectable manner. This work presented as part of this paper examines a number of popular client application and web based services, outlines their functionality, discusses the forensic consequences and proposes a number methods for potentially retrieving evidence from these services.

## 2. BACKGROUND READING

The technology used to facilitate covert file transfers is not new, however, until recently such measures were deemed too complex to be considered of benefit to the average user. Maintenance of FTP servers, dynamic DNS for public IP based system shares and management of public/private keypairs for SSH are not tasks a non-technical individual can accomplish without technical assistance or a very thorough how-to guide. The current transfer options, while easy to implement and in some cases, completely transparent to the user, can all be found to have a more complex root in one of the following methods or techniques.

### 2.1 Anonymizing Services

Today there are many anonymizing services available for communication and data transfer. The popular anonymous browser Tor allows users to explore the Internet without the risk of their location or identity becoming known [Loesing et al., 2010]. The Tails operating system which works in conjunction with Tor offers an extra layer of anonymity over traditional operating systems. When a user is operating Tails all connections are forced to go through the Tor

 

network and cryptographic tools are used to encrypt the users data. The operating system will leave no trace of any activity unless explicitly defined by the user to do so.

The Invisible Internet Project, also known as I2P is another anonymous service similar to Tor. As I2P is designed as an anonymous network layer which will allow users to utilize their own applications across the network. Unlike Tor circuits the I2P network traffic utilises multiple layers of encryption and an addressing system not based on IP or ISP to provide anonymity. This process decouples a user's online identity and physical location [Timpanaro et al., 2014]. I2P also groups network messages together in irregular groupings for encryption to discourage network traffic analysis. Like Tor, I2P allows for passage through its network to a server or service not hosted within its area of influence. This is managed through the use of "Outproxies" which perform the same function as Tor exit nodes.

Both Tor and I2P provide anonymity to the user with an open network of onion routers in the case of Tor and Garlic routing in the case of I2P. These networks of routers are run by participating volunteers and it is continually growing. The result of this network growth is an increase in anonymity and privacy for each individual user [Herrmann and Grothoff, 2011]. However these services are not without drawbacks such as a severe reduction in network throughput resulting in much slower access speeds (though I2P has greater reported performance than Tor, in particular for P2P downloading protocols but it has fewer Outproxies than Tor has exit nodes resulting in a lesser degree of anonymization). Many software packages (those not SOCKS aware in the case of Tor) are not designed to correctly route through these services and will instead provide information that will potentially reveal the identity of the user such as the local, true, IP address for response traffic to be delivered to. For the end user, there is also the issue of adding yet another step to the already technical task they find themselves performing.

### 2.2 Decentralized Offsite Data Storage

The MAIDSafe network is a P2P storage facility that allows members to engage in a data storage exchange. Each member of the network enables the use of a portion of their local hard drive by other members of the network. In return the member is given the use of an equivalent amount of storage distributed across the network and replicated to multiple locations referred to as Vaults. This allows the authorized member access from any location and resilience should a portion of the network not be active at any time. All data stored on the network is deduplicated and replicated in real time with file signatures to ensure integrity. In addition the data is encrypted allowing secure storage on untrusted remote systems. Authorized access is managed through a two factor authentication process involving a password and pin combination. The use of the MAIDSafe network is incentivized through SafeCoin, a cryptocurrency that members can earn by renting out space or providing resources such as bandwidth for file transfers. Other users can earn SafeCoins by participating in development of the protocol.

### 2.3 Data Exfiltration through Standard File Transfer Channels

Data exfiltration refers to the unauthorized access to otherwise confidential, proprietary or sensitive information. Giani et al. [2006] outlines a number of data exfiltration methods including most regular file transfer methods for "inside man" attacks, e.g, HTTP, FTP, SSH and email, and external attacks including social engineering, botnets, privilege escalation and rootkit facilitated access. Detection of most of these methods is possible using a combination of firewalls and network intrusion detection systems or deep packet inspection [Liu et al., 2009, Sohn et al., 2003, Cabuk et al., 2009].

### 2.4 File Delivery Services Built on Anonymizing Networks

OnionShare is a file sharing application that leverages the anonymity of Tor to provide secure file transfers for its users. File transfers

are direct from uploader to recipient though both users utilize the Tor browser to participate. OnionShare itself is a python based application that sets up a file share on the local system as a limited web server. This web server is then advertised as a Tor Hidden Service using the built in functionality of the Tor browser. The application uses random data to generate a 16 character onion address and more random data to generate a unique name for the file being shared to use as a reference for.

The process used by OnionShare is as follows:

1. Uploader starts Tor Browser to provide an entry point for OnionShare to the Tor network.

   OnionShare is started and a temporary directory is created in the users' default temp folder. All randomly generated names in OnionShare follow the same procedure:

   (a) A number of random bytes are generated using os.random. 8 are generated for a directory/host name and 16 for the filename "slug" used to generate the file portion of the share URL

   (b) These random bytes are SHA-256 and the rightmost 16 characters of the resulting hash are carved

   (c) h is then Base32 encoded, all characters are converted to lower case and any trailing '=' signs are removed

2. The result is then used as a URL using the format `<host>.onion/<fileID>` and this is the url the Tor browser advertises to the introduction nodes and registers on DHT.

3. The uploader then sends the URL to the downloader who must use the URL within a timeframe (24 hours by default) or the signature of the file HS timestamp will not match, a process controlled by the ItsDangerous library for python. In addition to this time limit, OnionShare also utilizes a download counter which has a default value of 1. Once the number of downloads successfully initiated matches this counter, the link is no longer considered valid and all incoming URLs with the same file signature are refused.

This combination of time and availability in conjunction with the anonymity of Tor makes OnionShare traffic extremely difficult to analyze effectively. If the traffic is observed then the link is already invalidated. Similarly, if the file is discovered on a local filesystem by an investigator any trace of the once off connection to the download point, if not already lost from local connection logs or memory, will only lead to a Tor entry point and not to the actual source of the file.

# 3. INVESTIGATIVE TECHNIQUES

While no work targeted specifically at forensic investigation of these zero configuration services has been published at the time of writing, there are a number of digital evidence acquisition methods published for related services. There has , however, been security focused work published on HTML5 additions including an analysis of the `webstore` and `localstore` introduced in this version of the protocol such as that produced by Bogaard and Parody [2012]. This section outlines a number of related investigation techniques and analyses their relevancy to the forensic recovery of evidence from covert file transfer services.

## 3.1 Cloud Storage Forensics

Forensics of cloud storage utilities can prove challenging, as presented by Chung et al. [2012a]. The difficulty arises because, unless complete local synchronization has been performed, the data can be stored across various distributed locations. For example, it may only reside in temporary local files, volatile storage (such as the system's RAM) or dispersed across multiple datacenters of the service provider's cloud storage facility. Any digital forensic examination of these systems must pay particular attention to the method of access, usually the Internet browser connecting

 

to the service provider's storage access page (https://www.dropbox.com/login for Dropbox for example). This temporary access serves to highlight the importance of live forensic techniques when investigating a suspect machine as a "pull out the plug" anti-forensic technique would not only lose access to any currently opened documents but may also lose any currently stored sessions or other authentication tokens that are stored in RAM.

Martini and Choo [2013] published the results of a cloud storage forensics investigation on the ownCloud service from both the perspective of the client and the server elements of the service. They found that artifacts were found on both the client machine and on the server facilitating the identification of files stored by different users. The module client application was found to store authentication and file metadata relating to files stored on the device itself and on files only stored on the server. Using the client artifacts, the authors were able to decrypt the associated files stored on the server instance.

### 3.2 Network Forensics

Network Forensic Analysis Tools (NFATs) are designed to work alongside traditional network security practices, i.e., intrusion detection systems (IDSs) and firewalls. They preserve a long term record of network traffic and facilitates quick analysis of any identified issues [Corey et al., 2002]. Most firewalls allow HTTP and HTTPS traffic through to allow users behind the firewall to have access to regular web services which operate over these protocols. With regards to the web based covert file transfer services (outlined in detail in Section4 below), blocking network traffic to these systems would require the maintenance of a comprehensive firewall list of such servers to ensure no unauthorized data exfiltration. NFATs collecting this information will only have the ability to capture the encrypted packets, their destination and associated metadata. Identifying precisely what has been transferred will likely prove impossible for network administrators.

The issue with always-on active network forensics is dealing real-time with the large volumes of traffic involved. One approach to overcome the massive volume of network data to process is to simply record every packet sent and received from the Internet, in the similar manner to the tactic employed in the case outlined by Garfinkel [2002]. This would facilitate an after-the-fact reconstruction of any data breaches to aid in determining precisely what was compromised.

### 3.3 Deep-Web Forensics

Although Tor and I2P are designed for users to communicate and transfer data on the Internet anonymously it is still viable and possible for investigators to gather user specific information. The process of an investigation into the Tor network requires advanced digital forensic knowledge as traditional investigation methods used for standard networks fail to heed the desired results. Loesing et al. [2010] published a study measuring statistical data in the Tor network. The study is weighted towards protecting the users anonymity while using Tor but nonetheless shows that it is technically possible to gather data on Tor users by setting up a Tor relay and logging all relayed user traffic.

When users install Tor the software first connects to one of the directory authorities. The directory authorities are operated by trusted individuals of Tor and from these authorities the Tor software downloads the list of currently available Tor nodes. These nodes are relay servers that are run by volunteers of Tor. The Tor client then selects three nodes from those available and builds an encrypted channel to the entry node. An encrypted channel is then built from the entry node to the middle node, and lastly this channel connects to the exit node.

Blond et al. [2011] demonstrated the results of an attack on the Tor anonymity network that revealed 10,000 IP addresses over 23 days. The authors used the attack to obtain the IP addresses of BitTorrent users on Tor. The study found that 72% of users were specifically using Tor to connect to the tracker. The authors launched their attacks through six instrumented Tor exit nodes resulting in 9 percent of all Tor streams being traced. Moreover, the paper anal-

 

yses the type of content discovered in the attack culminating in the result that the existence of an underground BitTorrent ecosystem existing on Tor is plausible. Alongside these attacks Tor users were also profiled. Using BitTorrent as the insecure application they hijacked the statistical properties of the DHT and the tracker responses.

# 4. EVOLUTION OF HTML5 POWERED COVERT FILE TRANSFER SERVICES

While services like Onionshare provide a method of file transfer that is difficult to investigate due to its limited lifespan and shifting end points, it still requires software installation and additional support services in the form of the Tor Browser. These requirements allow for several methods of access control such as a security policy blocking local services attempting to communicate on port 9159 and 9051, the default Tor Control Ports. On an application level local, group or layer 7 firewall policies can block `tor.exe` or `onionshare.py` based on path or file hash without undue interruption to normal user network usage.

## 4.1 Basic HTML5 File Transfer

Basic HTML5 File transfer as depicted in Figure 1 is accomplished using native browser APIs that allow a user to utilize a `data transfer` object. This object consists of a customizable array of `key:value` pairs that represent a group of file objects. This associative array is then accessible by client side scripts run from a web page or web application. These scripts must first be downloaded and allowed to run by the local user (this depends on the trust setting for the website being visited). Any element can be added to the array through a Drag and Drop (DnD) functionality or files can be added though a file browser interface. The actions available by default are:

- copy: A copy of the source item may be made at the new location.

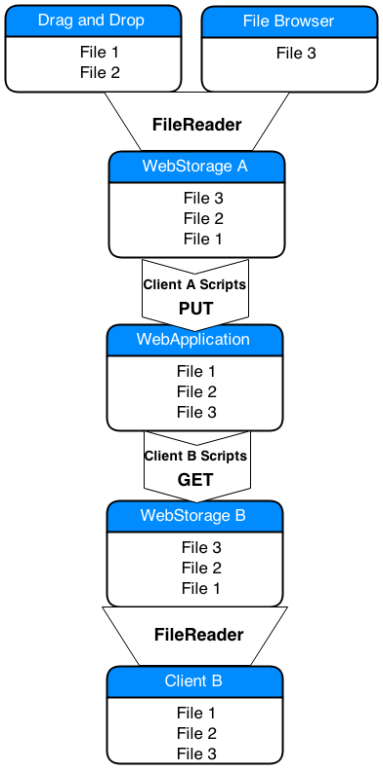


Figure 1: The Basic HTML5 Data Transfer Process

- move: An item may be moved to a new location.
- link: A link may be established to the source at the new location.
- copyLink: A copy or link operation is permitted.
- copyMove: A copy or move operation is permitted.
- linkMove: A link or move operation is permitted.
- all: All operations are permitted.
- none: The item may not be dropped

if the element added to the array is a file then the element is passed to a `FileReader` object that copies the data contained in the file to `localstorage` or `session storage` depending on the settings of the web application. Local Storage is shared across all browser sessions

 

currently active, session storage is only available to the owning application or window (for browsers with multiple tabs or windows). This local/session storage behaves very similarly to the standard cookie storage but with hugely increased capacity. ( 5MB for Chrome, Firefox and Opera, 10MB for Internet Explorer - DnD native is only available in version 9+ of IE - web storage in version 8+, and 25MB for Blackberry).

For basic data transfer, the `Filereader` reads the entire file indicated into RAM for processing. Once stored in web storage a file can only be accessed by local actions that have permission to access that web storage area such as client side scripts downloaded from the controlling web page or session. These scripts can use any scripting language but usually JQuery, JavaScript or AJAX. The local client can also call scripts to run on the remote server in order to pass variables or prepare for the establishment of additional sessions are required.

### 4.2 Cryptographically enhanced HTML5 data channels

Following on from their acquisition of ON2 in February 2010, Google continued to develop a browser to browser data transfer protocol, which was made open source in 2011 when it was adopted by W3C as a standard for HTML5. The protocol, which supported Real Time Communication between browsers was released as WebRTC 1.0, was developed to provide P2P voice, video and data transfers between browsers without an additional software requirements. WebRTC, provides a collection of protocols and methods as well as a group of codec libraries that can be accessed via a JavaScript API.

WebRTC improved data transfer over the standard HTML5 script based method by introducing data integrity, source authentication and end to end encryption. This is accomplished through the use of Datagram Transport Layer Security (DTLS) Modadugu and Rescorla [2004] extension to handle key exchange for the Secure Real-time Transport Protocol (SRTP). The use of DTLS-SRTP differs from standard VOIP encryption by removing the need to trust SIP relays that form the path between the source and destination.

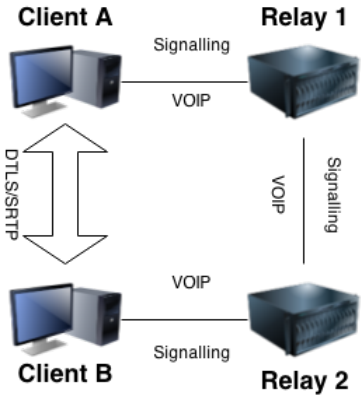


Figure 2: Traditional VOIP Data vs DTLS-SRTP

In Figure 2, the standard VOIP method of communication is displayed alongside the newer WebRTC method. Both systems start with establishing a signaling and control path to handle non-sensitive data such as connection auditing packets. In VOIP, the data stream would follow this established path and traffic between Client A and Client B involving relay through Relays 1 and 2. Unless the user fully trusts both relays and the security of the network path between each node on the network, there is a risk of an adversary eavesdropping on the data stream and either manipulating the content in transit or capturing it for offline inspection.

## 5. ANALYSIS OF EXISTING SERVICES

Services such as those presented in Table 1 are a sample set of HTML5 and WebRTC based file transfer utilities. While at first glance many of these applications appear to be homogenous closer examination shows important differences in both capabilities and requirements. Of the services listed, only Sharefest and JustBeamIt allow usage without local installation or some form of authentication. Of these, JustBeamIt is based on basic HTML5 file transfer while Sharefest utilizes WebRTC. While Sharefest purports to offer persistent storage, it does so

| Service Name | *Encrypted Transfer* | *HTML Based* | *Registration Required* | *Application Option* | *Anonymity* | *Mobile Compatibility* | *Persistent Storage* | *Relay Server* |
|---|---|---|---|---|---|---|---|---|
| Sharefest | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ |
| JustBeamIt | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ |
| Transfer Big Files | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ |
| Infinit | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ |
| Any Send | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ |
| Rejetto | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ |
| QikShare | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ |

Table 1: Comparison of Browser Based Transfer Services

by virtue of its distributed organization. Storage is only available as long as at least one share member is online. While many of the services do not provide encrypted file transfer, the threat posed lies in the fact that any illegal or illicit data transfer can be difficult, if not impossible, to differentiate from standard web traffic.

### 5.1 HTML5 Enabled Peer-to-Peer Transfer

This section examines a number of HTML5 enabled P2P transfer sites and describes their operation.

### 5.2 Sharefest

Sharefest.me is a file-sharing "one-to-many" based website that aims to dynamically generate and maintain file-sharing swarms by connecting peers that are interested in sharing the same data. Like the BitTorrent protocol, multiple peers are utilized simultaneously to transfer portions of the data thus increasing download speeds by avoiding the bottleneck that is the lower upload speed of a standard ADSL Internet connection. In order to achieve this, Sharefest is built on Peer5's (`https://peer5.com/`) platform for a distributed Internet, a P2P data transfer mesh network that utilizes the capabilities of the browser without additional plugins beyond a WebRTC capable browser.

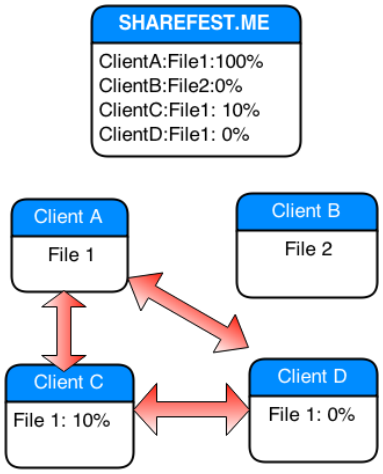


Figure 3: Sharefest P2P Mesh over WebRTC

As depicted in Figure 3, the Sharefest process is quite straightforward in design. The sharefest.me server acts as a transfer control server that records all files being offered for sharing and matches the resource to the client system request. In the scenario depicted, Client A has a complete file that it wants to share. Client A connects to the Sharefest server at `https://www.sharefest.me/` over port 443 and negotiate TLS1.2 where possible using SPDY if available for web content transfer. Given a full

 

range of options the Sharefest server negotiates the use of the ECDHE-ECDSA with AES 128 and GCM 256. As required by the IETF RFC 4492 (`http://tools.ietf.org/html/rfc449`), the Sharefest server passes the curve details as part of its `serverkeyexchange` packet.

Once a secure path is established the server delivers a small set of helper scripts to the client:

- files.js : a script to gather file details from the client
- ui.js : a script to control the update and display of the file management interface on the page.

Once a file has been selected for sharing the Sharefest server assigns a code value in the form of a URL such as `https://www.sharefest.me/67509cb244257b6643540dda512f8171` where the number after the domain name is the swarmID for this file. The SwarmID has 32 characters but is not based on the MD5 of the file, instead it appears to be derived from the Sharefest crypto.js script which incorporates SHA3 in a lengthy calculation. The swarmID is deterministic meaning that any client sharing the same file will identified with the same swarmID.

Once clients offering and requesting the same file or file set are identified the Sharefest server acts as a central traffic control and initiates a STUN connection directly between the participating clients. In Figure 3, Clients A, C and D are all participating in the swarm for File 1, but Client B is not involved. The STUN connection consists of each pair of clients issuing BIND requests from uploader to downloader using STUN over UDP which allows each host to discover its own public facing IP address in order to create end to end connections through a NAT gateway or firewall. Each BIND / Confirm pair is re-issued on a regular, short, interval to ensure the connection remains intact. Once the STUN session is active the two peers negotiate a WebRTC session and switch over to the protocol's encryption. ACK and STUN confirmation messages continue to be sent to and from the Sharefest server and the peers throughout the exchange.

Sharefest is an example of P2P privacy in a distributed network ensuring that data can be transferred without risk of interception. This level of privacy comes at a cost though as the ability of IT security to inspect the traffic is greatly diminished with the level of encryption in use at all stages of the transfer. Packet analysis can detect the IP addresses in use but without access to the key to decrypt the traffic the content being transferred is extremely difficult to determine. One option available to network admins is to block the use of the sharefest.me service by blacklisting the URL. This would have the effect of preventing casual usage of the service but the source for Sharefest is publicly available on Github `https://github.com/Peer5/Sharefest` along with instruction and support for installation of a personal server. Peer5 also provide the API key for free to anyone interested in the code. This means that any IP or URL could become a Sharefest server.

One method of detecting the use of this application is the STUN traffic generated once a peer is identified and connection is initiated. In testing an average of 5 STUN negotiation/confirmation exchanges were recorded every second depending on the level of file transfer data passing between the peers. This level of "noise" would make the user of Sharefest relatively easy to discover and no effort is made to obfuscate the communicating peers.

In an attempt to determine if this lack of anonymization could be overcome, we attempted to run Sharefest through a Tor circuit but both transfer utilities (JustBeamIt and Sharefest) failed to complete the initial negotiation with the relevant server. This was tested using Tor Browser installed on a Windows 7 VMWare image. A possible alternative may be to attempt the use of a SOCKS aware proxy to direct the traffic to and from the application. Alternatively a server running Sharefest could be adapted to run as a Tor Service but this would not alleviate the lack of privacy experienced once data transfer was initiated between peers.

 

### 5.3 Transfer Big Files

Transfer Big files (TBF) `https://www.transferbigfiles.com/` offers drag and drop transfer of files with size limits dependant on the user's account type. At the most basic free account level, the limit is 100mb per file totalling up to 20gb of files awaiting transfer at any one time. Files are held by the server for a default of 5 days before they are removed. While this service does offer HTML5 drag and drop upload capability, it is not a true P2P application in that it does not transfer directly from one local peer to one or more remote peers. Some features of TBF include:

- senders must have an account to avail of the service. This account requires a name, email address and a password though of note, there is no verification of any of these fields
- Recipients can either have their own account or can be sent a shortened URL alias link with the domain tbf.me and a short code for the file itself
- The initial DNS lookup for Transferbigfiles.com resolves to an IP address 69.174.247.183 - a server hosted in the US
- This account server negotiates TLS 1.0 as part of the initial handshake
- File upload prompts DNS lookup of 0storageuk4.transferbigfiles.com and 1storageuk4.transferbigfiles.com which both resolve to 83.222.233.155 (a server group hosted in the UK)
- once uploaded the files remain on the storage servers until picked up by the receipient who must be notified separately
- File download is performed from one of the tbfuk4.transferbigfiles.com group servers over standard HTTPS.
- TBF also offers an application and a command line client with enhanced capabilities over the browser based interface but this is beyond the scope of this paper

It is worth noting that while this transfer service is just standard HTTPS and so will be discovered by standard forensic practices, if the username and password can be recovered, the site's account activity summary contains a full activity log of all files uploaded including when the transfer was started and when the recipient collected the file as well as the recipient's account name.

### 5.4 JustBeamIt

An example of a basic HTML5 transfer application is the file transfer service offered at `http://www.justbeamit.com`. The sending user connects to the server over port `8080` (alternate HTTP port) and performs a standard TCP handshake followed by a series of `HTTP GET` requests for client side JavaScripts.

- JustBeamIt.js - the base script that sets up the variables and defines the communication functions
- BrowserDetectUtility.js - determines if the user browser can properly support HTML5 data transfers
- FileHandler.js - manages the transfer to and from the file array. Handles the array reset and webpage notifications if the array is emptied.
- UploadManager.js - defines the drag and drop actions and defines the "landing zone"
- UploadHandler.js - Determines if the Client needs to use XMLHttpRequest (XHR) or FORM based uploading and generates the QRCode.
- UploadHandler.XHR.js and UploadHandler.FORM.js - the actual uploading scripts

There is an option to drag and drop a file into the browser but in this instance the file browser is used to select a file from the local user pictures folder. Once selected the button "Create Link" is clicked and the link `http://www.justbeamit.com/di33x` is created along with a QRCode for mobile use. This link can copied and sent to

 

the receiving system, in the meantime the local client is redirected to a relay server URL for the upload itself (`http://b1.justbeamit.com/`. On the remote system, the URL is pasted into a browser and the system and the remote client loads `http://www.justbeamit.com` and immediately requests the download token for the file ID `di33x` and downloads and runs the set of JavaScripts. The server passes along the token along with the current download status (upload waiting) and the file descriptor (name, size, extension). Once the remote user clicks on the link to download the browser is redirected to `http://b1.justbeamit.com` where the file transfer is performed. Once complete the local user is notified that the transfer has been successful. The token used to download is now invalidated and a new link must be generated if the file is to be shared again. Similarly, there is a 1000 second timeout period during which the shared file must be downloaded before the opportunity expires and a new share token must be generated.

While this method of file transfer provides ease of use to the end user, all transfers are performed via unencrypted traffic. The data being transferred is susceptible to any form of eavesdropping capable of detecting traffic on any network segment the traffic passes through. The open nature of the transfer and the client side execution of scripts (as well as the open exchange of tokens) allow for trivial man-in-the-middle (MITM) attacks where an adversary capable of eavesdropping can use a proxy or other interception utility to alter the packets in transit. One possible scenario would be the substitution of a harmless UploadManager.js script for something less benign as identified by Jang-Jaccard Jang-Jaccard and Nepal [2014] as a rising risk or, even exchanging the generated download token for one that leads to a virus or other form of malware.

From a security standpoint, defense against the use of this service is quite straightforward. Because of the application's use of a centralized set of servers, a standard firewall rule to block access to `http://*.justbeamit.com` would prevent any upload but also any attempt to download.

### 5.5 Any Send

Any Send `http://www.anysend.com/` is a web based file transfer utility that offers a pure web based alternative to its own downloadable application. The web page anysend.com resolves to a set of four IP addresses divided between Utah and Texas and all registered to clickmein.com. The webpage consists of a large background image and a single "dropzone" for files to be sent. On dropping a file, the background javascript helpers manage the file upload to their servers. Once uploaded the user is presented with a URL to send to the recipient. The URL is comprised of the anysend.com domain and a file identifier generated by the server (a 32 character string). Once the recipient enters the URL into a browser they are taken to an anysend.com page with details of the file corresponding to the fileID part of the URL used. From here it is a standard HTTPS download from the anysend servers and the download URL will reflect the anysend server url as well as the full filename of the file with the term "%20via%20AnySend.exe" appended. eg: `http://www.anysend.com/dl.php?data=7f234e0920d8424833f97d3ab9380883\...\&fn=orientdb-community-2.0.4\%20via\%20AnySend.exe\&packageID=ED59EE58A0380BBDB197A88F8290BDDE`

## 6. FORENSIC CONSEQUENCES OF "UNTRACEABLE" FILE TRANSFER

The facilitation of "untraceable" or "anonymous" file exchange can lead to a number of potential malicious use cases. For each of the scenarios outlined below, an added dimension can be created by the originator of the content: time. Due to the ability to create "one-time" or temporary access to any piece of content, the timeframe where evidence may be recovered from remote sharing peers might be very short.

### 6.1 Cybercriminal Community Backup

Unmonitored covert transfer could be used to create a "share and share alike" model for the remote encrypted backup of illegal content. The sharing of these backups onto multiple remote machines effectively could provide the user with a cloudless backup solution requiring minimal trust with any remote users. The encryption of the data before distribution to the community can ensure that only the owner will ever have access to decrypt the data. Trust only comes into play should the remote nodes delete the information or it otherwise becoming unrecoverable. Having a secure, encrypted connection to a remote backup might be desirable to cybercriminals enabling the use of a kill-switch to their local storage devices should the need arise.

### 6.2 Secure Covert Messaging

For example, the proof of concept based on the BitTorrent Sync Protocol found at `http://missiv.es/`. The application currently operates by saving messages to an "outbox" folder in a synchronized share between peers that has a read only key shared to the person you want to receive the message. They in turn send you a read only key to their outbox. One to many can be achieved by sharing the read only key with more than one person but no testing has been done with synchronization timing issues yet and key management may become an issue as a new outbox would be needed for each private conversation required.

### 6.3 Industrial Espionage

Many companies are aware of the dangers of allowing unmonitored traffic on their networks. However, quite often corporate IT departments enforce a blocking of P2P technologies through protocol blocking rules on their perimeter firewalls. This has the effect of cutting off any file-sharing clients installed on the LAN from the outside world. In addition to Deep Packet Inspection (DPI) to investigate the data portion of a network packet passing the inspection point, basic blocking of known IP address blacklists in firewall rulesets can be used. The difficulty in blocking HTTP based file transfers is that the technology is likely used during regular employee Internet usage. HTTP transfers can be used when emailing file attachments or adding items to content management system. One additional scenario where these services could be used would be to transfer files within a LAN and subsequent external exfiltration from a weaker/less monitored part of the network, e.g., guest wireless access.

### 6.4 Piracy

Like any other P2P technology, the ability to transfer files in a direct manner from peer to peer lends itself well to the unauthorized distribution of copyrighted material. The sharing of copyrighted multimedia, software, etc., between peers using these covert services is less likely to lead to prosecution compared with public piracy on open file-sharing networks such as BitTorrent.

### 6.5 Alternative to Server Based Website Hosting

This scenario involves the creation of static websites served through a shared archive. These websites could be directly viewed on each user's local machine facilitating the easy distribution of any illegal material. The local copies of the website could receive updates from the "webmaster" through the extraction of archived updated distributed in a similar manner as the original.

## 7. POTENTIAL FORENSIC INVESTIGATION TECHNIQUES

Assuming access (physical or remote) can be acquired to either end of the file transfer, then a live acquisition of the evidence should be attainable. Performing evidence acquisition after the fact would rely on traditional hard drive and memory forensic techniques to see if any remnants of the network communication remain.

The investigation of the unauthorized transfer for information through one of these services without access to either end of the transfer can prove extremely difficult. Assuming

 

through some external means, the precise date and time of the transfer were discovered. The only method available to law enforcement is to effectively wiretap the transfer by running a software or hardware based deep packet inspection tool on the network at either end of the transfer.

To date there has been keen interest in research performed on the forensic examination of file sharing utilities and the type of security risks they pose. Chung et al. [2012b] outlined a best practice approach to the investigation of file sharing using cloud based Storage-as-a-Service (StaaS) utilities such as Dropbox, iCloud and OneDrive. In 2014, Federici [2014] presented Cloud Data Imager (CDI) a utility developed to automate the retrieval of cloud based storage artifacts from a suspect system and use these credentials to access their secure storage online.

Scanlon et al. [2014] described a methodology that leveraged the processes used by persistent file synchronization services to ensure data integrity to retrieve data that would otherwise have been inaccessible. This could be as a result of deliberate obfuscation such as encryption or anti-forensic activities or it could be caused by an error in the imaging process. The methodology presented utilized the need for synchronization group ongoing communication to enumerate remote peers and to identify any authorized peers that could provide a forensically true copy of the suspect data.

Emerging file transfer utilities, such as the purely browser based file transfer utilities based on WebRTC, do not advertise persistence of availability nor integrity checking beyond the initial transfer and in many cases are only associated for the length of time that both parties are online and in communication, directly or otherwise. After this time, such as with Onionshare for example, the address of the file source will change completely and no longer be available to any peer authorized or otherwise.

This ephemeral nature of data transfer can make any attempt to verify or re-create the circumstances of the file transfer difficult if not impossible and it very much depends on the features of the individual application being employed.

# 8. CONCLUSION

The evolution of online file transfer systems is becoming more and more covert through employing encryption-based, server less, P2P protocols. The development of HTML5 and JavaScript based services proves particularly interesting from a digital forensic perspective. As these technologies mature, a future can be easily envisioned whereby the investigation and evidence retrieval from these systems will prove extremely difficult, if not entirely impossible. Daisy-chaining a number of the technologies outlined in this paper has the potential to enable malicious users to securely transfer any desired information to another user/machine without arousing suspicions of system administrators. Identifying the use of a HTTPS, browser-based P2P file transfer with relatively small transfer sizes might prove prohibitively difficult. The investigation of these transfers may prove cost prohibitive in terms of both time and money for law enforcement to comprehensively investigate and mitigation of the risk by way of security policies or hardware and software based rulesets may come at a price in terms system usability that will be deemed too high.

## 8.1 Future Work

As privacy becomes easier for the end user to accomplish, the role of forensics will become all that much harder as not even the "low hanging fruit" of browser history can be expected as a starting point. Additionally, system security will no longer be able to react as this may already be too late. As a future developement in forensics there is clear potential for utilities and techniques to be developed to help bridge the gap between proactive security abnd reactive forensics. Some areas of interest are

- Automated Detection of HTML5 and WebRTC based Data Exfiltration.
- Approximate Hashing Signatures – Approximate hashing facilitates the analysis of network traffic as it could be applied to

recognise variations on patterns specific to protocols and their timings used in HTML5 and WebRTC

- Forensic analysis of P2P over anonymizing networks. Perhaps a lack of a footprint can be proven to be a footprint in and of itself in a networking environment.

## REFERENCES

Stevens Le Blond, Pere Manils, Chaabane Abdelberi, Mohamed Ali Dali Kaafar, Claude Castelluccia, Arnaud Legout, and Walid Dabbous. One bad apple spoils the bunch: exploiting p2p applications to trace and profile tor users. *arXiv preprint arXiv:1103.1518*, 2011.

Daryl; Bogaard, Daniel; Johnson and Robert Parody. Browser web storage vulnerability investigation: Html5 localstorage object. In *Proceedings of The 2012 International Conference on Security and Management*, 2012.

Serdar Cabuk, Carla E Brodley, and Clay Shields. Ip covert channel detection. *ACM Transactions on Information and System Security (TISSEC)*, 12(4):22, 2009.

Hyunji Chung, Jungheum Park, Sangjin Lee, and Cheulhoon Kang. Digital forensic investigation of cloud storage services. *Digital Investigation*, 9(2):81 – 95, 2012a. ISSN 1742-2876.

Hyunji Chung, Jungheum Park, Sangjin Lee, and Cheulhoon Kang. Digital forensic investigation of cloud storage services. *Digital investigation*, 9(2):81–95, 2012b.

Vicka Corey, Charles Peterman, Sybil Shearin, Michael S Greenberg, and James Van Bokkelen. Network forensics analysis. *Internet Computing, IEEE*, 6(6):60–66, 2002.

Corrado Federici. Cloud data imager: A unified answer to remote acquisition of cloud storage areas. *Digital Investigation*, 11(1):30 – 42, 2014. ISSN 1742-2876. http://dx.doi.org/10.1016/j.diin.2014.02.002. URL `http://www.sciencedirect.com/science/article/pii/S174228761400005X`.

Simson Garfinkel. Network forensics: Tapping the internet. *IEEE Internet Computing*, 6: 60–66, 2002.

Annarita Giani, Vincent H Berk, and George V Cybenko. Data exfiltration and covert channels. In *Defense and Security Symposium*, pages 620103–620103. International Society for Optics and Photonics, 2006.

Michael Herrmann and Christian Grothoff. Privacy-implications of performance-based peer selection by onion-routers: a real-world case study using i2p. In *Privacy Enhancing Technologies*, pages 155–174. Springer, 2011.

Julian Jang-Jaccard and Surya Nepal. A survey of emerging threats in cybersecurity. *Journal of Computer and System Sciences*, 80(5):973 – 993, 2014. ISSN 0022-0000. http://dx.doi.org/10.1016/j.jcss.2014.02.005. URL `http://www.sciencedirect.com/science/article/pii/S0022000014000178`. Special Issue on Dependable and Secure Computing The 9th {IEEE} International Conference on Dependable, Autonomic and Secure Computing.

Yali Liu, Cherita Corbett, Ken Chiang, Rennie Archibald, Biswanath Mukherjee, and Dipak Ghosal. Sidd: A framework for detecting sensitive data exfiltration by an insider attack. In *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*, pages 1–10. IEEE, 2009.

Karsten Loesing, Steven J Murdoch, and Roger Dingledine. A case study on measuring statistical data in the tor anonymity network. In *Financial Cryptography and Data Security*, pages 203–215. Springer, 2010.

Ben Martini and Kim-Kwang Raymond Choo. Cloud storage forensics: owncloud as a case

 

study. *Digital Investigation*, 10(4):287 – 299, 2013. ISSN 1742-2876.

Nagendra Modadugu and Eric Rescorla. The design and implementation of datagram tls. In *NDSS*, 2004.

Mark Scanlon, Jason Farina, Nhien-An Le Khac, and M-Tahar Kechadi. Leveraging Decentralisation to Extend the Digital Evidence Acquisition Window: Case Study on BitTorrent Sync. *Journal of Digital Forensics, Security and Law*, pages 85–99, September 2014.

Taeshik Sohn, JungTaek Seo, and Jongsub Moon. A study on the covert channel detection of tcp/ip header using support vector machine. In *Information and Communications Security*, pages 313–324. Springer, 2003.

Juan Pablo Timpanaro, Isabelle Chrisment, and Olivier Festor. Group-based characterization for the i2p anonymous file-sharing environment. In *New Technologies, Mobility and Security (NTMS), 2014 6th International Conference on*, pages 1–5. IEEE, 2014.

 

 