Edith Cowan University

Research Online

Australian Digital Forensics Conference

Conferences, Symposia and Campus Events

1-1-2011

Can current packet analysis software detect BitTorrent activity or extract files from BTP and µTP traffic streams?

William Pung
Edith Cowan University

Andrew Woodward
Edith Cowan University

Follow this and additional works at: https://ro.ecu.edu.au/adf

Part of the Computer Sciences Commons

DOI: 10.4225/75/57b2c14440cf0
9th Australian Digital Forensics Conference, Edith Cowan University, Perth Western Australia, 5th -7th December 2011

This Conference Proceeding is posted at Research Online.
https://ro.ecu.edu.au/adf/100

CAN CURRENT PACKET ANALYSIS SOFTWARE DETECT BitTORRENT ACTIVITY OR EXTRACT FILES FROM BTP AND µTP TRAFFIC STREAMS?

William Pung, Andrew Woodward
secau Security Research Centre, School of Computer and Security Science
Edith Cowan University, Perth, Western Australia
wpung@our.ecu.edu.au; a.woodward@ecu.edu.au

Abstract

BitTorrent is a peer to peer file sharing protocol used to exchange files over the internet, and is used for both legal and illegal activity. Newer BitTorrent client programs are using proprietary UDP based protocols as well as TCP to transmit traffic, and also have the option of encrypting the traffic. This network forensic research examined a number of packet analysis programs to determine whether they could detect such traffic from a packet captures of a complete file transmitted using one of four protocol options. The four states examined were: TCP without encryption, TCP with encryption, µTP without encryption and µTP with encryption, and the six programs investigated were: Network Miner, Tcpxtract, Honeysnap, OpenDPI, Netwitness Investigator and SPID. Of the six programs investigated, none of them were fully able to fully reconstruct a file, with most not even able to detect that the traffic related to BitTorrent usage. The Netwitness Investigator program was able to extract the announce and scrape files. The signature based SPID was able to partly match TCP based torrent traffic, but could not identify µTP traffic. The conclusion is that until new tools are developed, forensic investigators must continue to rely on artifacts created by the BitTorrent clients themselves in order to locate evidence in the event that a crime has been alleged.

Keywords

BitTorrent, packet analysis, deep packet inspection, µTP

INTRODUCTION

BitTorrent technology modified the *modus operandi* of peer to peer (P2P) file sharing by decentralizing the server or central connecting node to the users themselves. Previously, P2P file sharing applications such as Napster were based on a client-server architecture which relied upon indexing servers to locate files (Hughes & Lang, 2005). Whilst some of this activity is legal, there is a percentage of P2P traffic which trades in material which is considered both illegal and unwanted, such as child exploitation images (Thomas, J.T., personal communication, October 11 2011). The major advantage to law enforcement and other authoritative bodies of these technologies is that these servers are fairly easily identified and neutralised and thus, not a viable means to share files such as unlicensed media or other malicious software. It was inevitable that a replacement technology would emerge to make the task of censoring or stopping such activity more difficult. BitTorrent operates on a pure client-client platform that does not breakdown due to nodes being shut down by governing bodies, and has since been the file sharing application of choice for many users for some time. BitTorrent clients log all their activity and store significant evidence in relation to downloaded and shared files, and physical seizure of a suspect computer will yield such evidence (Woodward, 2008; 2009). There has also been research conducted into discovering forensic evidence the purposes of copyright infringement for P2P file sharing, but this also relies on interrogation of the user device (Kokkinen & Noyranen, 2009). There appears to be no evidence in the literature in relation to exploring the traces that the BitTorrent traffic may reveal, nor is there evidence of the files themselves being extracted from a network stream through packet capture and reassembly.

The objective of this paper is to determine whether common network tools and well known network monitoring suites have the ability to correctly identify and/or reconstruct data based on PCAPs containing full downloads of a certain file. Results may vary from tool to tool as some of them do not support torrents fully, while some of them claim to do so. This paper looks at traditional TCP BitTorrent packets and the new µTP in terms of network forensics to determine files can be carved from captured packets or packet history. Also, the effect of encryption on both streams of traffic and how it obscures file mining was investigated. A total of four PCAP files were produced, and these files were then scrutinised using various well known applications and results studied and analyzed for two main characteristics: detection of BitTorrent activity and the ability to carve data out of these PCAP files.

BITTORRENT BACKGROUND

The BitTorrent protocol, also known as BTP, was conceived with the objective of relieving uploading load on file servers serving large files, by distributing it across its simultaneous downloaders. This was based on the belief that clients do not utilize their upload capacity while downloading a file, hence not putting the client at any disadvantage. Furthermore, this protocol allows efficient file publishing without the support of a large infrastructure, enabling smaller organizations to publish with minimal overhead or administrative burden (Fonseca, Reza, Fjeldsted, & DIKU, 2005).

Download methodology

The BTP specification states that torrent content is divided into pieces and is simultaneously requested and received via a piece selection strategy, which requests for the rarest pieces from the cloud to reduce wait time on missing pieces. (Fonseca, et al., 2005)This technique effectively shuffles up the content's download sequence and receives them in parallel, rather than a traditional FTP sequential file transfer in circuit. This is akin to parallel fetching of a file from a FTP server via multipart downloading; just that BTP has increased complexity due to multichannel and multipart weaved into each other and its file source are peers from the cloud.

Encryption

In the light of severe bittorrent traffic throttling on ISP ends, the developers of Azureus (now known as Vuze), designed and implemented an encryption standard to obfuscate and encrypt traffic to conceal itself from prying listeners. This is further rectified and implemented into Vuze and uTorrent as 'Protocol Encryption (PE)'. PE primarily utilizes Diffie-Hellman public key cryptography for encryption, under RSA's standard. (Mennecke, 2006)

µTP specification

µTorrent transport protocol (µTP), introduced in June 2009 was an alternative protocol to TCP when dealing with bittorrent. The significant change lies in the utilization of UDP instead of TCP that BTP utilizes. µTP was introduced with efforts of not disrupt internet connections, while still utilizing the unused bandwidth fully when downloading files via bittorrent. (Norberg, 2009)

Effects on file mining

Since the conception of BTP, network forensics on bittorrent traffic has been a stalemate. There have been no new tools to fully explore the potentials of data recovery via packet captured files. Since files have been transported in jumbled sequences and multiple sources chipped into the construction of a single file, traditional network mining tools have not been able to reconstruct this traffic type, nor are they able to correctly identify them. Moreover, clients have the option to encrypt their traffic with PE, further evading scanners and file miners.

MATERIALS AND METHODS

Materials

A number of network forensics tools have been gathered to analyze packet captures. These tools were selected as they are commonly used and provide a range of different characteristics in terms of their analysis engines. All of the tools, with the exception of OpenDPI have one function in common, that is the ability to read PCAP files and extract files from it. The six programs usd in this research were as follows:

Network Miner V1.0 (http://sourceforge.net/projects/networkminer/)

This Windows based GUI tool offers simplicity in its usage, the application will process the loaded PCAP file and display any information that it discerns. Network miner is capable of file reassembly via deep packet inspection (DPI), this added advantage may be of use in the experiment.(Hjelmvik, 2009a)

Tcpxtract 1.0.1 (http://sourceforge.net/projects/tcpxtract/)

This Unix based tool captures all data streams within a PCAP file, and splices them out, dumping data in sequence of discovery.

Honeysnap 1.0.7 (http://www.honeynet.org/project/Honeysnap/)

This is a tool spawned off the Honeynet project that parses PCAP files and identifies significant events within the processed data. It creates a folder named analysis and dumps parsed results into it, including dumps and statics such as flow contents, incoming/outgoing TCP and UDP connection logs.

OpenDPI 1.2 (http://www.opendpi.org/)

OpenDPI is a deep packet inspection (DPI) engine released as open source by a network security company 'IPOQUE'. As stated in its Protocol and Application Classification Engine's (PACE) white paper, it supports the following protocols and applications that are of concern in this experiment (ipoque GmbH, 2011).

- BitTorrent
- BitTorrent-plain
- BitTorrent-encrypted
- µTorrent 2.2

Netwitness Investigator 9.5.5.6 (http://www.netwitness.com)

Carrying on with DPI technology, Netwitness has released a free version of their award winning Investigator. Investigator is an interactive threat analysis application of the Netwitness enterprise of network monitoring platforms (EMC Corporation, 2011).

SPID Algorithm Proof of Concept Application

(http://sourceforge.net/apps/mediawiki/spid/index.php?title=Main_Page)

This proof of concept open source tool compares captured traffic against pre calculated models in order to classify or categorise the traffic according to its protocol. This program uses a fingerprint system, similar to other pattern based algorithms used in intrusion detection and antivirus software (Hjelmvik, 2009b).

Method

A Windows XP SP3 machine will be prepared in a form of a Virtual Machine, with µTorrent installed in it.

Steps

1. Wireshark will be capturing all traffic from the virtual machine
   a. Capture filter set to VM's address only
2. On the virtual machine, OpenOffice will be downloaded via µTorrent with settings as below
   a. TCP without encryption
   b. TCP with encryption
   c. µTP without encryption
   d. µTP with encryption
3. Separate packet captures from Wireshark will be saved for each permutations of Step 2.
4. These PCAP files were then analysed using each of the forensic tools listed in Table 1

Standalone tools with single operation usage such as Network Miner, TcpXtract will be used first, for its lack of complexity in obtaining results. These tools will provide the base coverage and knowledge on the PCAP files obtained through a linear workflow.

Next, complete analysis toolkit Honeysnap will be used to gain more knowledge from different points of views.

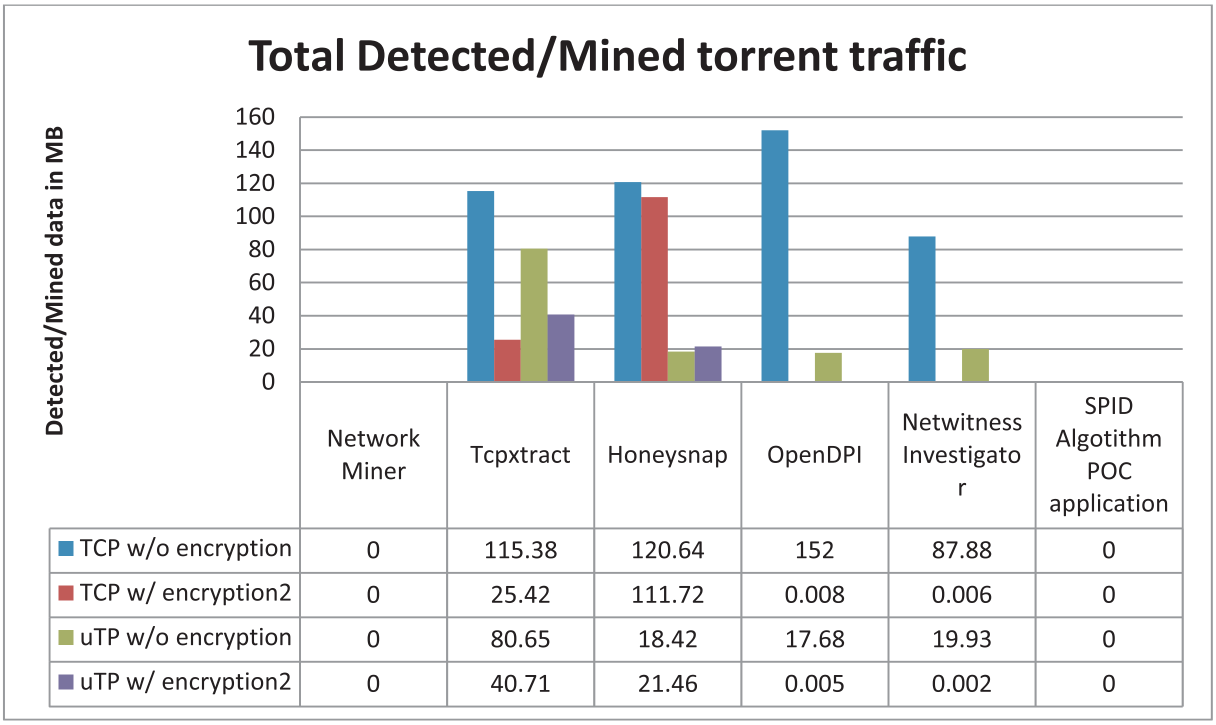
Lastly, tools and suites providing Deep Packet Inspection will be used. This will shed light on the internals of the PCAP, especially those that are encrypted.

RESULTS

As can be seen in Figure 1, SPID proved to be very close to its DPI counterparts in identifying torrent streams, despite not being able to calculate the data behind its identified streams. However, no tool was able to detect torrent streams when data obfuscation (encryption) or usage of UDP (µTP) was employed.

Data analysis reveals that the most detected/mined traffic is of TCP without encryption, followed by TCP with encryption. The protocol µTP without encryption was the third most detectable, and µTP with encryption was the least detectable protocol. This proves that most tools are tuned to seek out and tag TCP traffic with specific signatures as bittorrent protocol. However, efficiency plunges when an entirely different protocol is used, although still adhering to the general Bittorrent Protocol. Data obfuscation via encryption worsens the detection rate even further, dropping to near 0 in cases of DPI and SPID algorithms.

All these statistics does not seem to matter, when reaching the apex of this experiment. None of these tools offer a completely unscrambled or decoded file. Although entire PCAPs were captured with a full download of a torrent file, these tools were unable to reconstruct its contents. At best, the tools were able to identify but a chunk of the required raw data, further arrangement of data pieces was not possible.



	Network Miner	Tcpxtract	Honeysnap	OpenDPI	Netwitness Investigato r	SPID Algotithm POC application
🟦 TCP w/o encryption	0	115.38	120.64	152	87.88	0
🟥 TCP w/ encryption2	0	25.42	111.72	0.008	0.006	0
🟩 uTP w/o encryption	0	80.65	18.42	17.68	19.93	0
🟪 uTP w/ encryption2	0	40.71	21.46	0.005	0.002	0

Figure 1: Summary of results for an investigation into the ability of six packet analysis programs to extract complete files from a BitTorrent packet capture file.

Network Miner

It was observed that this tool was unable to grasp the internals of the provided PCAPS. However, it was able to pick up other characteristics such as DNS queries, Host addresses and total number of sessions. No recognizable files have been identified by Network Miner; hence none are displayed in the Files tab. Unfortunately, this tool is not able to display streams of data. It is possible that although it detects raw data in its TCP/UDP streams, it is unable to correctly thumbprint the files, therefore it did not display any.

Tcpxtract

Tcpxtract was unable to construct a single meaningful file from the provided PCAP files. It appears that these mined files were extracted from various streams of TCP traffic without any effort to piece them together (No support for torrent traffic). This simple tool reconstructs files based on simple file signatures from single streams. Therefore it was not coded to parse torrent traffic at all, additionally, all carved files contained raw data and had been wrongly assigned extensions.

Honeysnap

From the sample extract of a file dump, it is observed that although data flows can be identified by Honeysnap it is also, like tcpxtract, unable to decipher torrent traffic. This results in large numbers of file when dumped. Moreover, it is also observed that encryption obfuscates much data, causing less number of recognizable data streams.

OpenDPI

Test results put the white paper’s claims to the test indeed; only TCP without encryption was identified, while the other 3 have slipped past its scanners unscathed. This shows that so far, only Deep Packet Inspection has managed to correctly identify bittorrent traffic, let alone uTP, which remains fully obscure. However, with encryption, even TCP is able to evade this DPI’s engine.

SPID Algorithm Proof of Concept Application

This application does nothing more than to identify protocol within a PCAP file. Although simplistic looking, this algorithm is praiseworthy as it does not perform DPI to differentiate files. SPID determines a PCAP’s protocol based on pattern matching of its headers and flow; as depicted in the above figure, the stream tagged Bittorrent has matching signatures up to 34.25%.

Netwitness Investigator

These results show that Netwitness Investigator 's DPI engine is similar to OpenDPI's in that it was only able to detect plain TCP traffic, while the other 3 variations went undetected. However, it was able to pick up announce and scrape files which are used in torrent information transactions.

Further analysis of information extracted by Netwitness Investigator

As Netwitness Investigator was able to extract both the announce and scrape files, which are used to initiate the association of the user's machine with the peer cloud, further analsyis was warranted in order to dsetermine whether there was any forensically viable evidence. The contents of the extracted file are shown in Figure 2 as follows:

```
GET
/announce?info_hash=%ad%1dq%a3%0b%40%9e%28%ca%3f%b9%bb%93%2f%3e%a7%18q%b3%f
a&
peer_id=-UT2210-
%d6bBQ%d0%de1%f0%f7v%17%e6&port=34044&uploaded=0&downloaded=14343
2120&left=0&corrupt=0&key=88AE4C54&event=completed&numwant=200&compact=1&no_peer
_
id=1&ipv6=fe80%3a%3a20c%3a29ff%3afe15%3a25db HTTP/1.1
Host: core-tracker.enlist-a-distro.net:9800
User-Agent: uTorrent/2210(25302)
Accept-Encoding: gzip
Connection: Close
```

Figure 2: Information extracted from a BitTorrent packet capture by Netwitness Investigator

It was possible to reverse engineer a hash figure of the content file from the above announce file.

The info_hash is SHA-1 hash of the info dictionary in bencoded format.

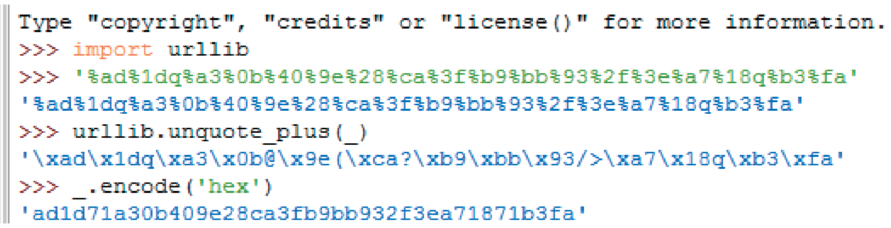
You have to get the *info dictionary* part of the file in order to compute its hash. Assuming you have access to a bencode encoder and decoder you could:

- decode the whole file
- take the *info dictionary* part of it
- re-encode it for hashing. (Jasmin, 2010)

Therefore, info_hash extracted from the announce file would be the following:

info_hash=%ad%1dq%a3%0b%40%9e%28%ca%3f%b9%bb%93%2f%3e%a7%18q%b3%fa

Using Python 's urllib to reverse the URL encoded binary hash into a HEX hash, yields a hexadecimal representation of the torrent data (Figure 3), which can then be further converted into plain text (Figure 4).



```
Type "copyright", "credits" or "license()" for more information.
>>> import urllib
>>> '%ad%1dq%a3%0b%40%9e%28%ca%3f%b9%bb%93%2f%3e%a7%18q%b3%fa'
'%ad%1dq%a3%0b%40%9e%28%ca%3f%b9%bb%93%2f%3e%a7%18q%b3%fa'
>>> urllib.unquote_plus(_)
'\xad\x1dq\xa3\x0b@\x9e(\xca?\xb9\xbb\x93/>\xa7\x18q\xb3\xfa'
>>> _.encode('hex')
'ad1d71a30b409e28ca3fb9bb932f3ea71871b3fa'
```

Figure 3: Hexadecimal information extracted from a BitTorrent packet capture by Netwitness Investigator

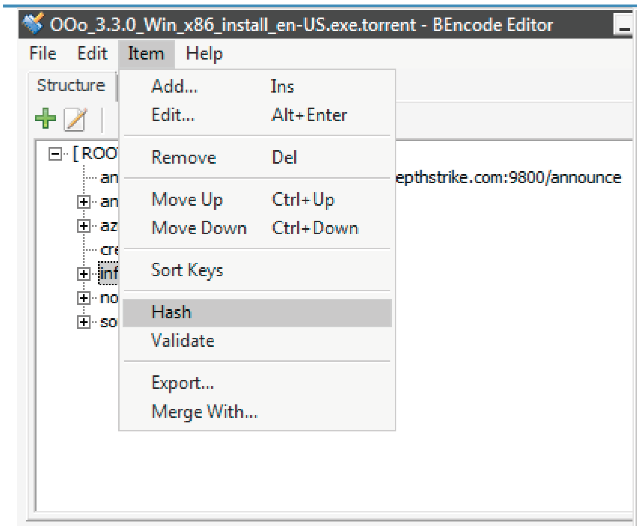
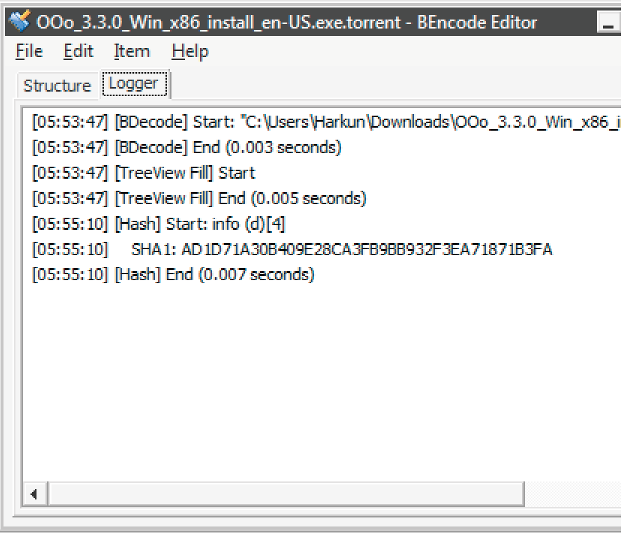


Figure 26: BEncode Editor - Hashing


OOo_3.3.0_Win_x86_install_en-US.exe.torrent - BEncode Editor
File Edit Item Help
Structure Logger

```
[05:53:47] [BDecode] Start: "C:\Users\Harkun\Downloads\OOo_3.3.0_Win_x86_i
[05:53:47] [BDecode] End (0.003 seconds)
[05:53:47] [TreeView Fill] Start
[05:53:47] [TreeView Fill] End (0.005 seconds)
[05:55:10] [Hash] Start: info (d)[4]
[05:55:10]    SHA1: AD1D71A30B409E28CA3FB9BB932F3EA71871B3FA
[05:55:10] [Hash] End (0.007 seconds)
```

Figure 27: BEncode Editor - hash value

Figure 4: Decoded bencode data extracted from a packet capture of a BitTorrent file download showing the name of the file and its associated hash value

DISCUSSION

Having analyzed the results from the selected tools, it is clear that all of them use a signature based matching algorithm. Even DPI uses content matching at the start of streams to determine traffic type. Although µTP has been around for over a year, no modification of current DPI engines appears to have taken place, and thus this protocol is not being detection by these tools. Unless an update is applied to the engine, the monitoring system will be of no value in relation to detection of BitTorrent traffic. Another inherent hurdle is the nature of BitTorrent content request in itself. A multi-piece parallel download approach as described in the RFC shuffles the sequence of raw data. Downloading from many peers simultaneously hurts the recomposing process even more.

As for file carving from network captures, according to Damaye (Personal communication, June 2011), no tool exists that could recompose torrent content as at the time of writing. He commented that it may be possible to do so, however, thorough knowledge of the RFC is necessary to write such a tool. Still, a problem lies in the existence of encryption; if one does not have the keypair, it will be difficult to recompose its contents.

Even though it is possible to decipher a SHA1 of a torrent's contents it is still impossible to identify it solely from the announce file alone: a match must still be made with the original torrent file. This may seem to be a hurdle to cross when only network traffic is available. Unless the host machine is in custody and torrent files been found on it, it will be difficult to determine what he/she has downloaded. Another option is to correctly identify and carve out the .torrent file itself from the traffic capture preceding the actual torrent content download (if any), then using it as the associating end to match data against.

Moving on to the future, as Internet Service Providers catch up with BTP's evolutions and begin throttling traffic, more evasion techniques will be born 'out of necessity' from the peer to peer file sharing community, much like uTP is today. Investigators not only need to first, identify BitTorrent traffic, he also has to associate them with other evidences found elsewhere. An intelligent investigator need not be on the cutting edge of technology to fight fire with fire, but to wisely connect the dots with given information.

Although this study focused on identification of traffic from an evidentiary perspective, there is another issue raised by this research. Given that some of the files transferred using BitTorrent are illegal in nature, and certainly highly undesirable, the Australian Federal Government has expressed its intention to filter all internet traffic into and out of Australia. Whilst the currently proposed mechanism for such filtering would be URL based, there is also anecdotal evidence that DPI based technology would be employed in order to classify and potentially block unwanted traffic. Whilst the exact mechanisms used by such devices are proprietary, it seems that use of such technology is unlikely to achieve the Governments aims, given that such software based implementations as tested in this research were unable to detect BitTorrent traffic when encryption or the µTP protocol was used.

CONCLUSION

It is conclusive that, no matter which tool was used, a complete reconstructed file was not able to be dumped. Detection of clear TCP torrent traffic was difficult in itself; stateful, deep packet inspection techniques employed by Network Miner and Netwitness Investigator were also unable to fully detect even clear TCP traffic. Only OpenDPI was very much attuned to the detection of clear TCP traffic, other than that, it suffered the same detection flaws as any other tool. Apart from DPI, SPID algorithm was used in hopes of detecting traffic. This algorithm does not scan deep into the packet to uncover contents, however, it analyzes flows and payload statistics. Still, the algorithm is not perfect.

Future research will endeavour to examine hardware based DPI solutions to determine whether such technologies are capable of detecting all formats investigated in this research. Namely, TCP and µTP, both encrypted and unencrypted.

REFERENCES

EMC Corporation. (2011). Investigator. Retrieved from http://www.netwitness.com/products-services/investigator

Fonseca, J., Reza, B., Fjeldsted, L., & DIKU. (2005). BitTorrent Protocol -- BTP/1.0. Retrieved from BitTorrent Protocol -- BTP/1.0 website: http://jonas.nitro.dk/bittorrent/bittorrent-rfc.html#anchor1

Hjelmvik, E. (2009a). Nothing but Network Forensics. Retrieved from http://www.dfrws.org/2009/challenge/hjelmvik.pdf

Hjelmvik, E. (2009b). SPID Algorithm Proof-of-Concept. Retrieved from http://sourceforge.net/projects/spid/

Hughes, J., & Lang, K. R. (2005). Peer-to-Peer Filesharing Systems for Digital Media. In Pagani, M. (Ed.), Encyclopedia of Multimedia Technology and Networking. (pp. 807-813). ipoque GmbH. (2011). Supported protocols and application. Retrieved from http://www.ipoque.com/sites/default/files/mediafiles/documents/data-sheet-protocol-support.pdf

Jasmin, A. (2010). How To Calculate Torrent Info Hash In VB6. Retrieved from http://stackoverflow.com/questions/4044733/how-to-calculate-torrent-info-hash-in-vb6

Kokkinen, H., & Nöyränen, J. (2009). Forensics for Detecting P2P Network Originated MP3 Files on the User Device Forensics in Telecommunications, Information and Multimedia. In M. Sorell (Ed.), (Vol. 8, pp. 10-18): Springer Berlin Heidelberg.

Mennecke, T. (2006). BitTorrent End to End Encryption and Bandwidth Throttling - Part I. Retrieved from http://www.slyck.com/news.php?story=1083

Norberg, A. (2009). uTorrent transport protocol. Retrieved from http://bittorrent.org/beps/bep_0029.html

Woodward, A. (2008). What Artefacts do Current BitTorrent Clients Leave Behind? In Proceedings of the 2008 International Conference on Security & Management (SAM '08)

Woodward, A. (2009). Do Current BitTorrent Clients Running on Windows 7 Beta Leave Behind Meaningful Data? In Proceedings of the 2009 International Conference on Security and Management (SAM'09)