Forensics Analysis of an On-line Game over Steam Platform

Raquel Tabuyo-Benito[1(✉)], Hayretdin Bahsi[1], and Pedro Peris-Lopez[2]

[1] Tallinn University of Technology,
Center for Digital Forensics and Cyber Security, Tallinn, Estonia
{ratabu,hayretdin.bahsi}@ttu.ee
[2] Universidad Carlos III de Madrid, COSEC Lab, Getafe, Spain
pperis@inf.uc3m.es

Abstract. Currently on-line gaming represents a severe threat to the forensic community, as criminals have started to use on-line gaming as communication channels instead of traditional channels like WhatsApp or Facebook. In this paper, we describe a methodology developed after conducting an in-depth digital forensic analysis of the central artifacts of a popular video-game - Counter Strike Nexon Zombies video-game (Steam platform) - where valuable artifacts are those that related to the chatting features of the game. For our research we analyzed the network, volatile, and disk captures for two generated cases and focused on chat-feature inside and outside of the in-game rounds and the live chat done through YouTube Live Streaming. Our results provide the forensic community a complete guideline that can be used when dealing with a real criminal case in which there is a Steam video-game involved. Besides the forensic analysis, we found a security vulnerability (session hijacking) which was reported to the game manufacturer as soon it was discovered.

Keywords: Digital forensics · Network forensics · Windows forensics
Live forensics · On-line gaming

1 Introduction

Over the last few years, the video-games industry has spread to new markets and is reaching new kinds of players. Due to this growth, chat services offered by on-line video-games are becoming more and more popular among criminals as criminals consider these services to be safer methods of communication without detection. In 2015 after the Paris terror attacks, security analysts investigated new communication channels terrorists could use [12], and they found that terrorists could have used PlayStation 4 as the way to exchange messages without being discovered.

This paper focuses on *Counter Strike Nexon Zombies* (CSNZ) video-game, a game offered through Steam. This platform has a total of 67 million monthly

F. Breitinger and I. Baggili (Eds.): ICDF2C 2018, LNICST 259, pp. 106–127, 2019.
https://doi.org/10.1007/978-3-030-05487-8_6

active players and controls between 50–70% of the gaming market [20]. The paid version of CSNZ, Counter Strike: Global Offensive, has already been involved in a gambling scandal – there was an illegal betting market underneath with lots of teenagers participating [7]. Besides, it is the second most played game with large revenues and it is played on Windows/OS/Linux computers—note that 56% of gamers prefer PCs rather than others devices [11]. CSNZ is set in a war environment; we think that it could be one of the perfect ways for criminals to communicate as they can hide inside this atmosphere. Moreover, CSNZ has two game modes, one of them is the mentioned war scenario (Zombie mode) and the other one is a scenario similar to Minecraft (Studio mode), a three-dimensional game with the goal of building entire worlds with pixelated blocks. In these kinds of games, there are different kinds of servers which personalize the user experience and create a big network, which becomes attractive to DDoS attackers (e.g., Mirai botnet was originated from Minecraft video-game [6]). Additionally, CSNZ allows players to do YouTube live streaming while playing the game. This communication channel is also one of the targets of our study since it might be used maliciously.

Due to all of these facts, trying to obtain as much information as possible from this on-line game results in a significant contribution to the digital forensics community and makes a difference in the way forensic experts analyze a system when entering a crime scene. We hope the proposed methodology can provide guidance to forensics experts in the analysis of any other on-line games. In relation to previous works, researchers have payed particular attention to artifacts originated from widely used Social Networks like Facebook [22], or Instant Messaging (IM) tools like Skype [14] or WhatsApp [1,8,9]. However, on-line games, which have an enormous audience as well, offer similar communication channels, which criminals can misuse. Consequently, valuable information could be lost if they are not appropriately analyzed or not taken into account. As previously mentioned, some criminal cases like money laundering and DDoS attacks have the usage of video-games as the main factor, but our contribution is focused on how to find artifacts related to the chatting features that on-line games provide. Some forensic studies analyze video-games [4,10], but they focus primarily on post-mortem state analysis [3] and not in volatile or network traffic analysis. To the best of our knowledge, it is the first time that an in-depth forensic analysis of an on-line video game is conducted—the analysis performed by Moore et al. for XboxOne can be considered the work most closely related to our proposal [15] but our proposal is more complete since live forensic analysis is part of the used techniques.

2 Methods and Materials

The framework that we followed for dealing with evidences is the McKemmish forensic framework [13]. This forensic process consists on the following steps: (1) Identification of digital evidence; (2) Preservation of digital evidence; (3) Analysis of digital evidence; (4) Presentation of digital evidences.

To enable the acquisition of realistic data similar to the one that we would find in real world investigations, we decided to conscientiously follow the McKemmish approach with all the processes involved in forensic analysis:

Network forensics: we captured the network traffic for all the cases of study—we disconnected the system from any external connection after capturing the traffic.

Live acquisition: we performed an acquisition of volatile data (RAM memory dump) while being aware that we should create the least amount of changes in the system under inspection. The changes are explained and reported in order to ensure a precise preservation of the evidence and admissibility in court.

Post-mortem acquisition: we physically acquired the hard disk. Consequently we have a complete copy of the disk (including the unallocated space) allowing us to find all information about the game stored locally in the system.

Windows forensics: the OS of the disk acquired is a Windows 7 Professional image; consequently, we performed an in-depth analysis of it based on Windows forensics manners.

Once all the data was extracted, we continued with the interpretation of the evidences: network artifacts obtained from the traffic capture, the analysis of the RAM memory, windows registry, dedicated folder of the game in the system, windows system files and folders ($MFT,$LogFile, Prefetch folder, shortcuts, Recent folder, JumpLists, thumbnails and web information). After that, we pursued to find enough information to build the time-line of the case, when the user connected, with whom the user played, the list of friends, chat information, connection time-stamps, history of the game and user information like username and passwords. When analyzing the network information, we consider the disclosure of the session ID and this value might be used for a successful session hijacking.

We have used well-known and reputable free tools for forensic analysis. *Wireshark* [21] and *NetworkMiner* [16] are the chosen tools for the network forensic analysis. Regarding the live acquisition (RAM memory) and post-mortem acquisition (Hard disk), *FTK Imager* is used to create the respective images. The analysis of the runtime state of the system is conducted through the analysis of the RAM memory with *FTK Imager* and the *Volatility* framework. For the analysis of the files inside the disk (Windows OS), *Autopsy* and *FTK Imager* are the main tools used –other tools like *JumpListView* [18] or *ChromeCacheView* [17] are also employed for particular processes in the file analysis. The complete list of tools is shown in Table 3 in Annex A.

We made copies of the evidence in order to ensure its integrity. The copies were the ones analyzed. As we wanted to make a research in the most realistic way possible, in Table 4 in Annex A we provide the list of hashes of the evidence taken from the network capture, memory dump and disk cloning.

2.1 Cases of Study

We have considered two different cases for simulating how a user plays CSNZ with their friends, paying special attention to the different chatting alternatives offered by the game. A short description of each case is provided below and a detailed workflow diagram can be seen in Figs. 1 and 2 in Annex B, respectively. In these workflow diagrams we have included the moments when the acquisition processes (network capture, live and post-mortem acquisition) took place.

Case 1: two players, "TTUPlayer" and "UC3MPlayer", inside Zombie mode ("war" scenario). The former (TTUPlayer) will send an invitation into the game-room to the latter (UC3MPlayer). The chat will take place before entering the game, in the lobby. After that, both users will enter in the Zombie scenario and will chat again. While playing, users will use chat features that include text messages and voice audios.

Case 2: two players, "TTUPlayer" and "UC3MPlayer", in Studio mode ("mining" scenario) with a defined password to enter into the game-room. From this point, "UC3MPlayer" will do nothing (he will not participate in the game rounds or exchange messages). "TTUPlayer" will live stream through Youtube while playing the game. An external viewer, "Juanma", from the streaming video, will chat with this last mentioned player.

Both cases were run using VMWare virtual machines. We created two different virtual machines for each example. The reason why we decided to use a virtual machine instead of having the game installed directly in our system is so the game is isolated. Additionally, thanks to snapshots, we can always go to a previous state if we crash the program.

Regarding the specifications, we used VMWare Fusion 10.0.1 for each case with a Windows 7 Professional guest OS installed. The host system was a macOS High Sierra MacBook Air 2,2 GHz Intel Core i7 with 8 GB of memory. In each virtual machine, there was 2 GB of memory and 20 GB of disk space. Each virtual machine has a different IP concerning host's IP. For network capture, we executed Wireshark in the host system, so that we could capture network traffic in the middle of the communication between both users (man-in-the-middle).

In regards to the game, we installed Steam platform for Windows, as the game cannot be run without it. After that, the game was downloaded from Steam and installed on the Windows systems. The rest of programs installed on the system are the ones that are pre-included in Windows like Internet Explorer (only pre-installed browser on the system) or Notepad. For the second case of study, we also installed Google Chrome Browser, in order to enable all the chatting features offered by YouTube Live Chat, which was not possible to have with Internet Explorer. Due to the fact that this forensic analysis is mainly focused on the communication between players while using the game, we have used two virtual machines for each player. Since there are two cases of study, there are a total of 4 virtual machines involved in our study.

We had to create 2 Steam accounts and 2 players of CSNZ (one for "TTUPlayer" and another for "UC3MPlayer"). The login process is done

through Steam, and then the game starts. We acquired the disk image and RAM from the device (virtual machine) of "TTUPlayer" as s/he was the suspect in our forensic analysis. We created an Outlook email account for this player as well as a Google account for the YouTube Live Streaming. In addition, we created a YouTube Channel for "TTUPlayer" with "TTU Thesis" as the nickname. The viewer of the streaming, "Juanma", is subscribed to this channel. This option is not necessary for watching streamed videos in YouTube; however, we decided that the viewer is also a subscriber because we would receive a notification as soon as "TTU Thesis" starts doing live streaming. In Table 1 we summarize all the details of the suspicious player.

Table 1. Player information under suspicion

Game name	Counter strike nexon zombies
Game acronym	CSNZ
UserName Steam account	thttu
Steam User ID	76561198404618625
NickName Steam account	TTU - Thesis
Password Steam account	pAssWd123
Associated email in Steam account	ttu.thesis@outlook.com
CSNZ username	TTUPlayer
CSNZ Game ID	273310
Google account email	ttu.thesis@outlook.com
Google account username	TTU Thesis
Google account password	youtubeaccount1
YouTube Channel name	TTU Thesis
YouTube Channel identifier	UC-pPH8RIVmlFBMbJjUcPZzw
YouTube Live Streaming key	2ppw-5x2j-cz0z-6rrv
YouTube Live Streaming video	https://www.youtube.com/watch?v=Qd4OL0t3bBw
CSNZ Player 2	UC3MPlayer
YouTube Live Streaming viewer	Juanma
Zombie in-game room number	30220
Studio in-game room number	55349
Password Studio room	2tudio

3 Forensic Analysis and Results

We have built a guideline for the forensic community based on the analysis of both cases in terms of network capture, live acquisition and post-mortem acquisition. We found some common artifacts that reflect accurately how it is possible to find specific information from the game. More precisely, a forensic expert facing a Steam video-game (CSNZ in our particular case) and its chatting features should consider the artifacts listed in Annex C. A complete list of all the results can be found in Annex D.

3.1 Network Forensics Analysis

Common Network Artifacts. After capturing the network traffic with Wireshark (we saved it as PCAP file), we found some relevant artifacts that tie the user to CNSZ game. It is worth noticing that we also found a security vulnerability (session hijacking) that allowed to find additional forensic information. The process of identifying a user in Steam, after the authentication process, is done through cookies. Once the user is authenticated, he is redirected by default to the Steam Store website and the rest of Steam websites (community and help site) using the same cookies for user identification (user sessions). This login process is always the same for games on Steam. More precisely, two main cookies are sent over HTTP:

- `sessionid`: it is a CSRF token. The first time someone accesses to one of its websites, Steam assigns this cookie randomly. It can be any value; the only requirement is that it has to match with the session parameter of the POST requests of that person. Therefore, as it is not linked to any account or specific session, a user does not need to authenticate first for getting this cookie value.
- `steamLogin`: It is built with the 16-numerical characters of the `Steam_User_ID` + `%7C%7C` (two pipe characters) + 40–character uppercase session token in hexadecimal. This cookie is only generated after the user is successfully authenticated.

From the following Wireshark output, we can see that from the `steamLogin` cookie sent via HTTP, we can obtain the `Steam User ID` (first 16-numerical characters=76561198404618625).

```
Cookie:
browserid=1154470811777340210;
recentapps=%7B%2236690%22%3A1509888488%7D;
timezoneOffset=7200,0;
_ga=GA1.2.694143193.1508840752;
_gid=GA1.2.377692037.1510395285;
Steam_Language=english;
vractive=0;
connectedDevices=0;
steamLogin=76561198404618625%7c%7cA84173EA7E194BDBF401219535C0
2500DB96913B;
sessionid=35eb7107e8a693252a4fd0bc;
clientsessionid=13b982bfb86e5d0a
```

Note that Wireshark supports the searching of a specific packet—we look for a packet that contains the string "steamLogin" and *Narrow and Wide* are used as optional parameters. After disclosing this value, we can visit the website of the steam community (`http://steamcommunity.com/profiles/<SteamUserID>`) and obtain more forensic information such as: NickName user account; games played; last time they were played and total number of hours played; current

status of the user (on-line/off-line) and last time being on-line; and personal information that the user wants to share.

Session Cloning. When attempting to perform session cloning, as it was done in other studies [19], we found that it was possible to hijack the session of an authenticated user[1]. Considering the fact that the `sessionid` is created before the user authenticates, and that there are some cookies transmitted over HTTP and after HTTPS authentication, which are not changed, we found a way to clone the session of the user while s/he keeps it opened. The steps performed for the session hijacking are described below:

1. With the Wireshark capture, search inside the packet details for the string "`steamLogin`". Copy it and all its related cookies as "printable text".
2. Create an account on Steam (whatever user and password) and login to generate the `steamLogin` cookie. We logged-in with Google Chrome Browser and used "Edit this cookie" extension in order to inject the cookies related with the user session we want to clone, but the attack can be done with other browsers and add-ons.
3. Click on "Edit this cookie" extension in order to modify the cookies `_gad`, `sessionID` and `steamLogin` with the ones that we obtained from the Wireshark capture (Step 1). This way, we are injecting cookies belonging to another user to hijack the session. Finally, we click on the green icon to save those cookies.
4. Click on the "arrow-Back" to go to the previous initial page of login and then click on "refreshing" the page. We will be logged as the legitimate user (user under attack). We can see all the information related to that user: list of friends, games played, last connection, status (on-line), etc.

It is worth noting that this method is related to sessions, so if the target user logs out, we would not be able to continue using his session. From a forensic point of view, the possibility to clone the session gives us a lot of information as mentioned above.

YouTube Live Streaming Specific Artifacts. In the second case, we found that the network capture contains references that indicate that the suspect was doing a YouTube Live Streaming. In detail, there are RTMP packets (without TLS/SSL encryption), which are used for live streaming. More specifically, there must be a packet that shows the connection with the main server of YouTube, the method used for this purpose is: `connect('live2')` and the associated URL is: rtmp://a.rtmp.youtube.com/live2.

Besides the live video streaming via Youtube is unequivocally identified by an identifier named "key". It is included in the method called `releaseStream('<key>')`. This key is unique for each user, and even if it was

[1] This vulnerability was reported to Valve Corporation via email as soon as it was discovered.

not possible to watch the video knowing this key, it could be used for doing YouTube Live Streaming with the account of another user.

Moreover, we also recovered the encoder that was used for video streaming. Thanks to this information, the forensic expert can watch the video if there is any possibility to extract it from the copy of the disk, as we will see in Sect. 3.3. In our case the encoder used was VLC and this information can be found by analyzing the RTMP packets with Wireshark.

3.2 Volatile Memory Analysis

We acquired two pieces of evidence (before and after logging out from the game) from the RAM (MEM files) using FTK Imager installed in an external USB. See Figs. 1 and 2 in Annex B. After that, we have analyzed both files with FTK Imager and Volatility. Note that the chat messages can only be extracted whether the user did not logout (Live acquisition 1).

Volatile Memory Analysis with FTK Imager. The methodology that we have developed in order to find forensic artifacts after a live acquisition is based on keywords and similar to the study performed about the XboxOne with Autopsy [15]. FTK Imager has a built-in search tool so, by doing right-click at the beginning of the memory dump and clicking on *Find* we can search for any word. It is also possible to do a more elaborate search using a regular expression. The keywords and regular expressions that are useful for finding valuable forensic data with respect to the game are summarized in Table 2.

Table 2. Keywords for memory analysis

Valuable Forensic data	Keyword
Username Steam account	`SteamUser`
NickName Steam account	`PersonaName`
Password Steam account	`password=`
Steam User ID	`steamid`
CSNZ Game ID	`steamGameID`
Associated email address	`\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]2,\b`
Friends who are on-line	`has logged in`
Players involved in the game	`has joined`
Room number and password	`(#No.room)`
YouTube Live Streaming Traces	`youtube`
	`rtmp.youtube`
	`codec`

As previously mentioned, from volatile memory, it is possible to obtain the messages sent and received by the user but only if s/he did not logout when taking the memory dump. In Counter Strike Nexon Zombies, the user can exchange messages directly without starting a game round, inside in-game rounds and also while doing YouTube Live Streaming. In those cases, messages can be sent to all players (`ALL` tag), to your family group -list of friends- (`Family` tag), or to the party group (`PARTY` tag). Consequently, using the keywords linked to the different recipients who can be addressed in a message, we are able to find the information about the exchanged messages in the whole memory dump. The following three layouts of messages are used in the game:

1. Chat in the lobby: (Type of receiver) [Nickname of the sender] : message
2. In-game chat: [Type of receiver] Nickname of the sender : message
3. YouTube chat: [YOUTUBE] Nickname of the sender : message

Therefore, the keywords to discover chat messages are based on the type of recipient: `All`, `Family`, `Party` and `YOUTUBE`. Once we have found the messages, we can state to where they were sent to (lobby, in-game or YouTube) based on the mentioned layouts.

Volatile Memory Analysis with Volatility. The network connections, as well as the processes used, can be obtained using Volatility Framework. First of all, to retrieve information with Volatility, we have to select the proper profile—in our particular case Windows7SP1x64. The running processes can be obtained with `pslist` command. To assure that the game was played, a reference to Steam as well as Counter Strike should appear in the output. Additionally, if a web browser was used, it could indicate that the user was doing a live chat on YouTube. The above can be confirmed taking a look to the open network connections (`netscan` command). We found that there was also a connection from Google Chrome browser which might indicate that the user was doing a chat via YouTube Live Streaming.

3.3 Disk Analysis

We performed a post-mortem acquisition with FTK Imager installed in an external USB. The images were saved in E01 format—it uses compression and contains a separate metadata file. The methodology used for the analysis with FTK Imager is described below. In particular, the files analyzed were inspired by previous forensics analysis [19].

Windows Registry. In NTUSER.DAT and SOFTWARE hives we found some references (e.g., the path where it was stored) to the game that shows it was installed in the system. Besides, from a forensic point of view, the dates and times are relevant as they are useful for building a time-line of the case.

In order to find this information, we can extract those hives with FTK Imager and open them with Access Data Registry Viewer.

NTUSER.DAT is stored in \Users\<username> while SOFTWARE is stored in SystemRoot\System32\config.

- **NTUSER.DAT:** In this hive, there must be a folder called Software. The path to the CSNZ game should be NTUSER.DAT\Software\Valve\Steam\Apps\273310. The description of it shows how it corresponds to Counter Strike Nexon Zombies. The last written time corresponds to the last time the game was played. In addition, in the path NTUSER.DAT\Software\Nexon\CStrike-Online it also appears the last written time when the game was played. Finally and regarding the chat of YouTube Live Streaming, it is needed that the user has installed a web browser on the system. Consequently, there should be an entity in the hive. For example, in our case, NTUSER.DAT\Software\Google\Chrome is the path to Chrome Browser.
- **SOFTWARE:** In this hive, there is an important reference to the game that shows when it was installed in the system. The way to find this information is by looking at the last written time of the Uninstalling CSNZ software since this program was stored in the system (SOFTWARE\Microsoft\Windows\CurrentVersion\Unistalled\Steam App 273110) at the same time the game was installed. Additionally, the path to the game also appears in this hive in the description of the SOFTWARE\Wow6432Node\Valve\Steam\Apps\273110 entity.

Shortcuts. By default, there should be a shortcut to Steam on the Desktop. Steam is needed for executing CSNZ. Besides, there could be a shortcut for the game as well. However, the user has to execute Steam first and then CSNZ. It can be found directly with FTK Imager under the Users\<Username>\Desktop path.

Prefetch. The Prefetch folder is stored in \Windows\Prefetch and it can be extracted with FTK Imager. After its extraction, we can open it with WinPrefetchView. There will be references to Steam and CSNZ video-game. From a forensic point of view, it is very relevant since it shows all the programs affected by the game and when they were executed. Consequently, we can assure that the game depends on Steam and we correlate facts between the mentioned programs and the game. Besides, we can see that CSNZ video-game is related to NAR files. The analysis of these files is explained in the section below.

NAR Files. From the Prefetch analysis, we found that CSNZ uses the NAR files (specific CSNZ file format). We analyzed them with NAR extractor tool, after obtaining them from \Program Files (x86)\Steam\steamapps\common\CSNZ\Data\cstrike.nar. Unfortunately, those files are used for modeling the game but they do not give any information about the user or that could be useful in the forensic case.

Jump Lists. We can extract the Jump Lists with FTK Imager from the path `%APPDATA%\Microsoft\Windows\Recent\AutomaticDestinations\[AppID] automaticDestinations-ms`. Jump Lists can be viewed with JumpListView to display the recent documents. Forensically, it is useful as it could lead to interesting files that are not considered beforehand. In our case, after checking them, we realized that CSNZ video-game stores by default screen-shots taken during the game rounds and videos—the default folder is "`Documents`". The analysis of the Jump Lists is useful to find the file locations because the user could change the default ones. These screenshots and video files are studied in the following section.

Documents Folder. CSNZ video-game stores by default the screen-shots taken by the user during an in-game round. Besides, it makes screen-shots automatically when a round finishes. Concerning the YouTube Live Streaming, it stores the video in MP4 format once the user finishes doing the streaming. From a forensic point of view, this is very valuable information since the chat messages can be watched and voice messages can be listened directly from this file. Inside the `Documents` folder, there are two directories: one for Zombie mode (`"Counter Strike Nexon Zombie"` directory) and the other one for Studio mode (`"Counter Strike On-line"` directory). The directories with the screen-shots and videos are stored inside these folders. The screen-shots can be seen directly with FTK Imager or any other Image Viewer program. The video files have to be exported from the Video Capture folder prior to be watched. From the network analysis, we already knew that the encoder used for the videos was VLC. Therefore, we opened the videos with VLC media player and we can disclose the whole YouTube chat. The first 9 characters of the file names correspond with the dates when the videos were recorded.

MFT. We can extract the $MFT with FTK Imager and convert it into CSV file with Mft2csv tool. Taking advantage of the CSV file, we can search for references about Counter Strike and find where the data is stored. From this analysis, we found that by default CSNZ folder is stored in the system in the `\Program Files (x86)\Steam\steamapps\common\CSNZ` path. In addition, there are also two directories (`\Program Files (x86)\Steam\` and `\Program Data\Nexon\Common`) related to the game.

Steam Dedicated Folders. We have focused our efforts on Nexon and Steam folders. The former facilitates timing information and the latter provides information linked with the user or the game.

– **Nexon folder.** The analysis of the Nexon folder gives handy forensic information since it contains (`\Program Data\Nexon\Common\nmcogame.log`) a log file related with the messenger feature of the game. Using this, we can assure that the user was chatting at a specific time period. In detail, CSNZ

loads a messenger module when a player uses the chat. In the log, there are references to this module and its associated time-stamps.

- **Steam folder.** Inside the Steam folder, there are some configuration and log files that provide information such as the Steam `UserID`, the Steam `Username`, the `ID` of the game, etc. The extension of the configuration files is `VDF` and these files can be opened with Notepad++. In the folder `\Program Files (x86)\Steam\config\` we found "config.vdf" which contains the `userID` and the `username`. In this folder there is also a "loginusers.vdf" file which stores the logged users. It provides the `Username`, the `Nickname`, `UserID` and the last time accessed (EPOCH to UTC time conversion is needed). Additionally, there is another configuration file ("localconfig.vdf") stored in `\Program Files (x86)\Steam\userdata\<localIDnumber>`, which links undoubtedly the user with the game since it contains the `GameID` and the `Username`. Next to the `GameID`, we can also find the last played time.
  In addition, the remote connection file ("connections.txt") is stored in `\Program Files (x86)\Steam\logs\` path. It is a TXT file that contains all the remote connections with the user's PC. Therefore we can disclose with whom the user is playing (host name and IP address). Additionally, there is a database file (`Users\<user>\AppData\Local\Steam\htmlcache\Cookies`) that stores some cookies. It is an SQLite database, which can be extracted with FTK Imager and opened with DB Browser for SQLite. In particular, there is a table with the `sessionIDs` and their corresponding time reference (EPOCH time), which represents the last times the user accessed into Steam. As mentioned in Sect. 3.1, it is possible to do a session cloning if the user is still logged into Steam platform.

Recent Files. The `Recent` folder can show the recent activity of the user linked to the game. It can provide information such as the paths to specific files stored in the system and that the user could have modified. The Recent folder (`Users\<user>\AppData\Roaming\Microsoft\Windows\Recent`) can be exported with FTK Imager and be opened with RecentFilesView tool. In our case, it provides information about the screen-shots taken and the folders used for their storage.

Thumbnails. The `Thumbcache` stored in `Users\<user>\AppData\Roaming\Microsoft\Windows\Explorer\thumbcache_*.db` can be opened with ThumbCache Viewer tool. It provides information about the game: path location and the used OS (Windows 7 in our particular case).

LogFile. The `\$LogFile` contains information about the logged users. In fact, it is the same information as in ``loginusers.vdf''. However, as it is a Windows System file, it is more difficult to be modified by a normal user. Note that ``loginusers.vdf'' file could be edited by a user with a Text Editor. The comparison of both files will assure the integrity of the data.

Web Browsing Information. For the YouTube Live Streaming, we used Chrome Browser. In the forensics analysis, we were able to find some references about YouTube, but we could not retrieve the exchanged chat messages. Fortunately, some time references helped to create a time-line of the case. We focused on three files stored in `Users\<user>\AppData \Google\Chrome\Default\` and extracted with FTK Imager:

- **Cookies:** it can be opened with DB Browser for SQLite and contains the `sessionIDs`.
- **Cache:** it can be opened with ChromeCacheView and contains traces about the YouTube Live Streaming activity.
- **History:** it can be opened with ChromeHistoryView and contains references to the streaming notifications that correspond to the received messages. Each time the user receives a message, there is a new notification.

Autopsy can be used to find web-surfing information since it categorizes automatically the searches done. The most relevant findings concerning YouTube Live Streaming are the following ones:

- **Link to the streamed YouTube video.** After performing the live streaming, the video is automatically uploaded to YouTube. It will be kept in the user's channel unless it is deleted manually. As previously mentioned, it was found in the `History` file. Using this file, we can know the YouTube channel name and the links to the videos that the user has uploaded.
- **Email information.** This represents the email used for the Google account to access to YouTube. It is found in the `Login Data` file of the `Default` folder.

Analysis with Autopsy. Apart from the findings related with YouTube Live Streaming, additional information can be obtained using the keyword search feature. In particular, we found the password associated with the Steam account of our suspect user. The way to obtain this password is by using "`password`" as keyword. Autopsy searches inside the file system data, carved files and unallocated space. We found the password in the unallocated space and carved data. There is also a file (`Users\<user>\AppData\Local\Steam\htmlcache\Cache\data_1`) that can be used to obtain this password—even though the user did not check the option of "password remember". Although this finding may or may not happen, it is worthy to consider that the password could be stored in this path or in the unallocated space or the carved data. Additionally, there are some links of CSNZ and therefore we can assure that the user was playing the game at a particular time.

Furthermore, Autopsy performs data carving by default and shows the results obtained. Nevertheless, after analyzing the carved files, we did not find any more relevant information apart from the password mentioned previously. Besides, we also examined the unallocated space by keyword searching without any other important finding.

3.4 Main Chatting Artifacts Summary

In previous sections, we have seen that there are some artifacts that suggest that the chatting features of the CSNZ game were used. These findings can be divided into two:

Disk analysis:

- **Messenger module logs:** It shows when the messenger module was loaded which is each time a player uses it. Thus we know when the user was chatting. If we correlate these timestamps with the ones found in the network connection logs, we are able to identify with whom the user was playing (IP source and destination).
- **YouTube Live Streaming:** If the user was doing YouTube Live Streaming, apart from the previous logs, we can find the streamed video and see the whole chat conversation done between players (messages sent and received). Besides, the message notifications shown in the `Google Chrome History` correspond to the messages received. In this way, we can identify when the user received a message—note that it is not possible to see the content of the message; this information is stored in the client's computer. Considering the data privacy policy by Google[2], Google stores the videos uploaded but there is no reference about live chat messages, however, in the video, as seen previously, it is possible to see the messages content.

Volatile memory: We can obtain the whole conversation (messages sent and received) if when doing the live acquisition, the user did not log out from the game and/or switched off the computer. In fact, using the layouts mentioned in Sect. 3.2, we can identify where it took place (lobby, in-game or YouTube). Furthermore, if we correlate these findings with the network connections logs, we can know with whom the user was playing (IP source and destination).

3.5 Anti-forensics

Anti-forensics was not the main concern of our study, however we identified that most of the logs stored in the disk, which contain relevant forensic data (username, user IDs, connection timestamps, messenger logs, etc.) can be edited with any text editor and without requiring a great expertise in destroying evidences. This data can be easily modified or removed and consequently, relevant forensic data could be lost accidentally or on purpose. If the files are deleted and the data is not overwritten in the disk, it could be recovered with forensic data carving procedures.

When the computer is rebooted or power-cycled, the non-temporal data stored locally in the system remains unchanged; nevertheless, the information in the volatile memory is lost. More specifically, which respect to the chatting features of the game, it will not be possible to obtain the chat messages sent and received. As the messages can only be recovered from volatile memory, if the user wants to hide his conversations, he just needs to log out from the game

[2] https://privacy.google.com/intl/en/your-data.html.

and/or switch the computer off and all the data will be flushed out. Besides, if the user deselects the default option of storing screen-shots and videos from the game, the video(s) with chat information from YouTube Live Streaming will not be available locally in the suspect's computer.

Finally, in case of an unexpected system shutdown due to a power error or critical system failure, the operating system will go back to a previous status of the file system with relevant forensic data stored in the `$LogFile` folder.

4 Conclusions

In this paper we performed a digital forensics analysis of *Counter Strike Nexon Zombies* video-game. This game allows users to play cooperatively or fighting between them. The game provides chat features, so players while playing can communicate with text or audio messages but they can also chat without being in an in-game round. It is worth noticing that messaging features offered in on-line gaming are lately being used for criminal purposes: money laundering [5], hidden channel communication by terrorists [12], etc. Additionally, a great number of players are making money thanks to YouTube Live Streaming [2], by recording themselves while playing a video-game and interacting at the same time with their viewers via the live chat.

Our forensics findings about chatting features for a Steam based on-line game, to the best of our knowledge, have not been documented yet, and could serve as a guideline for the computer forensic community when facing with on-line games over the Steam platform. Using two cases of study, we provide a full analysis from a network, hard disk and volatile memory perspective, being the last one not considered in similar forensic analysis developed for messaging applications such as Facebook or WhatsApp [9,22].

The developed methodology can be generalized to games that belong to Steam platform since the login procedure is identical for all the games in that market. In fact, we have checked that the login via a web browser is identical to the login procedure used in the video-game. Furthermore, we have found that the related data on the disk about the game is stored in the "`Steam`" folder instead of an specific folder for each game. Thus, an examiner should focus on the "`Steam`" folder as it contains most of the relevant data. Additionally, we strongly suggest analyzing the volatile memory first, if it is possible to perform a memory dump when taking the evidences. The valuable data that could be found in RAM is enormous, even being possible to recover the whole chats conversations. Finally, we would like to mention that since we have done the forensics analysis manually, the development of a tool for an automatic analysis is pending as a possible future work.

Acknowledgments. This work has been supported by the CAM grant S2013/ICE-3095 (CIBERDINE: Cybersecurity, Data, and Risks) and by the MINECO grant TIN2016-79095-C2-2-R (SMOG-DEV—Security mechanisms for fog computing: advanced security for devices).

Appendix A List of Tools and Evidences

Table 3. Used tools

Network forensics	Wireshark	
	NetworkMiner	
Live acquisition	FTK Imager	
	Volatility	
Post-mortem acquisition	FTK Imager	
Windows forensics	FTK Imager	Autopsy
	Mft2csv	Microsoft Excel
	NAR extractor	JumpListView
	WinPrefetchView	RecentFilesView
	AccessData Registry Viewer	Regripper
	Notepad++	Thumbcache Viewer
	VLC media player	ChromeCacheView
	ChromeCookiesView	DB Browser for SQLite

Table 4. Evidences

Case name	MD5 hash	Description
Case1.pcap	ec451d2fb12890c4b6cea7dbf3992233	Network capture Case 1
Case1-1.mem	257249692e38dd3d0b52a774beb2f00e	Memory dump Case 1 before the user logs out
Case1-2.mem	1d5b55b11bd250603d4a0742e32cda07	Memory dump Case 1 before the user logs out
Case1.e01	40750e123186f1a8b7bcfaa7aded7b1d	Disk copy of Case 1
Case2.pcap	dc56888f8ba320bdbfbc3b9a8ea624e0	Network capture Case 2
Case2-1.mem	59d7d18a8b472113c96c335a85aab2d7	Memory dump Case 2 before the user logs out
Case2-2.mem	bf41cce069f78cda38247ac0ec9a1d8d	Memory dump Case 2 before the user logs out
Case2.e01	ea2d48011ae501968683727b4bf4269b	Disk copy of Case 2

Appendix B Workflows of Case 1 and 2

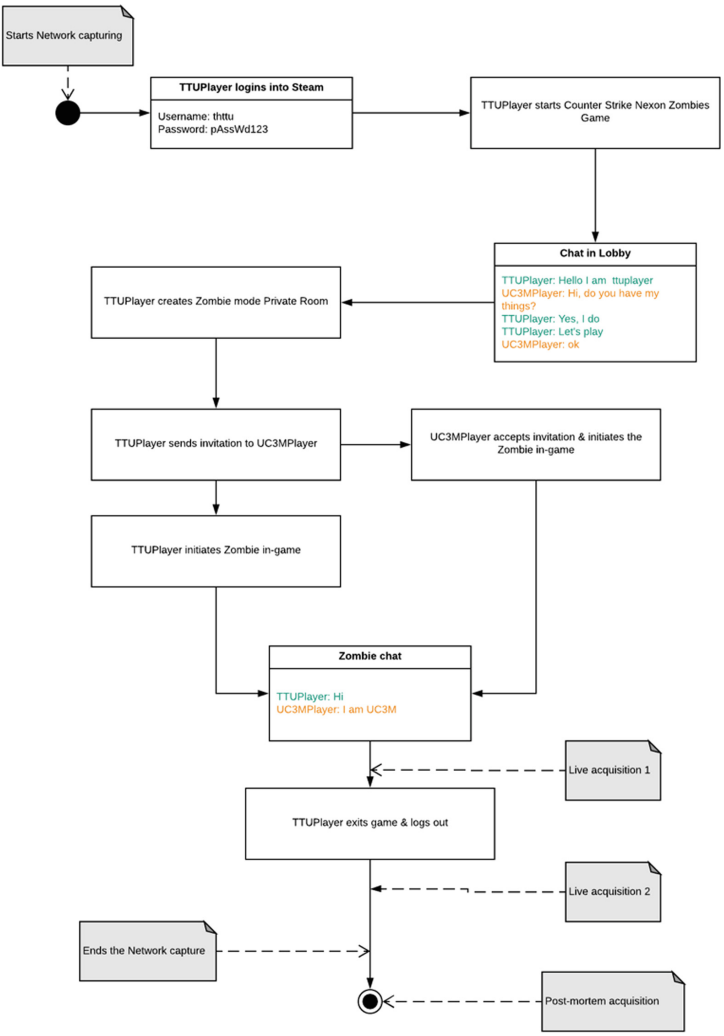

Fig. 1. Workflow case 1 (Zombi Mode)

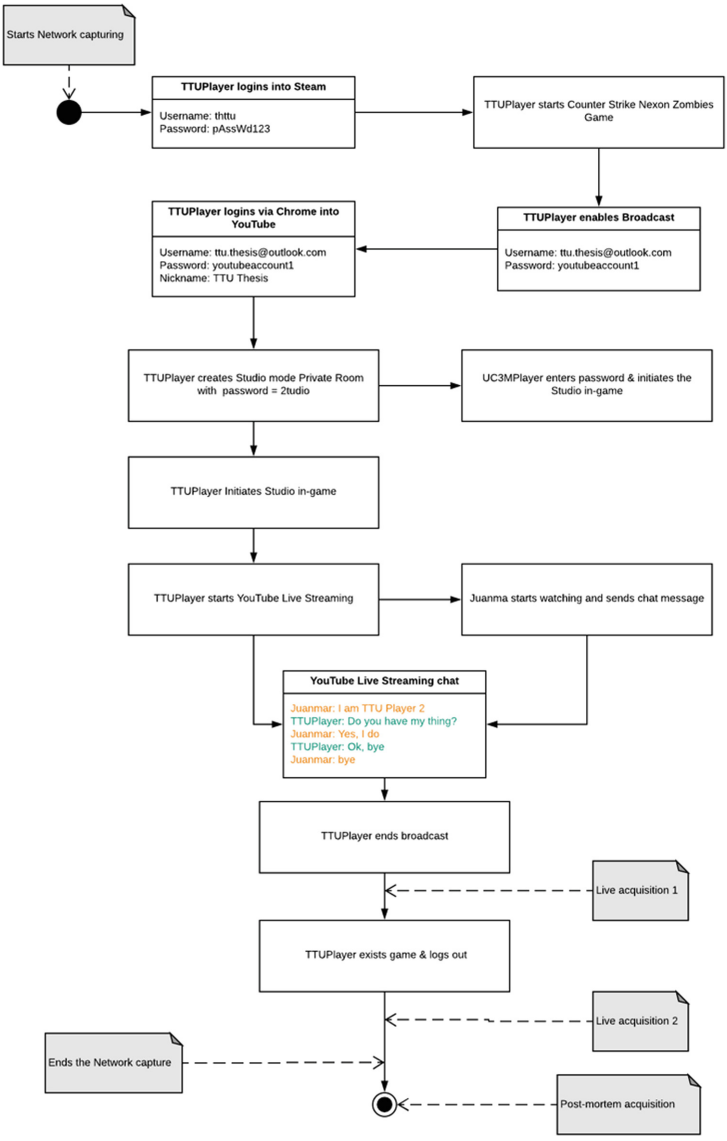

Fig. 2. Workflow case 2 (Studio Mode)

Appendix C Primary Artifacts

Table 5. Primary artifacts

	Network analysis	**Volatile memory analysis** (keywords)	**Disk analysis**
UserID	steamLogin cookie	steamID	\Program Files(x86)\Steam \config \config.vdf
User name	http:// steamcommunity. com/profiles/ <SteamUserID>	SteamUser	$ LogFile \Program Files(x86)\Steam\config \loginusers.vdf
User password		password=	Users \<user> \AppData \Local \Steam \htmlcache \Cache \data_1
Nickname	http:// steamcommunity. com/profiles/ <SteamUserID>	PersonaName	$LogFile \Program Files(x86)\Steam\config \loginusers.vdf \Program Files(x86)\Steam \userdata \<localIDnumber>\localconfig.vdf
List of friends	Session cloning	has logged in	
Chat information		Lobby chat:(Type of receiver) [Nickname of the sender] : message In-game chat:[Type of receiver] Nickname of the sender : message YouTube chat:[YOUTUBE] Nickname of the sender : message	Messenger module: \Program Data \Nexon\Common \nmcogame.log Remote connections: \Program Files(x86)\Steam\logs \remoteconnections.txt
YouTube Live Streaming traces	connect('live2') rtmp://a.rtmp.youtube. com/live2 releaseStream('<key>')	youtube channel_id rtmp.youtube codec	Users \<user> \AppData \Google \Chrome \Default \History
GameID	http:// steamcommunity. com/profiles/ <SteamUserID>	steamGameID	\Program Files(x86)\Steam\steamapps\CSNZ \Bin\steam_appid.txt
User timestamps: - Game installed - Last time accessed - Last time played	http:// steamcommunity. com/profiles/ <SteamUserID>	volatility_2.6_win64_ standalone.exe pslist -profile=<profile> âĂŤf <memoryimage>	SOFTWARE \Microsoft \Windows \CurrentVersion \Unistalled \Steam App <GameID> NTUSER.DAT \Software\Valve \Steam \Apps\<GameID> \Program Files(x86)\Steam \config \loginusers.vdf NTUSER.DAT\Software\Nexon \CStrike-Online \Program Files(x86)\Steam\userdata \<localIDnumber>\localconfig.vdf
History of the game	http:// steamcommunity. com/profiles/ <SteamUserID>		\Program Files (x86)\Steam\logs \remote connections.txt Documents folder

Appendix D Summary of Cases Results

Table 6. Results of case 1

	CASE 1					
	Network	Steam Session cloning	Volatile memory	Disk		
				Steam & CSNZ dedicated folders	Windows registry	Windows system files & folders**
CSNZ Game name	✓	✓	✓	✓	✓	
CSNZ Game mode			✓			
Game acronym			✓	✓		
UserName Steam account		✓	✓	✓		✓
Steam User ID	✓	✓	✓	✓		✓
NickName Steam account	✓	✓	✓	✓		✓
Password Steam account			✓	✓		
Associated email in Steam account		✓	✓			
CSNZ username			✓	✓		
CSNZ Game ID			✓	✓	✓	
Chat messages sent in lobby			✓*			
Chat messages received in lobby			✓*			
CSNZ Player 2			✓	✓		
Zombie in-game room number			✓			
CSNZ family members (list of friends)			✓			
Chat messages sent in Zombie game			✓*			
Chat messages received in Zombie game			✓*			
Game execution logs (last time accessed and last played time)	✓	✓	✓	✓	✓	✓
Network Session IDs	✓	✓		✓		
Network connections	✓		✓	✓		
User status (online/offline)	✓	✓	✓			

* Chats can only be obtained if the acquisition of volatile memory is done before the user logs out.
The Windows files and folders considered are: Shortcuts, Prefetch, Jump Lists, LogFile, MFT, Thumbnails, Recent files, Google Chrome Default folder (Cookies, Cache, History and Login Data files).
*** Chats can only be obtained if the video is still uploaded to YouTube (the user didn't delete it manually) or if the user didn't disable the option of saving the video in the system.

Table 7. Results of case 2

	CASE 2					
	Network	Steam Session cloning	Volatile memory	Disk		
				Steam & CSNZ dedicated folders	Windows registry	Windows system files & folders**
CSNZ Game name	✓	✓	✓	✓	✓	
CSNZ Game mode			✓			
Game acronym			✓	✓		
UserName Steam account		✓	✓	✓		✓
Steam User ID	✓	✓	✓	✓		✓
NickName Steam account	✓	✓	✓	✓		✓
Password Steam account			✓	✓		
Associated email in Steam account		✓	✓			
CSNZ username			✓	✓		
CSNZ Game ID			✓	✓	✓	
Google account email						✓
Google account username			✓			✓
YouTube Live Streaming main server URL	✓		✓			
YouTube Live Streaming encoder	✓		✓			
YouTube Channel name			✓			✓
YouTube Live Streaming key	✓		✓			
YouTube Live Streaming video			✓			✓
YouTube Live Streaming viewer			✓			
Studio in-game room number			✓			
Password Studio room			✓			
YouTube Live Streaming chat messages sent			✓*	✓***		
YouTube Live Streaming chat messages received			✓*	✓***		
CSNZ Player 2			✓	✓		
Game execution logs (last time accessed and last played time)	✓	✓	✓	✓	✓	✓
Network Session IDs	✓	✓		✓		✓
Network connections	✓	✓	✓	✓		✓
YouTube Live Streaming execution logs	✓					✓
User status (online/offline)		✓	✓			

* Chats can only be obtained if the acquisition of volatile memory is done before the user logs out.
** The Windows files and folders considered are: Shortcuts, Prefetch, Jump Lists, LogFile, MFT, Thumbnails, Recent files, Google Chrome Default folder (Cookies, Cache, History and Login Data files).
*** Chats can only be obtained if the video is still uploaded to YouTube (the user didn't delete it manually) or if the user didn't disable the option of saving the video in the system.

References

1. Anglano, C.: Forensic analysis of whatsapp messenger on android smartphones. Digital Invest. **11**(3), 201–213
2. Bourne, W.: Youtube vs. twitch: how to make money live streaming (2018). https://goo.gl/cxafwX. Accessed 05 July 2018
3. Daniel, L.E.: Multiplayer game forensics (2018). https://www.forensicmag.com/article/2010/05/multiplayer-game-forensics. Accessed 02 Feb 2018
4. Davies, M., Read, H., Xynos, K., Sutherland, I.: Forensic analysis of a sony playstation 4: a first look. Digital Invest. 12, 81–89
5. Editor: Why online gaming is the new frontier for cybercrime (2015). https://www.welivesecurity.com/2015/12/24/online-gaming-new-frontier-cybercriminals/. Accessed 20 Jan 2018
6. Graff, G.M.: How a dorm room minecraft scam brought down the internet (2017). https://www.wired.com/story/mirai-botnet-minecraft-scam-brought-down-the-internet/. Accessed 20 Jan 2018
7. Grayson, N.: The counter-strike gambling scandal, explained. https://steamed.kotaku.com/why-people-are-flipping-out-over-the-counter-strike-gam-1783369102. Accessed 20 Jan 2018
8. Jhala, G.J.: Whatsapp forensics: decryption of encrypted whatsapp databases on non rooted android devices. J. Inf. Technol. Software Eng. **5**(2), 1 (2015)
9. Karpisek, F., Baggili, I., Breitinger, F.: Whatsapp network forensics: decrypting and understanding the whatsapp call signaling messages. Digital Invest. **15**, 110–118 (2015)
10. Khanji, S., Jabir, R., Iqbal, F., Marrington, A.: Forensic analysis of xbox one and playstation 4 gaming consoles. Digital Invest. **12**, 81–89 (2016)
11. Lofgren, K.: Video game trends and statistics - who's playing what and why? (2017). https://goo.gl/9CeDFb. Accessed 20 Jan 2018
12. Mastroianni, B.: How terrorists could use video games to communicate undetected (2015). https://goo.gl/F5Jvnb. Accessed 20 Jan 2018
13. McKemmish, R.: What is Forensic Computing?. Australian Institute of Criminlogy, Canberra (1999). Art 118
14. McQuaid, J.: Skype Forensics: Analyzing Call and Chat Data from Computers and Mobile. MAGNET Forensics, Herndon (2014)
15. Moore, J., Baggili, I., Marrington, A., Rodrigues, A.: Preliminary forensic analysis of the xbox one. Digital Invest. **11**, S57–S65 (2014)
16. NETRESEC: Networkminer. http://www.netresec.com/?page=NetworkMiner, http://www.netresec.com/. Accessed 02 Feb 2018
17. NirSoft.: Chromecacheview. https://www.nirsoft.net/utils/chrome_cache_view.html. Accessed 02 May 2018
18. NirSoft: Jumplistview. https://www.nirsoft.net/utils/jump_lists_view.html. Accessed 02 May 2018
19. Sgaras, C., Kechadi, M.-T., Le-Khac, N.-A.: Forensics acquisition and analysis of instant messaging and VoIP applications. In: Garain, U., Shafait, F. (eds.) IWCF 2012/2014. LNCS, vol. 8915, pp. 188–199. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20125-2_16
20. Smith, C.: 34 interesting steam statistics and facts (2018). https://expandedramblings.com/index.php/steam-statistics/. Accessed 02 May 2018
21. Wireshark: About wireshark. https://www.wireshark.org/. Accessed 02 Feb 2018
22. Wong, K., Lai, A.C.T., Yeung, J.C.K., Lee, W.L., Chan, P.H.: Facebook Forensics, pp. 1–24. Valkyrie-X Security Research Group, Singapore (2013)