Computer Science and Information Systems 14(1):197–218 DOI: 10.2298/CSIS160212032V

Methodology to Investigate BitTorrent Sync Protocol

Algimantas Venčkauskas, Vacius Jusas, Kęstutis Paulikas, and Jevgenijus Toldinas

Computer Department, Kaunas University of Technology,
Studentu st. 50, LT-51368, Kaunas, Lithuania,
{algimantas.venckauskas,vacius.jusas, kestutis.paulikas, eugenijus.toldinas}@ktu.lt

Abstract. The BitTorrent Sync client application is the most progressive development in the BitTorrent family. Nevertheless, it can be used for the activities that draw the attention of the forensics invetigators. The BitTorrent Sync client application employs quite largely the encryption for sending data packages. The initiation of the activity is carried out in the plain text only. Therefore, we proposed the methodology that enables to capture the initiation step and to inform the forensics investigator, which then takes the reactive actions. The experiment was carried in two modes: 1) simulating of the use of the BitTorrent Sync application; 2) monitoring of real traffic on the Internet. During the monitoring, it is possible to calculate the public lookup SHA1 hash of the shared file. The comparison of the calculated hash with the list of publicly available hashes allows determination whether sharing of the file is legal or illegal. The presented methodology can be applied to any BitTorrent protocol.

Keywords: BitTorrent protocol, forensics investigation, computer network, cybercrime.

1. Introduction

BitTorrent protocol was designed for file sharing over the Internet. It was not the first one to be used for such a purpose, however, it was the most elaborate and became de facto standard for file sharing. The BitTorrent protocol was designed with good intent but it is quite largely used for sharing of copyrighted material [1]. Illegal copies of copyrighted content can be found in more than two thirds of torrents registered at one of the most popular BitTorrent trackers [2]. Consequently, such use of the protocol creates a revenue that downgrades the copyright holder's share. Therefore, the active use of BitTorrent protocol creates new challenges for forensics investigators that have to monitor the use of the protocol and collect the evidence whether the use is legal. Moreover, the users involved in the use of the BitTorrent protocol have to allow exploiting their resources for file sharing since BitTorrent applications may punish non-uploaders by limiting their download bandwidth [3].

An universe of BitTorrent protocol can be regarded as the structure consisting of several levels of hierarchy. At the highest level, the universe of BitTorrent network can be represented as being divided into many BitTorrent swarms. Each shared content forms a BitTorrent swarm that is composed of trackers and peers [4]. A peer is an agent

that runs an implementation of the protocol. The same peer can participate in multiple swarms, if he wishes to share several files.

In order to initiate a download of the content the user must first download a metadata .torrent file from some website. Then, the BitTorrent client application of the user interprets the metadata and uses it to detect other peers participating in that swarm using one of the following methods: tracker, distributed hash table (DHT), peer exchange (PEX). A tracker is a server that maintain the content-to-peers-IP-address mapping for all the contents they are tracking. Once a peer has downloaded the .torrent file from a website, it contacts the tracker to subscribe for that content and the tracker returns a randomly chosen subset of peers that have previously subscribed for that content. During content transfer, the client application periodically reports to the tracker in order to update its status and to keep up to date the list of active peers. PEX allows a direct interchange of peer lists with other peers.

DHT is a distributed tracker that allows peers to locate the other peers requesting information from BitTorrent clients without the requirement for a central server. DHT implemented in BitTorrent client is called Mainline. The results of recent measurement show that Mainline DHT is the largest P2P network having from 15 million to 27 million users concurrently online, with a daily churn of at least 10 million users [5].

The BitTorrent protocol reduces the impact of distributing large files on both the server and the network. The main strength of the protocol is the division of the file into separate equal size parts and separate management of these parts. Instead of downloading a file from a single source, BitTorrent enables users to join a swarm of peers. In the swarm, peers simultaneously download and upload from each other. In addition, protocol is adaptive to low band networks since it divides the file into smaller pieces in this case.

The file to be distributed is divided into pieces. When peer receives a new piece of the file, it becomes a source for other peers. Therefore, those peers, who participate in the swarm, have an obligation to allow using their resources for content distribution. Once the user is connected with the swarm, he can download available pieces from several peers simultaneously. This mechanism improves the download speed.

The pieces of a file are downloaded randomly and the BitTorrent client rearranges them into the correct order. The client also monitors which pieces it has, which it can upload to other peers, which it needs and who has them. In separate swarm, all the pieces of the file are of the same size (for example, a 100 MB file can be downloaded as ten 10 MB pieces or as five 20 MB pieces).

Family of products using BitTorrent like protocol constantly evolves and increases. The latest and the most progressive product in this family is BitTorrent Sync. It is a file replication utility released in April 2013 [6]. This utility is very desirable to those who are involved in illegal activities, because it ensures to keep data transfer secure from inspection while in transit [7]. The initiation of the activity of the BitTorrent Sync utility is carried out in the plain text only. Therefore, the utility can be exploited for several potential crimes as follows: to share copyrighted material, to share child pornography, to distribute malicious software, for industrial espionage, etc. We present the tool that helps the forensics investigators to discover the initiation of the BitTorrent Sync utility and then to collect the evidences.

The artifacts left by BitTorrent client can be separated into two parts: 1) the artifacts on the client computer, and 2) the artifacts on the computer network. In the next section, we review the related work that considers the artifacts on the computer network.

2. Related Work

Although the goal of BitTorrent Sync is not to replace BitTorrent as a file downloading utility, it will likely be used for such a purpose. This prediction is based on the fact that as soon as BitTorrent Sync was released, publicly available secrets started to appear online [6]. Such use of BitTorrent Sync presents many possibilities for potential crimes.

Three research works [6], [7], [8] are known to this moment, which are dedicated to the investigation of BitTorrent Sync tool. Farina et al. [6] investigate in detail the traces left by BitTorrent Sync in the client computer. The authors conducted an experiment during which they installed the BitTorrent Sync client on Windows XP machine. A complete list of the files created during the installation procedure is provided. Registry keys created during installation procedure are presented, as well. Default installation process creates BTSync folder that is filled in with three hidden files: .SyncID, .SyncIgnore, .SyncArhive (Folder). BitTorrent Sync utility can be configured to use the fourth method to locate peers additionally to known three methods: tracker, DHT, PEX. The user can add a list of IP address:Port to denote the peers that will be contacted directly without any lookup. This method is least detectable. Important part of communication is BitTorrent Sync keys. They are of four varieties: master key, readonly key, 24 hour read/write, 24 hour readonly key. In order to collect the data three folders were synchronized. Three separate synchronizations using network data were carried out, as well. The content of BitTorrent log file is provided. Finally, the BitTorrent Sync application was uninstalled. The remaining Windows registry keys are provided, as well.

Scanlon et al. [7] move further in the investigation and propose a methodology to outline the required steps to retrieve a digital evidence from the network. The authors perform the analysis of the network traffic, which is formed during regular operations of BitTorrent Sync, and present the contents of each request and response packets. The detailed description of LAN multi-cast, tracker, relay server, known host and file synchronization is outlined. While much of the network topology of BitTorrent Sync is the same as in case of regular BitTorrent, but the sent and received packets slightly differ. The differences are outlined. However, these differences already have no value since the protocol is changed beginning version 2.

The proposed investigation methodology consists of the following five steps: 1) obtain the secret or key; 2) calculate public lookup SHA1 hash; 3) crawl the network to identify the peer information; 4) download the content and verify; 5) place the information gathered into suitable evidence bag. The fact of implementation of crawling application and few experimental results are provided only. The implementation is devoted for emulation rather than for investigation.

Scanlon et al. [8] present a methodology for a remote recovery of digital evidence related to files recognized as being accessed or stored on a suspect's computer or smart device. The methodology consists of five following steps: 1) discover entry points; 2) investigate local metadata; 3) identify remote data stores; 4) download a remote

evidence; 5) verify the evidence. The methodology presented is not related to the BitTorrent Sync, but the experiment was carried out using BitTorrent Sync. One weakness of the approach is that the remote machine can detect an unknown IP connecting to it and requesting to synchronize the evidence. Upon detection of an unknown IP address, the remote machine might disconnect.

BitTorrent Sync is based on the BitTorrent protocol. We are going to present the tool that tracks BitTorrent Sync activities on the network. Therefore, the research works devoted for the investigation of the suspicious activities on the network using BitTorrent protocol are of interest to the investigator of BitTorrent Sync. Chow et al. [9] proposed BitTorrent Monitoring System (BTM), one of the first systems for tracking illegal downloaders. BTM system searches websites in order to find the torrent files, attempts to locate the torrent files that have metadata on illegal files. Then, BTM system extracts peer lists from torrent files and communicates with peers in order to collect an information. BTM system provides set of rules for torrents files and peers. Rules define the attributes for torrent files and peers. Rules provide flexibility for different users of BTM system. However, BTM system has several drawbacks. Firstly, BTM system is not designed to run in real time, therefore, it cannot keep with constantly varying peer lists. Secondly, BTM system is limited to predefined level of hyperlinks in searching on websites.

Bauer et al. [10] developed an active probing framework called BitStalker that identifies active peers and collects forensic evidence that they were involved in sharing a particular file. Firstly, to obtain the list of peers who are potentially sharing the file, the tracker is queried. For each IP address and port number obtained, a series of probes are conducted to determine whether the peer really exists and is participating in the file transfer. The probes consist of the following messages: TCP connection, handshake, bitfield, block request of 16KB. It is possible that peers will be able to detect the monitors and blacklist them. To solve this issue, Bauer et al. recommended distributing of the monitoring across a large number or dynamic set of IP addresses. The overhead of active probing is dependent on the fraction of peers that respond to active probes. According to conducted experiments, 61 peers were probed per second. However, the developed framework BitStalker can be employed only if the peer exchange method is a tracker.

Schrader et al. [11] suggested a field programmable gate array (FPGA)-based embedded software TRAPP system designed to process file transfers using the BitTorrent protocol and VoIP phone calls. The TRAPP system is designed to be set up on the gateway between a local area network and the Internet. As packets pass through the gateway, copies are sent to the system for analysis. TRAPP system extracts the first 32 bits of packet and compares them to the first 32 bits of a BitTorrent Handshake message. If the case of match, the first 32 bits of the Handshake's file info hash are extracted from the packet, and compared against a list of hashes belonging to files of interest using a binary search. Then, in the case of match, the packet is recorded in Wireshark format log file for future analysis. The goal of the experiment was to find the most effective FPGA configuration that it would impact the load of the network in the least possible way. The drawback of method is that it can locate the packets of the known illegal files only. There will be lost the time till the hashes of the new illegal files will be included into the database.

Mansilha et al. [4] presented TorrentU architecture to observe BitTorrent networks. TorrentU architecture consists of two components: TorrentU Observer and TorrentU Telescope. The Observer is a manager of Telescopes. The Observer directs the telescopes and stores the information obtained from the Telescopes. The telescopes are divided into subcomponents. The most important component is a Monitor, which is further divided into three parts: Community lens, Tracker lens, and Peer lens. The names indicate for which part of BitTorrent network they are responsible. A prototype of the architecture was implemented in order to evaluate the feasibility of the approach. TorrentU architecture resembles to the BTM system [9], but it is more elaborate than BTM system. They both represent the global view of the BitTorrent network when the observation starts on the torrent sites rather than the local computers.

Lee and Kim [12] introduces an anti-BitTorrent filter (ABF) system that consists of hash database, network analysis part, content feature extraction module, torrent meta-file crawler, and copyright identifier. The goal of the system is to guard the content from BitTorrent distribution. This goal is quite different from all the research works presented so far. Therefore, the key feature of ABF system is a hash DB. The ABF system targets to have hashes of all the illegally distributable files. The network analyzer employs a PCAP (Packet Capture) library to extract BitTorrent traffic from the network. The system is introduced very shortly; no results of the experiment are provided.

Park et al. [1] suggest a methodology for the investigation of illegal file sharers using BitTorrent networks. The methodology is oriented to the popular BitTorrent client program, uTorrent. The methodology consists of the following steps: 1) acquire torrent file; 2) analyze packets; 3) seize suspect's PC, 4) analyze suspect's PC. Only general information is provided about packet analysis. Obvious information is presented for suspect's PC analysis. The prototype of the proposed methodology was implemented. Some results of the experiment are delivered. We can conclude that the presented methodology is very general and it is presented at the high level of the abstraction.

In the next section, we introduce the new methodology to collect the evidence of using BitTorrent Sync utility. The methodology is based on the monitor that is connected into the computer network near the suspect as a passive observer. The monitor catches all the traffic going to the suspect computer. Later the captured information is analyzed and the peers taking part in the information sharing using BitTorrent Sync are identified. Finally, the tool presents the following information: the IP addresses of the peers, which participated in the sharing file, and shareID of the shared file. Then the forensics investigator on the base of the presented information can decide to prosecute or not the identified peers. The advantages of the proposed methodology are as follows: 1) the tools used are oriented to the latest version of the BitTorrent Sync; 2) it presents the peer information without crawling the network; 3) it can be adjusted to monitor other protocol of the BitTorrent family.

3. Methodology to Monitor the Activity of BitTorrent Sync on the Computer Network

Nowadays implementations of BitTorrent protocol (traditional BitTorrent, BTSync, Bleep) employ data encryption that makes the work of law enforcement agencies more

cumbersome in the investigation of the suspicious activities. Decryption algorithms are usually not accessible publicly. Since decryption of the data traffic is not possible using available resources we propose a methodology that is oriented to the plain text information and infrastructure.

BitTorrent protocols use trackers to locate peers. Not all the trackers acting in the global network are known, moreover, the trackers is not the only method to locate the members of the swarm. Nevertheless, the communication of the client computer with the tracker indicates the possible existence of the active BitTorrent program in the client computer. However, such only existence of the active BitTorrent program is not a discriminative feature since the BitTorrent protocol is used for sharing legal files as well. The more evidence is needed to prove the illegal activity. We propose the architecture of the system (Fig. 1) that would allow collecting such an evidence.

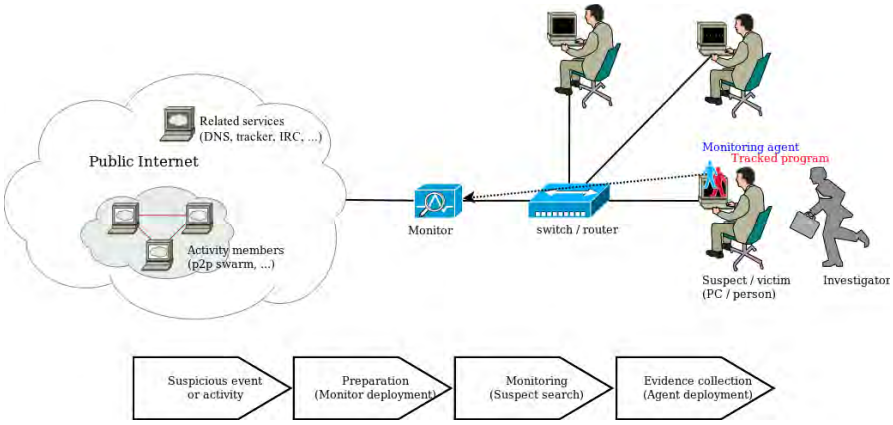

Fig. 1. The general architecture of the system.

The system is composed of the following components:

1. Internet – public network, which hosts the members of the swarm sharing the copyrighted material and holds the required infrastructure to maintain such an activity (DNS, tracker sites, storage sites of torrent files);
2. Suspect/victim – machine that participates in suspicious activity and which is possibly criminal or victim;
3. Switch/router – layer-2 device that connects the victim machine. This device is emphasized since the suspect's machine has direct cable connection to this device and the disconnection of the cable, which is required to plug in the monitor, will alert the suspect's machine.
4. Monitor – layer-2 device that is connected into the network line and is able to capture all the traffic passing along this line. Additionally, the monitor has an installed software that enables dynamically to change the rules for storing and analysis of passing traffic;
5. Investigator – officer of the law enforcement agency, which physically seizes the suspect's machine and installs the tracking agent;

6. Tracking agent – the software making the minimal changes to the system, acting on behalf of the ordinary user, devoted for monitoring of the system and sending the information to the monitor.

The use case is as follows:

1. When the illegal activity is suspected, the monitor is plugged in into the network close as possible to the suspect's machine but not closer as to one device of the layer-2;
2. The traffic to suspect's machine is observed using the monitor;
3. When the enough amount of the data showing the illegal activity of the suspect is collected, the investigator installs directly into the suspect's machine the tracking agent without the overloading computer in order would not lose the volatile artefacts of the illegal activity.
4. Tracking agent keeps track of the flowing information and sends it to the monitor.

The detailed tracking scheme is presented in Fig. 2. The main task of tracking agent is to keep track of the ports used by the tracked program (BitTorrent/BTSync or similar program) and send this information to the monitor. The information on used ports and programs is obtained using the interfaces of operating system (OS). It needs to take into account that the program firstly starts using some ports and only then this information becomes available to the tracking agent and finally – to the monitor. During that period of time, program can send some portion of data, therefore, the monitor has to save the corresponding actual information till the updates come from the tracking agent.

Principal scheme of the monitor is presented in Fig. 3. The monitor does not change the traffic between the suspect's machine and Internet. The primary filter excludes only the traffic to the suspect's machine from the common flow of the traffic (the amount of the data is decreased, if several devices are plugged in into the network on the suspect's side). The captured traffic is sent to a buffer having a purpose to store the captured packets till the updates will come from the installed tracking agent on the suspect's machine. The management service of the filter analyzes the information coming from the tracking agent and regulates the adaptive filter according to the coming data. The packets captured in the adaptive filter and the information obtained from the tracking agent are placed into an evidence bag.

The sequence of the captured events during time flow and the actions of components of the system are shown in the diagram of component interaction (Fig. 4). The monitor can register the DNS queries to known tracker servers and links to these addresses. BTSync client (suspect) after obtaining IP address of the tracker sends request to it in order to obtain the addresses of the members of the swarm. Nevertheless that the contents of this message is encrypted, the fact of request is seen and can be registered. Then BTSync client starts the communication with peers participating in the swarm. All the packets of the suspect are registered in the buffer. Agent periodically collects an information of the used sockets in the suspect's machine and programs using them and sends this information to the monitor. The monitor using the information obtained from the agent filters the captured packets and stores the packets related to BTSync in the evidence bag.

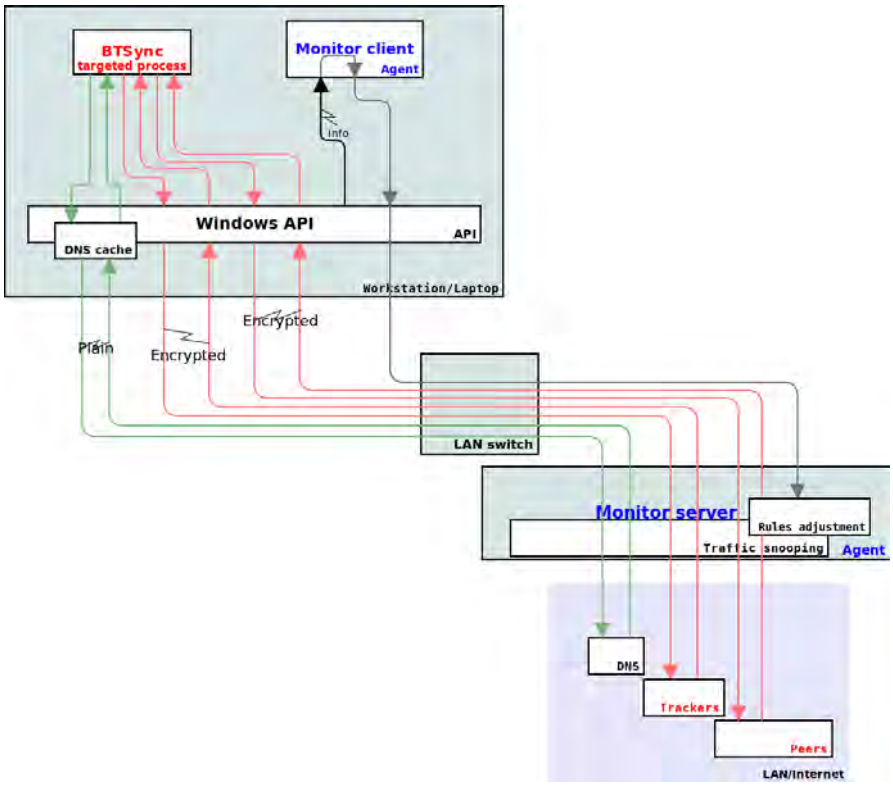

Fig. 2. The detailed tracking scheme

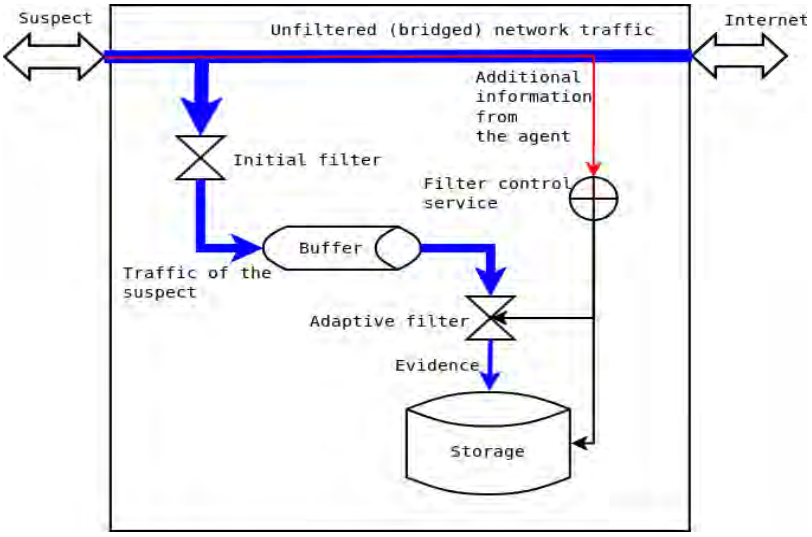

Fig. 3. The principal scheme of the monitor

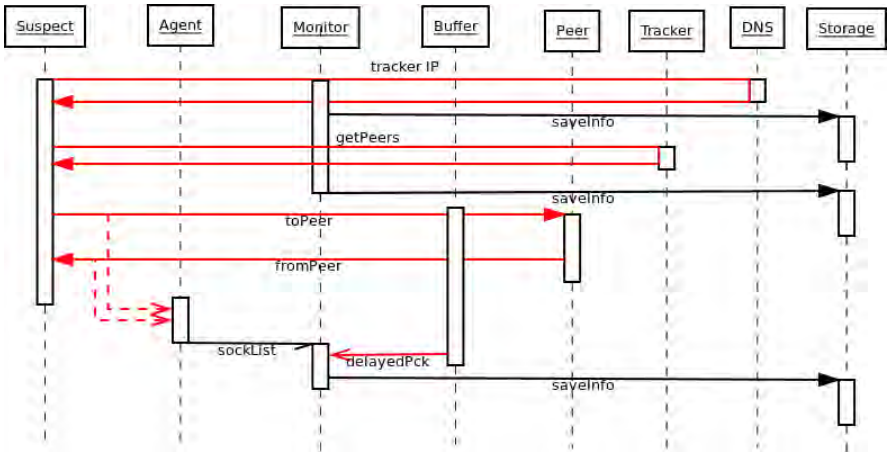

Fig. 4. The diagram of system component interaction

4. Experiment in the Simulated Environment

The proposed software tool (a monitor with an agent) was tested for an ability to capture BTSync related traffic. The setup of the experiment was as follows:

- standard PC (laptop, 4GB RAM, Intel i5 4 core CPU, 500GB HDD, LAN, WiFi);
- Linux Mint 16;
- VirtualBox v4.3 virtual machine (VM);
- BTSync client v2.2.2.112;
- Berkeley Packet (BP) filters (via tcpdump);
- ngrep (BP filter + grep);
- Wireshark.

Network management was performed using Linux OS on laptop (routing, NAT, firewall). Host-only networking was used on Virtual box and it was routed to a public Internet. Windows 7 OS was used on virtual machine (guest). BTSync client and agent were installed on this Windows virtual machine. Monitor was installed on Linux (host) machine. A few shareIDs from public database at http://btsynckeys.com/ were downloaded.

We have chosen the configuration that is as close as possible to the real conditions in the computing world and the best tools are available for this configuration. The network management is usually done using Linux OS computer. The Wireshark is an open source tool for the network packet analysis. The Berkeley Packet filters are adjusted to capture specific type of the packets. Windows 7 OS was used for the client computer, since many users use on their computers Windows OS. Finally, the latest version of

BitTorrent Sync was installed on the client computer. This version was not yet explored in the research works.

Steps of the experiment were as follows:

1. Monitor started on the host.
2. Guest virtual machine started.
3. Agent started on the guest.
4. BTSync client started on the guest.
5. Several shareIDs were added to the BTSync client.
6. System was left running for a few days.
7. BTSync client stopped.
8. Agent stopped.
9. Guest virtual machine was closed.
10. Monitor stopped.

While the tool lacked ability to select UDP traffic, TCP streams were captured and saved for later analysis. DNS queries coming from guest VM were also captured.

The general results can be presented as follows:

- DNS requests to config.getsync.com and downloads of http://config.getsync.com/sync.conf every 2 hours, while BTSync client is running. The DNS name resolves to config.getsync.com.s3-website-us-east-1.amazonaws.com, which resolves to IP address and changes over time.
- DNS requests to i-2000.b-2-2-2.sync.bench.getsync.com, but the purpose was not identified (no relevant TCP traffic was captured). The name resolves to CNAME bench.utp.st, which itself resolves to com-utorrent-prod-bench-290894750.us-east-1.elb.amazonaws.com and then to IP address pool.
- DNS queries to help.getsync.com resolve to getsync-help.bittorent.desk.com (both IPv4 and IPv6 queries). BTSync client related TCP traffic was not captured.
- DNS queries to getsync.com (both IPv4 and IPv6 queries) did not have BTSync client related TCP traffic.
- TCP connections to trackers and relays received from config.getsync.com were identified; both were mainly plaintext.
- TCP traffic used to transfer data content between peers in the swarm was captured.

BTSync client was controlled from local WWW browser, which probably is the source of DNS queries to help.getsync.com and getsync.com, where no BTSync client related TCP traffic was captured.

Public read-only secret BP6P6PMB7HVG5YGSTIP24SXQY6DC36R5C from the http://btsynckeys.com site was taken for a detailed analysis on 2015-09-30 (the key was published on the site on 2015-09-29 04:04:58). Share hash for this secret is as follows [13], [14]:

A9DBB686EF24BE29D131127667615987B9E616AD

This value can be obtained on Linux using the following command:

```
$ echo -n P6P6PMB7HVG5YGSTIP24SXQY6DC36R5C | ./base32dec.pl | sha1sum
```

The source of base32dec.pl script [15]:

```
#!/usr/bin/perl
use MIME::Base32 qw( RFC );
$string = <STDIN>;
$encoded = MIME::Base32::decode($string);
print "$encoded";
```

BTSync infrastructure configuration can be found using the following command (BTSync client periodically fetches this data, while running):

```
$ curl http://config.getsync.com/sync.conf
{
 "trackers" :
 [
  {"addr" : "52.0.104.40:3000"},
  {"addr" : "52.0.102.230:3000"},
  {"addr" : "52.1.40.103:3000"},
  {"addr" : "52.1.1.135:3000"}
 ],
 "relays" :
 [
  {"addr" : "67.215.231.242:3000"},
  {"addr" : "67.215.229.106:3000"},
  {"addr" : "66.63.177.26:3000"}
 ],
 "mobile_push_proxies" :
 [
  {"addr" : "54.235.182.157:3000"}
 ]
}
```

Dump of HTTP query performed by a BTSync client itself is as follows:

```
00000000 47 45 54 20 2f 73 79 6e 63 2e 63 6f 6e 66 20 48 GET /syn c.conf H
00000010 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 63 TTP/1.1. .Host: c
00000020 6f 6e 66 69 67 2e 67 65 74 73 79 6e 63 2e 63 6f onfig.ge tsync.co
00000030 6d 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 42 m..User- Agent: B
00000040 54 57 65 62 43 6c 69 65 6e 74 2f 32 32 32 30 28 TWebClie nt/2220(
00000050 33 33 36 38 35 35 30 36 29 0d 0a 41 63 63 65 70 33685506 )..Accep
00000060 74 2d 45 6e 63 6f 64 69 6e 67 3a 20 67 7a 69 70 t-Encodi ng: gzip
00000070 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 43 6c ..Connec tion: Cl
00000080 6f 73 65 0d 0a 0d 0a 0d 0a                      ose..... .
```

The simple Wireshark filter allows to select tracker traffic only:

tcp.port == 3000 and (ip.addr == 52.0.104.40 or ip.addr == 52.0.102.230 or ip.addr == 52.1.40.103 or ip.addr == 52.1.1.135)

The same task can be achieved using BP filter notation:

tcp and port 3000 and (host 52.0.104.40 or 52.0.102.230 or 52.1.40.103 or 52.1.1.135)

The BTSync client contacts a tracker to register itself and to get peers related to a share:

```
00000000 00 00 00 62 64 32 3a 6c 61 36 3a c0 a8 38 67 36 ...bd2:l a6:..8g6
00000010 46 32 3a 6c 70 69 31 33 38 39 34 65 31 3a 6d 39 F2:lpi13 894e1:m9
00000020 3a 67 65 74 5f 70 65 65 72 73 34 3a 70 65 65 72 :get_pee rs4:peer
00000030 32 30 3a 10 cb 15 eb 71 93 96 26 07 13 6d 4c d3 20:....q ..&..mL.
00000040 9a d8 47 d9 39 3b d2 35 3a 73 68 61 72 65 32 30 ..G.9;.5 :share20
00000050 3a a9 db b6 86 ef 24 be 29 d1 31 12 76 67 61 59 :.....$. ).1.vgaY
00000060 87 b9 e6 16 ad 65                               .....e
```

The share hash is clearly visible in a request. It is shown in bold.

Decoded structure of the request is as follows (bencoded format [16]), starting from 4-th byte, these 4 bytes contain total length of the bencoded data):

```
{
  la (str) = 192.168.56.103:13894 (hex: C0 A8 38 67 3646)
  lp (int) = 13894
  m (str) = get_peers
  peer (str) = 10cb15eb7193962607136d4cd39ad847d9393bd2
  share (str) = a9dbb686ef24be29d131127667615987b9e616ad
}
```

Tracker response is shown below:

```
00000000 00 00 00 b7 64 32 3a 65 61 36 3a 9e 81 07 51 c1 ....d2:e a6:...Q.
00000010 13 31 3a 6d 35 3a 70 65 65 72 73 35 3a 70 65 65 .1:m5:pe ers5:pee
00000020 72 73 6c 64 31 3a 61 36 3a 73 96 72 09 32 f7 32 rsld1:a6 :s.r.2.2
00000030 3a 6c 61 36 3a c0 a8 01 66 32 f7 31 3a 70 32 30 :la6:... f2.1:p20
00000040 3a 10 55 2a be ed b4 3c 4b 36 7d bb ff e7 5e 7c :.U*...< K6}...^|
00000050 54 9d 12 89 30 65 64 31 3a 61 36 3a 9e 81 07 51 T...0ed1 :a6:...Q
00000060 36 46 32 3a 6c 61 36 3a c0 a8 38 67 36 46 31 3a 6F2:la6: ..8g6F1:
00000070 70 32 30 3a 10 cb 15 eb 71 93 96 26 07 13 6d 4c p20:.... q..&..mL
00000080 d3 9a d8 47 d9 39 3b d2 65 65 35 3a 73 68 61 72 ...G.9;. ee5:shar
00000090 65 32 30 3a a9 db b6 86 ef 24 be 29 d1 31 12 76 e20:.... .$.).1.v
000000A0 67 61 59 87 b9 e6 16 ad 34 3a 74 69 6d 65 69 31 gaY..... 4:timei1
000000B0 34 34 33 36 31 31 39 38 32 65 65                44361198 2ee
000000BB 00 00 00 84 64 32 3a 65 61 36 3a 9e 81 07 51 36 ....d2:e a6:...Q6
000000CB 46 31 3a 6d 35 3a 70 65 65 72 73 35 3a 70 65 65 F1:m5:pe ers5:pee
000000DB 72 73 6c 64 31 3a 61 36 3a 9e 81 07 51 36 46 32 rsld1:a6 :...Q6F2
000000EB 3a 6c 61 36 3a c0 a8 38 67 36 46 31 3a 70 32 30 :la6:..8 g6F1:p20
000000FB 3a 10 cb 15 eb 71 93 96 26 07 13 6d 4c d3 9a d8 :....q.. &..mL...
0000010B 47 d9 39 3b d2 65 65 35 3a 73 68 61 72 65 32 30 G.9;.ee5 :share20
0000011B 3a a9 db b6 86 ef 24 be 29 d1 31 12 76 67 61 59 :.....$. ).1.vgaY
0000012B 87 b9 e6 16 ad 34 3a 74 69 6d 65 69 31 34 34 33 .....4:t imei1443
0000013B 36 31 31 39 38 32 65 65                         611982ee
```

The same share hash is shown in bold. The decoded structure of the tracker response is as follows:

```
{
 ea (str) = 158.129.7.81:49427 (hex: 9E 81 07 51 C113
 m (str) = peers
 peers (list) [
  {
   a (str) = 115.150.114.9:13047 (hex: 73 96 72 09 32F7)
   la (str) = 192.168.1.102:13047 (hex: C0 A8 01 66 32F7)
   p (str) = 10552abeedb43c4b367dbbffe75e7c549d128930
  }
  {
   a (str) = 158.129.7.81:13894 (hex: 9E 81 07 51 3646)
   la (str) = 192.168.56.103:13894 (hex: C0 A8 38 67 3646)
   p (str) = 10cb15eb7193962607136d4cd39ad847d9393bd2
  }
 ]
 share (str) = a9dbb686ef24be29d131127667615987b9e616ad
 time (int) = 2015-09-30 14:19:42 GMT+3 (UNIX 1443611982)
}
```

Field names and their meaning:

- la – “local address” and port of the client (6 bytes);
- lp – “local port”;
- a – “address” and port of a peer (6 bytes);
- m – command name;
- peer, p – peer identity (20 bytes);
- share – share hash (20 bytes SHA1 hash of share secret as described above);
- peers – list of peers involved in distributing a particular share;
- time – timestamp of response.

IP addresses seen in the conversation with a tracker correctly represent network infrastructure used in the experiment:

- 192.168.56.103 is an address of virtual machine, where BTSync client was run;
- 158.129.7.81 is NATed IP address of this machine used to access public Internet;
- 115.150.114.9 is an IP address of another peer involved in distributing a share (data transfer directly from 15.150.114.9:13047 was captured too).

It was observed that one share was added to the BTSync client in the tested virtual machine on the 52.1.1.135:3000 tracker with a hash value 22DA23C75F5C7A6605320B1EEDDB715A98284833. The reason of it's appearance is

unknown. The content of the SQLite3 database file left on the hard disk of the virtual machine contained the following entries:

```
$ sqlite3 22DA23C75F5C7A6605320B1EEDDB715A98284833.1.db
SQLite version 3.7.9 2011-11-01 00:52:41
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> select path from files;
identity.dat
folders\-636494180347531526
folders\-636494180347531526\info.dat
notifications\14436115055220373755.dat
folders
devices
devices\CDFRL23RSOLCMBYTNVGNHGWYI7MTSO6S
devices\CDFRL23RSOLCMBYTNVGNHGWYI7MTSO6S\folders
devices\CDFRL23RSOLCMBYTNVGNHGWYI7MTSO6S\folders\-
636494180347531526
devices\CDFRL23RSOLCMBYTNVGNHGWYI7MTSO6S\info.dat
notifications
notifications\10110802179274914772.dat
```

The files listed are present on the virtual machine in the *C:\Users\Admin\AppData\Roaming\BitTorrentSync\.SyncUser1443611213* folder, where CDFRL23RSOLCMBYTNVGNHGWYI7MTSO6S is base32 encoded peerID 10cb15eb7193962607136d4cd39ad847d9393bd2, which is visible in the conversations with a tracker. It can be decoded using the following command:

```
$ echo -n CDFRL23RSOLCMBYTNVGNHGWYI7MTSO6S | ./base32dec.pl |
hexdump -C
00000000 10 cb 15 eb 71 93 96 26 07 13 6d 4c d3 9a d8 47 |....q..&..mL...G|
00000010 d9 39 3b d2 |.9;.|
```

These findings allowed making an assumption that some data of each peer is accessible to other related peers. In order to be sure the experiment was repeated on a fresh Windows7 virtual machine. BTSync Client v2.2.2.112 was installed, no shares were added to it and all traffic to/from this machine was captured. Once again a similar share was registered:

- suspected share key `MDRG7J6PYOFOIP5GZKILHM3Q5YLZEG5YR` was observed in *C:\Users\Admin\AppData\Roaming\Bittorent Sync\sync.dat* bencoded file.
- share hash: 8c1f7d00828b1dfecec6e3fa1b4cfea39ffb5aa0
- peerID: 107b8575eff196732620e7f604dda14243357196

5. Experiment on the Real Network

An experiment was carried out on the real Internet traffic. The traffic going in and out of student dormitories was selected at Kaunas University of Technology in Lithuania. The

performed experiment represents a monitoring preparation phase of the general proposed scheme (tracking agent was not used). To discover and monitor the BTSync traffic of the real world, monitor machine was attached to academic network. The monitor possessed the following properties:

- failover setup (two aggregated network links from a single department);
- 1Gbps each link (total 2Gbps full duplex);
- standard PC as a monitor (16GB RAM, 2x 1Gbps LAN interfaces, Intel i5 4 core CPU, 2TB SATA3 HDD, 250GB SSD SATA3 HDD);
- FreeBSD 10.2 OS (64bit);
- network link is loaded above 70% most of the time (load drops were slightly late at night)

The monitoring of the network traffic lasted approximately for one month. Hardware proved to be adequate, bottleneck being hard disk I/O only.

Several challenges had to be dealt with, which were not considered important before the experiment:

- network topology limits possibilities to choose location of the monitor for deployment. Ability to attach the monitor near the interesting endpoint is limited by a space and available access. Use of the monitor in bridge mode on high capacity links may be not feasible due to risks to introduce link failures. Traffic redirection to fixed network location is limited by available network bandwidth on the path from interception point to the monitor. Complex network topologies (failover mechanisms) require in-depth knowledge of monitored network to be able to intercept all related traffic;
- once the monitor is deployed, intercepted network traffic is not uniform and not always can be monitored properly using common tools. While multiple physical links can be aggregated into single one using OS mechanisms (lagg device), and virtual local area network (VLAN) can be separated into different VLAN devices, these devices simplify further traffic analysis, but do not cover all cases discovered. For example, one VLAN carried exclusively CAPWAP [17] traffic (RFC 5415) that further encapsulated not encrypted traffic;
- while hardware routers/switches have means to duplicate traffic (either mirroring it to some interface or sending a copy to some IP address), it becomes a problem on heavily loaded links, since two-way traffic becomes one-way, which overloads monitoring link capacity (for example, 1Gbps+1Gbps full duplex failover lines generate not up to 2Gbps, but up to 4Gbps traffic to be monitored). This issue was not addressed during the experiment. Therefore, only part of the traffic was intercepted.
- monitored network segment may have caching DNS servers, which are used by nodes in this network. These servers may have encrypted or side

channels to upstream DNS and at least some DNS queries are not visible in the captured traffic.

Consequently, the used experimental setup showed to be viable for the real overloaded Internet traffic and allowed reaching the goals of the experiment, except that it enabled to capture only half of the full duplex traffic. The full duplex traffic is not usual for the computer networks, and capturing it fully requires many efforts that are not always worthwhile.

The following tools and OS mechanisms were used to capture and analyze intercepted traffic:

- lagg driver;
- vlan driver;
- Berkeley Packet (BP) filters (via tcpdump);
- ngrep (BP filter + grep);
- Wireshark;
- grep in custom shell scripts.

The following tasks were attempted:

1. Identify BTSync users (by their IP)
2. Identify data shared (shareIDs available in public database at http://btsynckeys.com/)
3. Identify members of the swarm sharing a particular data (IPs related to sharing some shareID).

It was noticed that, when started, BTSync client “calls-back” to it's vendor via number of WWW links, DNS queries, tracker and proxy connections:

- http://config.getsync.com – the link replies with a JSON data, representing vendor infrastructure, used to support BTSync: addresses and ports of trackers and proxies. While these addresses remained stable for the duration of the experiment, config.getsync.com resolves to Amazon IPs and is changing all the time (resolved IP of config.getsync.com can not be used to accurately pinpoint BTsync client). DNS queries may be coming from dedicated DNS server within a network segment, so they don't identify swarm related node;
- https://link.getsync.com – used to exchange links to shares [18]. It resolves to getsync.com and remains stable;
- http://update.getsync.com – likely is used to distribute software updates, it is unstable (Amazon IPs);
- running BTSync client periodically connects to trackers received from config.getsync.com request (52.0.104.40:3000, 52.0.102.230:3000, 52.1.40.103:3000, 52.1.1.135:3000). This information is easily visible in unencrypted network traffic. Even if some encapsulation (CAPWAP) or IP

address translation is used, this traffic can be identified according to plaintext data contained in it;

- proxy and relay connections were not noticed during the experiment.

TCP conversations between BTSync clients can be identified by *BSYNC* at the start of the conversation. This information was used to identify BTSync users.

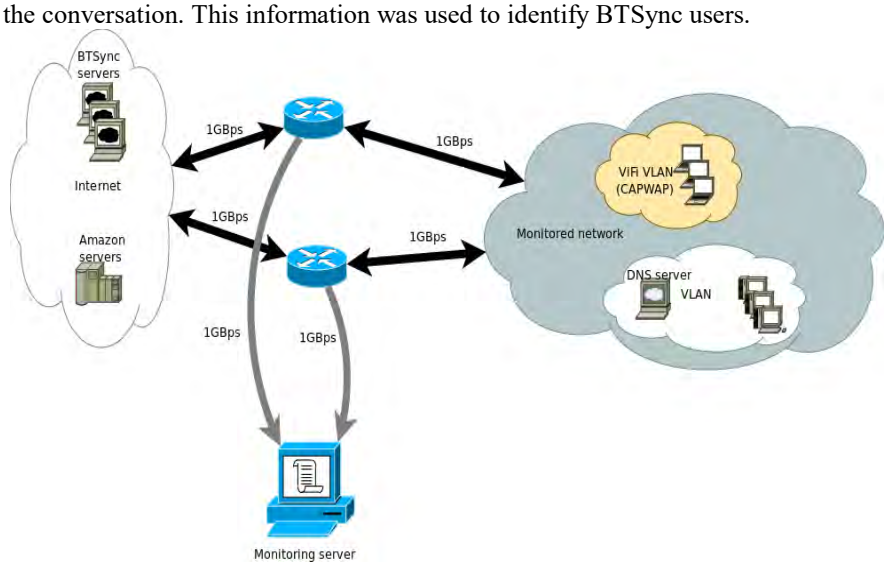

Fig. 5. The diagram of system component interaction

Data objects shared by BTSync are identified by 33-byte shareID. The first character specifies access rights for shared resource (Farina et al., 2014). Protocol analysis performed by community [13], [14] lead to discovery relation between shareID and "share" identity registered on the tracker. Remaining 32 bytes of the shareID are base32-encoded. This decoded value hashed using SHA1 produces "share", which may be found locally on the disk and it is visible in the network traffic sent to the tracker. This allows easily recognizing users of known shareIDs' (either from http://btsynckeys.com/ or obtained using other means) just by looking at network traffic going to/from BTSync trackers. Conversations with the tracker also hold unencrypted list of peers (returned to the client); client IP is sent to the tracker, as well.

Physical deployment of the experiment is shown in Fig 5.

The network to be monitored contains two 1Gbps failover links to a public Internet. Both paths are in use simultaneously. Inside of the monitored network, devices are attached to separate VLANs. Two VLANS were selected for a closer monitoring on a criteria of containing user devices (PCs, laptops, mobile devices, shared servers, etc.). A traffic to/from monitored network was mirrored to an interfaces attached to a monitor by a means available on routers (both simple mirroring and routing were used). A few obstacles for easy detection of BTSync users (as previously expected) were noticed:

1. VLAN, which contained WiFi devices, was using CAPWAP [17] encapsulation. The OS used on the monitor does not support this protocol and there is a limited selection of tools to process such traffic.

2. “Plain” VLAN had a forwarding/caching DNS server deployed locally, consequently, DNS queries could not be used to identify BTSync users.

3. Some BTSync infrastructure servers use Amazon servers to distribute content via HTTP, and IP addresses used to access BTSync related content (configuration updates), as resolved by DNS, change frequently.

4. Monitor had access only to 2Gbps traffic bandwidth (1Gbps per router monitored), while there was flowing 4Gbps traffic to/from the network (1Gbps incoming + 1Gbps outgoing traffic on each line from the network monitored to its routers because of full-duplex operation). Therefore, some traffic is not visible on a monitor.

A setup used to capture BTSync related traffic is shown Fig 6. Here re0 and re1 represent network cards where 1GBps interfaces with a mirrored traffic are attached. These interfaces are combined into single virtual interface lagg0 using OS lagg driver. VLANs vlan0 and vlan1 then are extracted from it using vlan driver. A number of BP filters (scripts) are started on virtual network interfaces. Some of these scripts, which generate a big amount of data very quickly, rotate captured buffers, and a script (match?) tests the previous buffer for implications of BTSync related traffic. Based on the result of this test, the buffer is either discarded or saved for later analysis. For performance reasons, only simple tests were used to prevent queueing of untested buffers, which would lead to swapping in the OS and as a result to a process stall.

DNS capture filter is a simple BP filter expression (vlan is needed to drop VLAN header):

```
vlan and port 53
```

Full command line (to rotate log every 6 hours) is as follows:

```
tcpdump -ni lagg0 -s 1500 -w dns_lagg0_%s.dump -G 21600 'vlan and port 53'
```

Similar command to capture suspected BTSync traffic was used, but with a different filter to capture any traffic to/from trackers and relays, any packet beginning with *BSYNC* string or traffic to/from *link.getsync.com* (at a time of the experiment it was resolving to 98.143.146.7):

```
vlan and (
    port 3000 and (
        host
52.0.104.40 or
52.0.102.230 or
52.1.40.103 or
52.1.1.135 or
67.215.231.242 or
67.215.229.106 or
66.63.177.26 or
54.235.182.157
    ) or (
tcp[20] = 66 and
tcp[21] = 83 and
tcp[22] = 89 and
tcp[23] = 78 and
tcp[24] = 67)
      or host 98.143.146.7
)
```

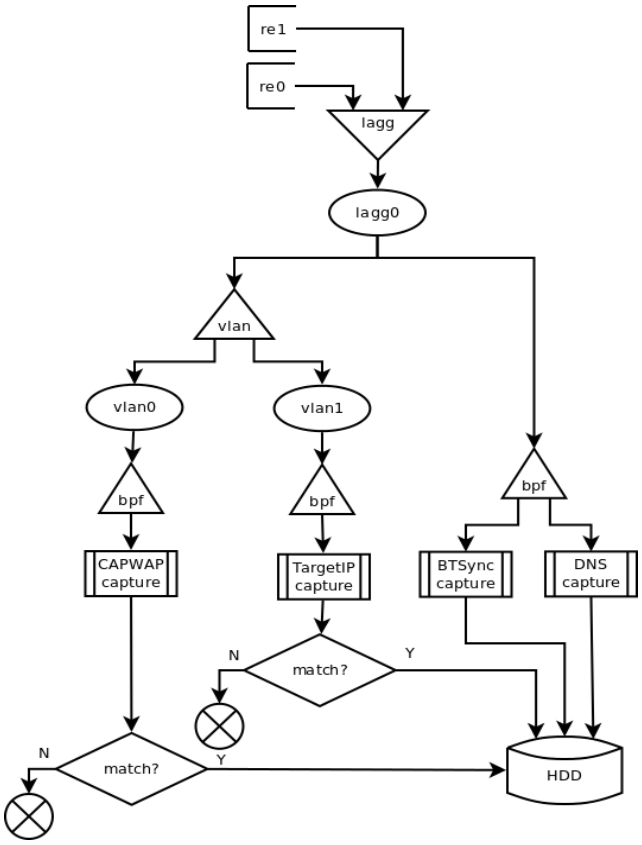

Fig. 6. Traffic capture setup

These two filters served as an early detection tool. Once a BTSync user's IP is detected, a *TargetIP* capture filter should be started. A sample command (N.N.N.N should be replaced with IP) is as follows:

tcpdump -ni vlan1 -w N.N.N.N.dump. -C 2000 -W 100 -z testscript.sh host N.N.N.N

This script captures all the traffic from an IP and rotates once 2GB of data is captured. When rotating, *testscript.sh* (match?) is run, which searches for the implications of BTSync usage and either saves dump file for a later analysis or discards it.

The same principle is used to capture CAPWAP traffic. Matching script used in the experiment was pretty simple, searching for *getsync.com* (domain used by BTSync Clients), *BSYNC* (frequently present in BTSync traffic) and *share20* (present in BTSync tracker traffic) strings in the captured files. If any of these values is found, file is saved for a later analysis.

During monitoring time of one month, only a single IP using BTSync client was detected with two shares registered on the trackers. SHA1 hashes of these shares were compared with the list from *htpps://btsynckeys.com*, but no matches were detected.

One conversation with a 52.0.104.40 with a *share32* bencoded field was observed. We have hidden some parts of the message, which may contain a private information:

00000000 00 00 00 6e 64 32 3a 6c 61 36 3a XX XX XX XX 9d ...nd2:l a6:XXXX.
00000010 ec 32 3a 6c 70 69 34 30 34 32 38 65 31 3a 6d 39 .2:lpi40 428e1:m9
00000020 3a 67 65 74 5f 70 65 65 72 73 34 3a 70 65 65 72 :get_pee rs4:peer
00000030 32 30 3a XX XX XX XX XX XX XX XX XX XX XX XX XX 20:XXXXX XXXXXXXX
00000040 XX XX XX XX XX XX XX 35 3a 73 68 61 72 65 33 32 XXXXXXX5 :**share32**
00000050 3a 63 a6 5a b1 a8 b8 80 5f 48 cc 08 e7 6b 19 31 :c.Z.... _H...k.1
00000060 cb ee 68 bb 7f 9e a7 54 fb c6 e3 9d e8 4c bc af ..h....T .....L..
00000070 9b 65 .e

The response was as follows:

00000000 00 00 00 90 64 32 3a 65 61 36 3a XX XX XX XX ff ....d2:e a6:XXXX.
00000010 2e 31 3a 6d 35 3a 70 65 65 72 73 35 3a 70 65 65 .1:m5:pe ers5:pee
00000020 72 73 6c 64 31 3a 61 36 3a XX XX XX XX 9d ec 32 rsld1:a6 :XXXX..2
00000030 3a 6c 61 36 3a XX XX XX XX 9d ec 31 3a 70 32 30 :la6:XXX X..1:p20
00000040 3a 10 a1 71 8e c8 49 09 99 e3 c8 ba e6 b7 ba 9c :..q..I. ....
00000050 b3 e7 b9 47 fa 65 65 35 3a 73 68 61 72 65 33 32 ...G.ee5 :**share32**
00000060 3a 63 a6 5a b1 a8 b8 80 5f 48 cc 08 e7 6b 19 31 :c.Z.... _H...k.1
00000070 cb ee 68 bb 7f 9e a7 54 fb c6 e3 9d e8 4c bc af ..h....T .....L..
00000080 9b 34 3a 74 69 6d 65 69 31 34 34 38 38 31 38 32 .4:timei 14488182
00000090 33 39 65 65 39ee

We have captured CAPWAP traffic in the same way. The BP filter sees CAPWAP traffic as UDP packets to/from 5246 and 5247 ports. Some buffers were selected by testscript.sh matching script for a later analysis. To decode this protocol Wireshark was used. Closer analysis showed that these captures were mostly false positives.

There were no conversations with trackers captured on CAPWAP VLAN. We have captured only single user that was downloading an older version (1.4.111) of BTSync client, while we were looking for the latest version (2.2.2.112) of BTSync client.

6. Conclusions

In order to aid the forensics investigator to fight against the cybercrimes, we suggested the methodology to track and discover the suspicious activity of BitTorrent Sync client on the computer network. The base of the methodology is the monitor inserted into the network close as possible to the suspect's computer. The monitor observes all the traffic coming to the suspect's computer and filters out the suspicious traffic for the further analysis. The monitor catches the IP addresses of the peers participating in the sharing of the file. During the monitoring, it is possible to calculate the public lookup SHA1 hash of the shared file. The comparison of the calculated hash with the list of publicly available hashes allows determination whether sharing of the file is legal or illegal. Then the forensics investigator on the base of the presented information can decide to prosecute or not the identified peers.

The experiment was carried out in two modes: simulating of the use of the BitTorrent Sync application for file download; monitoring of the traffic on the real computer network. The architecture of the used hardware, the communication details of the protocol and the scripts to obtain them are provided for both cases. The both experiments proved the ability of the presented methodology to track the users of the BitTorrent Sync protocol. The proposed methodology has a limitation, because it requires the seizing of the suspect's computer. However, this limitation is not specific to the proposed methodology only. The method [1] presented has the same limitation. Moreover, the seizing of the suspect's computer enables to collect the evidence and suspect can not hinder to this process.

The presented methodology can be applied to monitor the use of any BitTorrent protocol, since it needs to change only several patterns for the monitoring.

References

1. Park, S.; Chung, H.; Lee, C.; Lee, S.; Lee, K. Methodology and Implementation for Tracking the File Sharers using BitTorrent. Multimedia Tools Appl. 74(1), 2015, pp. 271-286.
2. Schmidt, A. H.; Antunes, R. S., Barcellos, M. P.; Gaspary, L. P. Characterizing Dissemination of Illegal Copies of Content through Monitoring of BitTorrent Networks. In: Proceedings of Network Operations and Management Symposium (NOMS), 2012: 327-334.
3. Liberatore M., Erdely R., Kerle T., Levine B. N., Shields C. Forensic investigation of peer-to-peer file sharing networks. Digital Investigation, Vol. 7, 2010, S95–S103.
4. Mansilha, R. B.; Bays, L. R.; Lehmann, M. B.; Mezzomo, A.; Gaspary, L. P.; Barcellos, M. P. Observing the BitTorrent Universe through Telescopes. In Proceedings of International Symposium on Integrated Network Management, 2011, pp. 1–8.
5. Wang, L.; Kangasharju, J. Measuring Large-Scale Distributed Systems: Case of BitTorrent Mainline DHT. In Proceedings of IEEE Thirteenth International Conference on Peer-to-Peer Computing (P2P), 2013, pp. 1-10.
6. Farina, J.; Scanlon, M.; Kechadi, M-T. BitTorrent Sync: First Impressions and Digital Forensic Implications. Digital Investigation, May 2014, 11(S1):S77 – S86.
7. Scanlon, M.; Farina, J.; Kechadi, M-T. BitTorrent Sync: Network Investigation Methodology. In Proceedings of Ninth International Conference on Availability, Reliability and Security (ARES 2014), Fribourg, Switzerland, September 2014, pp. 21-29.
8. Scanlon, M.; Farina, J.; Le Khac, N. A.; Kechadi, T. Leveraging Decentralization to Extend the Digital Evidence Acquisition Window: Case Study on BitTorrent Sync. Journal of Digital Forensics, Security and Law: Proc. of Sixth International Conference on Digital Forensics & Cyber Crime, New Haven, USA, September, 2014, pp.85-99.
9. Chow, K.P.; Cheng, K.Y.; Man, L.Y.; Lai, P.K.Y; Hui, L.C.K.; Chong, C.F.; Pun, K.H.; Tsang, W.W.; Chan, H.W.; Yiu, S.M. BTM – An Automated Rule-based BT Monitoring System for Piracy Detection. In Proceedings of Second International Conference on Internet Monitoring and Protection 2007, Jul 2007, pp. 2-7.
10. Bauer K., McCoy D., Grunwald D., and Sicker D. BitStalker: Accurately and Efficiently Monitoring BitTorrent Traffic. In Proceedings of the IEEE International Workshop on Information Forensics and Security (WIFS), London, UK, Dec. 2009, pp. 181-185.
11. Schrader, K. R.; Mullins, B. E.; Peterson, G. L.; Mills, R. F. An FPGA-Based System for Tracking Digital Information Transmitted Via Peer-to-Peer Protocols. International Journal of Security and Networks, Vol. 5, No. 4, 2010, pp.236-247.

12. Lee, J.; Kim J. Piracy Tracking System of the BitTorrent. International Journal of Security and Its Applications Vol.7, No.6, 2013, pp.191-198.
13. BitTorrent. Forum. http://forum.bittorrent.com/topic/21338-inofficial-protocol-specification/. Accessed on February 04, 2016.
14. GitHub. Picosync/workingDraft. https://github.com/picosync/workingDraft. Accessed on February 04, 2016.
15. ServerFault. Encoding to base32 from the shell. http://serverfault.com/questions/386199/encoding-to-base32-from-the-shell. Accessed on February 04, 2016.
16. BitTorrent Specification. Bencoding. https://wiki.theory.org/BitTorrentSpecification#Bencoding. Accessed on February 04, 2016.
17. CAPWAP. https://en.wikipedia.org/wiki/CAPWAP. Accessed on February 04, 2016.
18. BitTorrent. Sync 1.4 Help Center. http://sync-help.bittorrent.com/customer/portal/articles/1628463-link-structure-and-flow. Accessed on February 04, 2016.

Algimantas Venčkauskas received his Engineering degree in 1979 and PhD in 1987 at the Faculty of Informatics, Kaunas University of Technology (KTU), Kaunas, Lithuania. Currently he is Professor of Computer Science and head of Department of Computer Science at KTU. He has published more than 50 papers in various International Journals. The most of his research work was orientated on the study of internet technology, information security and cybercrime.

Vacius Jusas graduated from Kaunas Polytechnic Institute (Lithuania) in 1982. He received the D.Sc. degree from Kaunas Polytechnic Institute in 1988. Since 2006, he is professor at Department of Software Engineering, Kaunas University of Technology, Lithuania. He is author and co-author of more than 90 papers. He is Editor of journal "Information Technology and Control". His research interests include brain-computer interface, adaptive signal processing, forensics investigation, cybercrime.

Kęstutis Paulikas graduated from Kaunas University of Technology (Lithuania) in 1997. He received the D.Sc. degree from Kaunas University of Technology 2007. Since 1997, he has been working in various, mostly engineering, positions both in academia and industry. Since 2012, he is a lecturer at the Department of Applied Informatics, Kaunas University of Technology, Lithuania. He is author and co-author of more than 10 papers. His research interests include computer networks, forensics investigation, cybercrime.

Jevgenijus Toldinas graduated from Kaunas Polytechnic Institute (Lithuania) in 1980. He received the D.Sc. degree from Kaunas Polytechnic Institute in 1987. Since 1992, he is associated professor at Department of Computer Science, Kaunas University of Technology, Lithuania. He is author and co-author of more than 20 papers. His research interests include energy consumption in mobile devices, Internet of Things, digital forensics.

Received: February 12, 2016; Accepted: July 30, 2016