Forensic Artifacts of the flareGet Download Manager

Prachi Goel[1] and Babu M. Mehtre[2(✉)]

[1] School of Computer and Information Sciences, University of Hyderabad, Hyderabad, India
prachi_8dec@rediffmail.com

[2] Institute for Development and Research in Banking Technology (IDRBT) Established by Reserve Bank of India, Hyderabad, India
bmmehtre@idrbt.ac.in

Abstract. There is an increasing interest in finding artifacts (digital evidence) created by various software tools. flareGet is an advanced multi-threaded and multi-segment download manager for Linux. This is the only download manager for Linux that integrates with almost all the browsers. In this paper, we examine (from a digital forensics angle) the artifacts created by flareGet for Linux, specifically on Ubuntu 12.04 distribution. The flareGet artifacts include download path, URL address, settings of flareGet, date and time of the activity performed, the encryption technique used by flareGet, etc. This is useful for the digital forensic investigator to search and interpret the artifacts created or left in the process of using flareGet.

Keywords: Artifacts · Digital forensics · Investigation · flareGet

1 Introduction

There is an increasing interest in finding artifacts (digital evidence) created by various software tools. There are a number of software tools which have been examined for artifacts on Windows and Linux platform. Relatively, the number of such tools examined for artifacts on Windows platform is more than those on Linux platform. flareGet is a native Linux application written in C++, using the Qt framework. For installing flareGet, the system should meet the following minimum dependencies:

1. Qt libraries with version >=4.8.1
2. glibc (C library) with version >=2.13.

flareGet is a full featured, advanced, multi-threaded, multi-segment download manager and accelerator on Linux [1]. It supports all 32 and 64 bit ‘Debian’ and ‘Red Hat Package Manager’-based Linux distributions. flareGet is proprietary software however; it is also available freely with limited features. The following features are not present in the freely available flareGet version 1.4-7:

1. Up to 16 parallel connections per download.
2. Browser Integration with all the browsers.

© Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2014

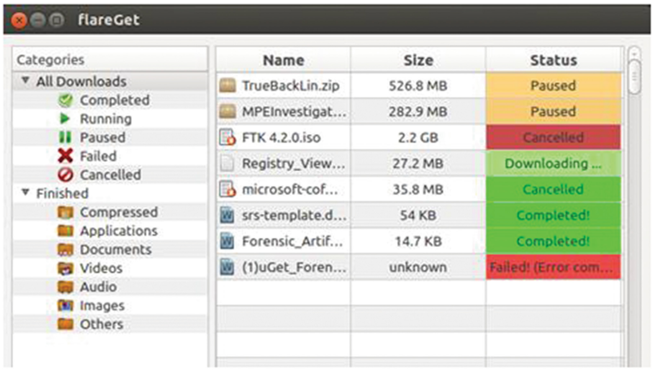
DOI: 10.1007/978-3-319-14289-0_3

3. Support for download speed limits.
4. Support for auto-refreshing of URL and cookies.

Freely available, flareGet can support up-to 8 parallel connections per download and provides the integration with Mozilla Firefox via FlashGot (a third party add-on). flareGet supports HTTP, HTTPS and FTP for downloading the files from Internet. It also supports Meta links. It uses a robust dynamic file segmentation algorithm which splits the download into segments to accelerate the process of downloading. In addition to dynamic file segmentation, it uses HTTP-pipelining in which multiple requests are sent on a single TCP connection without waiting for the corresponding responses. This further accelerates each segment up to six times. It uses intelligent file management to automatically categorize the downloaded files based on their extensions. The downloaded files are grouped into different folders as per their categories.

This paper focuses on the artifacts created in the process of using flareGet (freely available version 1.4-7) on Linux (Ubuntu 12.04 distribution). Even though the examination is done by using the free version of flareGet, the same type of artifacts are applicable to professional flareGet (because the features which are not available in the free version would not affect the artifacts created by flareGet). It is found that flareGet creates the artifacts in the location: '/home/< user >/.config' with different folder names. The important folders created by flareGet from the forensics point of view are discussed in the following sections. All the traces created by flareGet are traced by using Strace (a debugging utility for Linux).

The GUI of the flareGet download manager is shown in Fig. 1. The Left Panel of flareGet contains two main tabs: 'All Downloads' and 'Finished'. Under the 'All Downloads' tab there are various status tabs which include *Completed, Running, Paused, Failed* and *Cancelled*. Under the 'Finished' tab there are various categories tab which include *Compressed, Application, Documents, Videos, Audio, Images* and *Others*. The Right Panel displays the information of the corresponding tab on the left panel when clicked by the user. For example, in Fig. 1, the right panel shows the information of all the files including their states like *completed, running, paused, failed* and *cancelled*, shown when the 'All Downloads' tab on left panel gets clicked.



Name	Size	Status
TrueBackLin.zip	526.8 MB	Paused
MPEInvestigat...	282.9 MB	Paused
FTK 4.2.0.iso	2.2 GB	Cancelled
Registry_View...	27.2 MB	Downloading ...
microsoft-cof...	35.8 MB	Cancelled
srs-template.d...	54 KB	Completed!
Forensic_Artif...	14.7 KB	Completed!
(1)uGet_Foren...	unknown	Failed! (Error com...

Fig. 1. The GUI of flareGet.

This paper is organized into 5 sections. The first section gives a brief introduction of the flareGet. The second section discusses the related work done by others. The third section details the artifacts of flareGet in five sub-sections. Sections 3.1, 3.2 and 3.3 describe from where the forensic investigator can find the downloaded/downloading files, proxy settings and paused file information. Section 3.4 describes from where the forensic investigator can find the information about websites requiring authentication during the download process, and it also describes the encryption technique used by flareGet to store the user's passwords for these websites. Section 3.5 shows the installation and un-installation artifacts of the flareGet. The summary of flareGet artifacts is given in Sect. 4. Finally the paper concludes in Sect. 5.

2 Related Work

Numerous researchers have worked to find the artifacts left behind by different software applications from the digital forensics perspective. In the literature there are many papers for detecting software application artifacts on the Windows platform but less research has been done to find software application artifacts on the various Linux platforms. The Windows Registry provides essential information from forensics point of view [2]. Vivienne Mee et al. [3] examined the use of the Windows Registry as a source of forensic evidence in digital investigations, especially related to Internet usage.

Bit Torrent is a peer-to-peer file sharing protocol used for distributing large amounts of data. It has been seen that usage of Bit Torrent client application leaves the traces in the registry [4]. Increase of the Gnutella network (peer-peer network) usage lead researchers to find the artifacts left behind after the use of Limewire [5] and FrostWire [6] software tools.

Geoffrey Fellows [7] presented WinRAR temporary folder artifacts which provide the essential evidence to the investigator to prove which files were viewed or extracted using WinRAR program. Muhammad Yasin et al. [8, 9, 10] analyzed the 'Download Accelerator Plus', 'Free Download Manager' and 'Internet Download Manager' for collection of digital forensic artifacts.

Many Instant messenger software applications have been examined which provide exchange of text messages in real-time. These include Yahoo Messenger 7.0 [11], Trillian basic 3.x [12], MSN Messenger 7.5 [13], AOL Instant Messenger 5.5 [14], Windows Live Messenger 8.0 [15] and Pidgin Messenger 2.0 [16].

Steganography, whole disk encryption and private browsing are some of the challenging areas for forensics investigators. Rachel Zax et al. [17] presented the traces left behind after a number of freely available steganography tools were installed, run, and uninstalled. Sungsu Lim et al. [18] investigated the installation, runtime, and deletion behaviors of virtual disk encryption tools in a Windows XP SP3 environment. Huwida Said et al. [19] examined the artifacts left by conducting Web browsing privacy mode sessions in three widely used Web browsers (Firefox, Google Chrome and Internet Explorer), and analyzed the effectiveness of this tool in each Web browser.

3 flareGet Artifacts

3.1 Downloaded Files Information

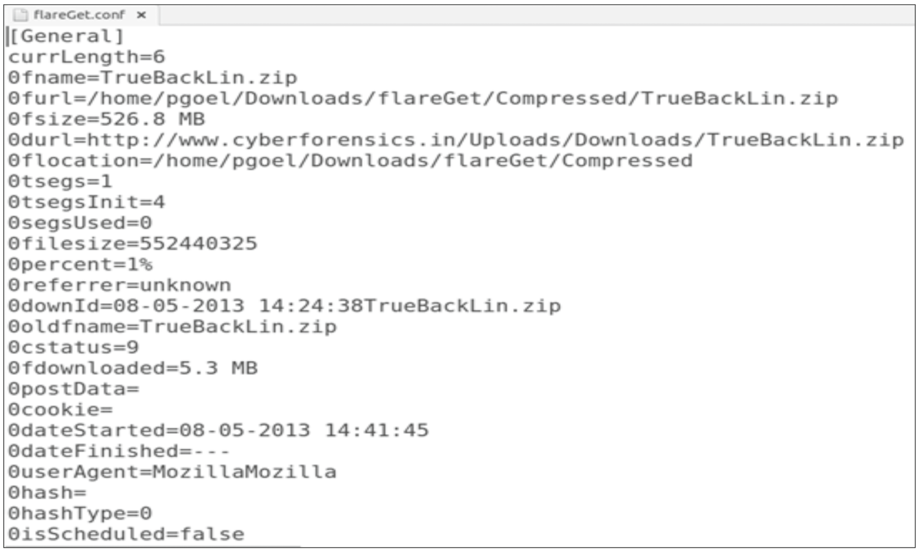
The important question during investigation of a download manger is what are the files that were downloaded or are still being downloaded by the user. There are two ways to find this information. The first way is to look at flareGet.conf file which is located in the directory '/home/<user>/.config/flareGet_ALLDOWN'. This file contains the entire downloaded and downloading file information. Figure 2 is showing the portion of flareGet.conf file. The *CurrLength* in this file indicates the total number of files which is the count of *downloaded*, *downloading*, *cancelled*, *failed*, *paused* and *queued*, provided the user has selected to remember the *finished*, *cancelled* and *failed* downloads in settings page of flareGet. If the user has not selected to remember the *finished*, *cancelled* and *failed* downloads then the *CurrLength* indicates the total number of files which are paused or queued. Even if the user has *unchecked* to remember the *finished*, *cancelled* and *failed* downloads then the investigator can also find this information in the file 'flareGet.conf', provided it is not overwritten by another file information because flareGet overwrites the whole flareGet.conf file at the time of exit. The same is applicable to the files which get deleted by the user from flareGet, i.e., the file information remains until overwritten by the flareGet. Each downloaded file information record starts with an integer. For example, in Fig. 2 the file having the name 'TrueBackLin.zip' indicated by 'fname' started with integer '0'. All the attributes of this file start with integer '0'. Similarly the next file attribute starts with integer '1' and so on. The important attributes from the forensic point of view are explained below:

furl: shows where the downloaded file is stored.
fsize: shows the size of the downloaded file in string format.
durl: shows the web address from where the file gets downloaded.
flocation: shows where the downloaded file gets stored.
fdownloaded: shows how much file gets downloaded.
filesize: shows the size of the downloaded file in bytes.
referrer: holds the website from which the download started (required for auto refreshing of URL).
hash: holds the hash value given by the user
hashType: whether the hash is MD5 or SHA1.
cstatus: an integer which takes different values for different states of the file which is shown in Table 1.
dateStarted: stores the date and time when the downloading process started in plain text.
dateFinished: stores the date and time when the downloading process gets finished in plain text. If the downloading process was not finished then it would not show any value.

The second way is to look at folder 'flareGet_FINDOWN' which is located at '/home/<user>/.config/'. This folder contains the file 'flareGet.conf' which stores all the file information whose downloading gets completed. This file contains the same attributes as explained for the file located at folder 'home/<user>/.config/flareGet_ALLDOWN'. Even if the user has unchecked to remember the finished downloads then also

Table 1. Different *cstatus* values for different *states* of file.

cstatus	States of the file
0	Downloading of file gets finished
2	File is downloading
3	Downloading file is in paused state
4	Downloading of file gets failed
5	Downloading file gets cancelled by the user
6	file is queued by the flareGet
8	file is queued by the user
9	Downloading of file gets failed and paused by the user
11	File downloaded using Metalink



```
[General]
currLength=6
0fname=TrueBackLin.zip
0furl=/home/pgoel/Downloads/flareGet/Compressed/TrueBackLin.zip
0fsize=526.8 MB
0durl=http://www.cyberforensics.in/Uploads/Downloads/TrueBackLin.zip
0flocation=/home/pgoel/Downloads/flareGet/Compressed
0tsegs=1
0tsegsInit=4
0segsUsed=0
0filesize=552440325
0percent=1%
0referrer=unknown
0downId=08-05-2013 14:24:38TrueBackLin.zip
0oldfname=TrueBackLin.zip
0cstatus=9
0fdownloaded=5.3 MB
0postData=
0cookie=
0dateStarted=08-05-2013 14:41:45
0dateFinished=---
0userAgent=MozillaMozilla
0hash=
0hashType=0
0isScheduled=false
```

Fig. 2. The portion of ‘flareGet.conf file’ located at ‘/home/<user>/.config/flareGet_ALLDOWN’.

the investigator can find this information in file ‘flareGet.conf’, provided it is not overwritten because flareGet overwrites the whole ‘flareGet.conf’ file at the time of exit.

3.2 Proxy Settings Information

The investigator can find the proxy setting by looking at the folder ‘flareGet Settings’ which is located at ‘/home/<user>/.config/’. If the manual proxy is set by the user then ‘proxy_addr’ and ‘port_num’ contain the address and port number respectively. The ‘username’ and ‘pwd’ holds the user name and password if required for the proxy setting respectively. The password is stored in plain text which accelerates the process of investigation. For manual proxy setting it provides 3 options:

1. HTTP/HTTPS
2. SOCKS v5
3. FTP.

The ‘proxyType’ can takes three values, i.e., 0, 1 and 2 for HTTP/HTTPS, SOCKS v5 and FTP respectively. Since this file contains the universal settings of flareGet, it can also give the following information:

1. Where are the files stored by default?
2. Which browsers are integrated with flareGet?
3. Scheduled activities which are configured by the user etc.

3.3 Information of Paused Files

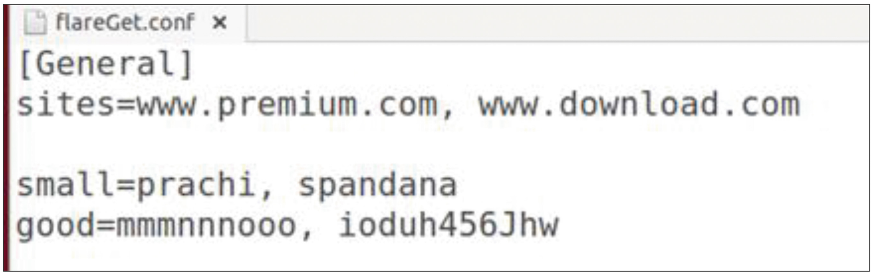
The paused file information is present in two folders. The investigator can look at the folder ‘flareGet_ALLDOWN’ (explained in Sect. 3.1) or ‘flareGet’. The ‘flareGet’ folder is located at ‘/home/<user>/.config/’. The file ‘datasegment.conf’ in this folder is essentially used to provide resume capabilities for file downloads if the download is paused by the user or the Internet connection is not available. flareGet also provides resume capabilities for downloading files after closing flareGet or shutting down the PC. The ‘datasegment.conf’ file keeps the name of the paused file with a record of how many bytes have been downloaded by each thread of the download. Records of threads of each download are also located at this location. If the paused file is subsequently downloaded or deleted by the user then the ‘datasegment.conf’ file would not contain any information about that file, and the thread records created by flareGet get deleted.

3.4 Information About Websites that Require Authentication During Download

There is a facility given by flareget to store the username and password for websites which require authentication during the download in the ‘site manager’ setting page. The website name with the corresponding username and password provided by the user in the ‘site manager’ setting page is stored in file ‘flareGet.conf’, which is located at ‘/home/<user>/.config/flareSBase’. All the passwords are stored in encrypted form. Figure 3 shows the content of this file. The fields ‘sites’, ‘small’ and ‘good’ store the website name, username and password respectively. The site names, usernames and passwords are separated by commas. For example, for the website ‘www.premium.com’, the user name is ‘prachi’ and password is ‘mmmnnnooo’, which is in encrypted form.

To find the encryption technique used by flareGet, the following experiment is performed by taking 3 types of password samples which are described below:

1. Sequence of repeated letters.
2. Combination of letters with numbers.
3. Alphanumeric characters.



```
[General]
sites=www.premium.com, www.download.com

small=prachi, spandana
good=mmmnnnooo, ioduh456Jhw
```

Fig. 3. The content of ‘flareGet.conf’ file located at ‘/home/<user>/.config/flareSBase’.

Table 2 shows the three different samples of passwords with the corresponding encrypted password stored by flareGet. It is clear from Table 2 that flareGet uses a simple additive cipher technique whose key is equal to 3. flareGet first converts the characters entered by the user as a password into ASCII code and then adds 3 to the corresponding ASCII code.

Table 2. Analysis of encrypted password.

Password in plain text	ASCII code	Password in encrypted form	ASCII code
jjjkkklll	106 106 106 107 107 107 108 108 108	mmmnnnooo	109 109 109 110 110 110 111 111 111
flare123Get	102 108 97 114 101 49 50 51 71 101 116	ioduh456Jhw	105 111 100 117 104 52 53 54 74 104 119
flare@!$Get*	102 108 97 114 101 64 33 36 71 101 116 42	ioduhC$’Jhw-	105 111 100 117 104 67 36 39 74 104 119 45

Since ASCII is defined for 128 (ASCII code from 0-127) characters, in boundary cases, the addition of ‘3’ to the ASCII code exceeds the 126th ASCII code (127 is reserved for DEL). So to handle these boundary cases, flareGet uses the additive modulo 94 (127 - 33). The first 32 ASCII characters are reserved for control characters and the 33rd ASCII character is for space. flareGet uses the additive modulo 94 if the resultant ASCII code of a character + 3 is greater than 126. Table 3 shows the cases where modulo 94 comes into the picture. It is clear from the Table 3 that flareGet uses the escape character ‘\’ for ‘"’.

3.5 Installation and Un-installation Artifacts

Ubuntu stores the installation and un-installation information in dpkg.log which is located in ‘/var/log/’. Un-installing flareGet does not remove any directory created by flareGet at location ‘/home/<user>/.config/’. This provides important evidence for the investigator even if the flareGet application is un-installed.

Table 3. Analysis of boundary cases.

Password in plain text	ASCII code	Password in encrypted form	ASCII code
pass|}rd	112 97 115 115 124 125 114 100	sdvv!\"ug	115 100 118 118 33 92 34 117 103
pass ~ ord	112 97 115 115 126 111 114 100	sdvv#rug	115 100 118 118 35 114 117 103
pass ~ }rd	112 97 115 115 126 125 114 100	sdvv#\"ug	115 100 118 118 35 92 34 117 103

4 flareGet Artifacts Summary

flareGet creates the artifacts at the location '/home/<user>/.config/' which is summarized as follows:

1. All the downloaded/downloading file information is stored in the folder 'flareGet_ALLDOWN'.
2. All the finished/cancelled file information is stored in the folder 'flareGet_FINDOWN'.
3. The universal setting information of flareGet is stored in the folder 'flareGet Settings'.
4. Data for resumption of paused files is stored in the folder 'flareGet'.
5. Data for websites that require passwords for authentication is stored in the folder 'flareSBase'.

Installation and un-installation artifacts of the flareGet application on Ubuntu are found in the folder '/var/log/' and the file name is dpkg.log.

5 Conclusion

All the folders created by flareGet are located in one single directory, i.e., '/home/<user>/.config/'. This helps the investigator to collect the evidence easily from a single location. The artifacts, like username and password for proxy settings, date and time, etc., are found in the plain text (not encrypted) which accelerates the process of investigation. Hence, this eases the task of the forensic investigator. flareGet uses encryption only for storing the password of websites which require authentication for downloading. The encryption technique is explained in Sect. 3.4 and is quite simple. Even after the un-installation of the flareGet application on Ubuntu, the directory '/home/<user>/.config/uGet' remains intact and contains valuable evidence for the investigator.

References

1. Flareget. http://flareget.com/
2. Carvey, H.: The windows registry as a forensic resource. Digit. Invest. **2**, 201–205 (2005)

3. Mee, V., Tryfonas, T., Sutherland, I.: The windows registry as a forensic artefact: illustrating evidence collection for internet usage. Digit. Invest. **3**, 166–173 (2006)
4. Lallie, H.S., Briggs, P.J.: Windows 7 registry forensic evidence created by three popular BitTorrent clients. Digit. Invest. **7**, 127–134 (2011)
5. Lewthwaite, J., Smith, V.: Limewire examinations. Digit. Invest. **5**, S96–S104 (2008)
6. Lewthwaite, J.: Frostwire P2P forensic examinations. Digit. Invest. **9**, 211–221 (2013)
7. Fellows, Geoffrey: WinRAR temporary folder artefacts. Digit. Invest. **7**, 9–13 (2010)
8. Yasin, M., Wahla, MA., Kausar, F.: Analysis of download accelerator plus (DAP) for forensic artefacts. In: 5th International Conference on IT Security Incident Management and IT Forensics, pp. 235–238. Fraunhofer Gesellschaft Institutszentrum Stuttgart, Germany (2009)
9. Yasin, M., Wahla, MA., Kausar, F.: Analysis of free download manager for forensic artefacts. In: First International ICST Conference on Digital Forensics and Cyber Crime, pp. 59–68. Albany, NY, USA (2009)
10. Yasin, M., Cheema, A.R., Kausar, F.: Analysis of internet download manager for collection of digital forensic artefacts. Digit. Invest. **7**, 90–94 (2010)
11. Dickson, M.: An examination into yahoo messenger 7.0 contact identification. Digit. Invest. **3**, 159–165 (2006)
12. Dickson, M.: An examination into trillian basic 3.x contact identification. Digit. Invest. **4**, 36–45 (2007)
13. Dickson, M.: An examination into MSN messenger 7.5 contact identification. Digit. Invest. **3**, 79–83 (2006)
14. Dickson, M.: An examination into AOL instant messenger 5.5 contact identification. Digit. Invest. **3**, 227–237 (2006)
15. van Wouter, S.: Dongen, forensic artefacts left by windows live messenger 8.0. Digit. Invest. **4**, 73–87 (2007)
16. van Wouter, S.: Dongen, forensic artefacts left by pidgin messenger 2.0. Digit. Invest. **4**, 138–145 (2007)
17. Zax, R., Adelstein, F.: FAUST: forensic artifacts of uninstalled steganography. Digit. Invest. **6**, 25–38 (2009)
18. Lim, S., Park, J., Lim, K., Lee, C., Sangjin, L.: Forensic artifacts left by virtual disk encryption tools. In: 3rd International Conference on Human-Centric Computing (HumanCom), pp. 1–6. Cebu, Philippines (2010)
19. Said, H., Al Mutawa, N., Al Awadhi, I., Guimaraes, M.: Forensic analysis of private browsing artifacts. In: International Conferences on Innovations in Information Technology, pp. 197–202. United Arab Emirates, Abu Dhabi (2011)