WhatsApp Web Client Live Forensics Technique

Alberto Magno Muniz Soares
Criminalistics Institute of the Civil Police, Federal District, Brasília, Brazil

Keywords: WhatsApp, Live Forensics, Web Browser, Data Extraction.

Abstract: At crime scenes or when participating in arrest warrants, forensic experts may come across situations where there is a WhatsApp Web service session available on an on-site computer, which can be a very important source of data for an investigation. Some justice systems may consider video recordings or printing of conversations screens from a computer illicit or questionable evidence. This article analyses WhatsApp Web in browsers, presents a live acquisition technique that allows automated extraction of messages, attachments, contacts, and account data, even if in a disconnected computer, from WhatsApp Web sessions opened in web browsers. The technique extracts, in line with forensics procedures, digital data that can be loaded in forensic tools for analysis.

1 INTRODUCTION

The use of messaging services has become a very important reality in the daily lives of many people, and very often-criminal activities leave traces in messages that circulate through these services. Evidence based on digital artefacts is increasingly present in any criminal investigation, and the concern with the proper handling of these artefacts has become the focus of many legal discussions. Messaging services have different security protocols, and the security level of these protocols is a strong factor considered when choosing a service by users. More and more messaging services offer strong security protocols, which makes it difficult for the criminal expert to extract data from these services. In addition, many of them offer web client services that work in parallel to mobile and desktop applications.

In January 2015, WhatsApp LLC introduced a web client for all major desktop browsers: Google Chrome, Mozilla Firefox, Opera, Microsoft Edge, and Safari (WhatsApp, 2015). To use the WhatsApp web client, a user navigates to https://web.whatsapp.com on supported browsers. Next, the user would scan a quick response (QR) code within the WhatsApp application on a smartphone to start sending and receiving messages. In September 2021, a beta release of WhatsApp web client started to offer a new multi-device feature. This release allows the user to chat on up to four devices, even when the phone disconnected or switched off.

Commonly, in places of enforcement of court decisions for apprehensions, law enforcement agents have contact with computers with open WhatsApp Web application sessions containing conversations with potential investigative interest. Frequently, they do not have access to the mobile device linked to these conversations, and thus, screen prints of conversations are a widely used resource of evidence. Some justice systems consider recording images of conversations from a computer screen as illicit or questionable evidence. As example, recently, in the Brazilian justice system, according to process number 2020/0217582-8, a judge of the Superior Court of Justice ruled that "…consider the messages obtained through the screen print of the WhatsApp Web tool as illicit evidence…" (STJ, 2021). This legal situation brings new challenges to computer forensics and demands the development of techniques to better assist to the justice system.

Live forensics consists of a digital analysis performed through forensic procedures and conducted on computational equipment that is still running (Hay, 2009). The procedure can be relatively complex and time-consuming, such as copying data from open web application sessions. The procedure has advantages and disadvantages compared to conventional digital expertise, also called post-mortem analysis, performed after shutdown the target system. A special advantage of live forensics is the capability of acquiring important volatile data, like malwares, cypher keys, passwords, active

Soares, A.
WhatsApp Web Client Live Forensics Technique.
DOI: 10.5220/0011006400003120
In *Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP 2022)*, pages 629-636
ISBN: 978-989-758-553-1; ISSN: 2184-4356

connections, or chat messages not stored in local persistent memory devices.

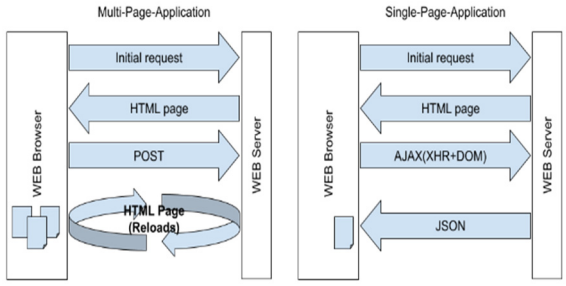

Figure 1: Basic difference between traditional Multi-Page-Application and Single-Page-Application. In SPA, transactions never require a browser reload. Source: Author.

A common technique in live-forensics is volatile memory acquisition using a tool capable of dump all RAM content, including virtual memory page files, for later RAM image analysis (Johansen, 2020). The image data analysis uses specialized tools capable of rebuild all internal memory structures of the target operational system. One powerful and frequently used tool is Volatility Framework (Ligh, 2014), but to analyse data of a web application inside a browser process, a big decoding effort would be necessary to be able to retrieve chat messages, and until the moment, a specific plugin or tool for that is unknown.

Other approach in live forensics is recover data directly operating commands or using live response tool. With this approach, no further decoding effort is necessary.

The current article presents a live response technique for direct acquiring forensic artefacts left behind the WhatsApp Web client. A technique based on browser JavaScript console capabilities that is available on most supported browsers. The first part of this article shows related works with WhatsApp Web forensics. In further sections, this work shows modern web application concepts and WhatsApp Web internal properties, and the live forensic technique, named ZAPiXWEB, to acquire data through web browser console.

2 RELATED WORKS

Most forensic work regarding WhatsApp Web client data extraction present post-mortem approaches.

The research in (Yudha et al, 2017) identifies the possibility to find browser artefacts and their location in post-mortem investigations, and presents a model that helps researchers to investigate WhatsApp Web in several browsers, but included no room for live-data forensics.

In (Vukadinović, 2019), the author presents a research on ways of obtaining artefacts of WhatsApp, including those maintained by browser with WhatsApp Web on which artefacts could be further recovered and identified from cache, history, logs, scripts files left behind. It also observes the lack of research in WhatsApp Web forensics and showed limitations of recovered data.

As discussed in (Paligu, 2020), it describes methods for obtaining digital traces based on artefacts stored in IndexedDB an emerging browser technology, however, among the traces, there is no recovery of full chats contents.

As a study of interception possibilities, the work in (Wijnberg, 2021) presents a forensic approach to create real-time insight in the WhatsApp communication based in on the wiretapping, decrypting WhatsApp databases, open source intelligence and WhatsApp Web communication analysis. The work mentions (Kloeze, 2017 apud Wijnberg, 2021) describing a way of automated acquisition of data using the JavaScript API from WhatsApp Web.

After careful research of publications, the existence of previous academic works was identified that mention the use of JavaScript to intervene in the functioning of WhatsApp, such as in actions of intercepting communications. Open source frameworks that make it possible to programmatically operate WhatsApp Web services, including through installed browser extensions, are also public. However, no forensic study or tool was found where the browser's JavaScript console could be used for a forensic data acquisition. Therefore, an efficient technique for this is important, especially for web messaging services.

3 SINGLE-PAGE-APPLICATION

A single-page application (SPA) is a web architecture that interacts with the user by dynamically rewriting the current web page with new data from the web server, instead of the web browser loading entire new pages (Figure 1). The browser retrieves all necessary HTML, JavaScript, and CSS code with a single page load, dynamically loads the appropriate resources, and adds to the page as necessary, usually in response to user actions (Scott, 2015). In SPA, there are important concepts such as asynchronous functions, promises, web workers, and modules.

Asynchronous functions allow writing promise-based code as if it were synchronous, but without blocking the main thread and promises simplify deferred and asynchronous computations and represent the operations not completed yet. Web workers make it possible to run a script operation in a background thread separate from the main execution thread of a web application. Moreover, modules provide abstractions and encapsulation boundaries, breaking programs up into discrete chunks of functionality.

3.1 Webpack

Multiple tools exist that support modules with asynchronous and modules functionalities, and Webpack is one of them (Zammetti, 2020). It is a static module bundler for Web applications capable of packing all resources and their dependencies into static assets.

This tool allows splitting code into various bundles, which can then be loaded on demand or in parallel. During the application development, each module is a file, but in a bundle, the module is a combination of a function that contains the code of that file, and an ID of this module. In addition, chunks aggregate these modules in files. Structurally, a chunk is an object list, in that each object has keys as module IDs, and values are the modules themselves. A chunk contains its own name and a small piece of code for registration. One can note that each generated chunk file has its names suffixed with a hash number for cache control.

In the generated Webpack code, there is the runtime code to support modularity and asynchronous loading of assets. It is possible to bundle code into two build modes: development and production. In production mode, Webpack runtime works with optimized assets to improve load time (Bouzid, 2020). One optimization applied is unnecessary characters removal from the source code, without changing its functionality (known as “minification”). This optimization affects directly the processed bundle codes readability.

4 WhatsApp WEB CLIENT

Through reverse engineering efforts, it is possible to identify that WhatsApp web client is a SPA and that its artefacts, loaded in the Browser, presents a structure of assets chunks built with Webpack tool (Figure 2).

Figure 2: Example of a WhatsApp web client loaded files. Source: Author.

The HTML code (index) in main page contains reference to a deferred loaded JavaScript file, prefixed with “runtime”, that loads a regular webpack chunk object named webpackChunkwhatsapp_web_client (in previous versions, it is named webpackChunkbuild).

In this the chunk object, it is possible to identify that the array structure with modules entries and their functions, and the override method “push” pointing into runtime code. Figure 3 shows an example of the chunck structure, including the ‘push’ function entry. Analysis of runtime code shows the possibility to add a module into the chunk array using this function as this function binds code to the runtime. Then, it is possible that someone craft a module, insert into chunk, and gain access to all internal modules references.

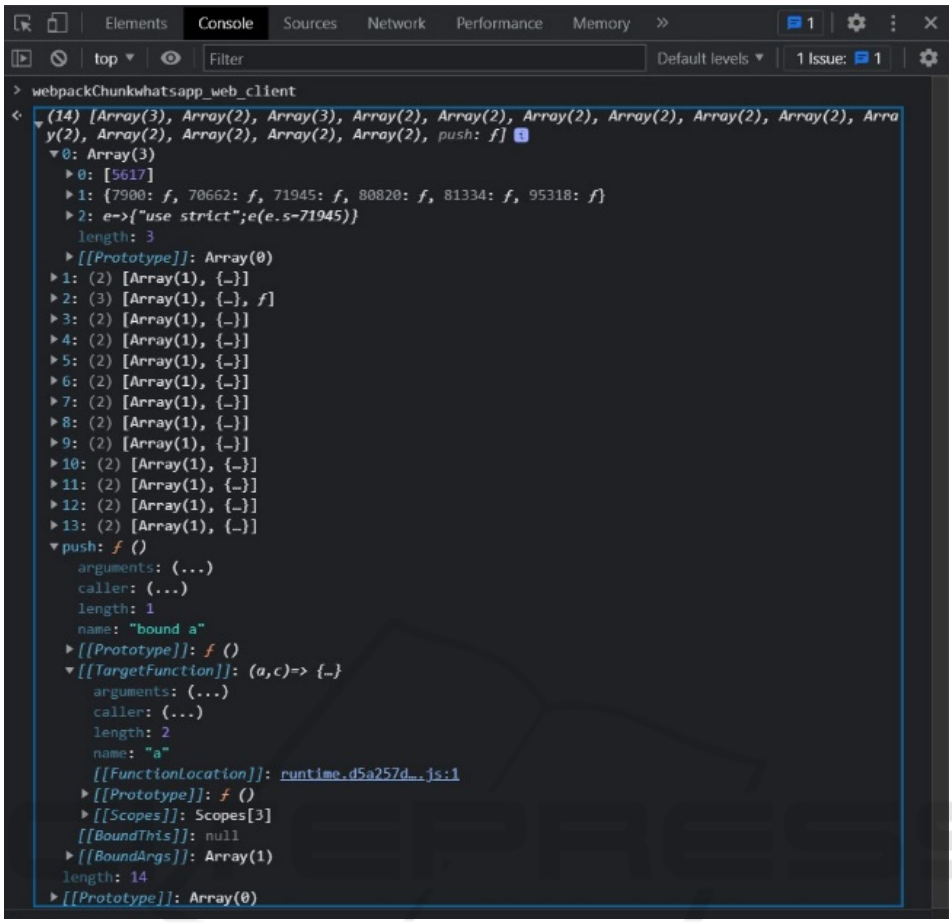

Figure 3: View of WhatsApp Webpack chunk showing details of 'push' function. Source: Author.

5 ACCESSING MODULES

Analysing the chunk object described in last section, it possible identify that it contains chunk with modules definitions, functions, constants in the application domain, as the name suggest.

The objects in chunk contains meaningless numeric IDs, but analysing its code, it is possible to identify that some elements that contain a property named as "default", and, inside it exists functions which their names give tips of the module responsibilities.

Then, a simple module code can be built to list all loaded elements into console, and pushed into chunk:

```
webpackChunkwhatsapp_web_client.push(
[["zapixModuleID"],    {},    function
(runtime, e, t) {
for (let chunkId in runtime.m) {
   chunk = runtime(chunkId);
   if(typeof chunk!=='undefined'){
    if(typeof
chunk.default!=='undefined'){
   console.log('ChunkId:'+chunkId);co
nsole.log(Object.keys(chunk.default))
;
    }
   }
}   }]);
```

Through browser's JavaScript console shell, it is possible to inspect objects elements and verify that the elements containing the internal object `Chat` and the element with the internal object `downloadAndDecrypt` have properties and methods with access to chat messages and media files. Then, it is possible to modify the built module code to

externalize these objects to the variables named `zapix_msgStore` and `zapix_downloadManager`:

```
var zapix_msgStore= null;
var zapix_downloadManager = null;
webpackChunkwhatsapp_web_client.push(
[["zapixModuleID"], {}, function
(runtime, e, t) {
    // o = runtime
    // m  = runtime webpack chunks
IDs
    for (let chunkId in runtime.m) {
     chunk = runtime(chunkId);
     if (typeof chunk!=='undefined'){
      if(typeof
chunk.default!=='undefined'){
       // Message Store
        if (typeof
chunk.default.Chat!=='undefined'){
console.log(chunk.default);
    zapix_msgStore = chunk.default;
        }
    // DownloadManager
     if (typeof
chunk.default.downloadAndDecrypt!=='u
ndefined')         {
     console.log(chunk.default);
     zapix_downloadManager =
chunk.default;
         }
    }
  } }   }]);
```

In this manner, it is possible to access chat messages in a WhatsApp session. For example, we can access all messages serialized in json from chatId 5561xxxx-xxxx@g.us, using the following command:

```
JSON.stringify(zapix_msgStore.Chat._i
ndex["5561xxxxx-
xxxx@g.us"].getAllMsgs())
```

Figure 4 shows a single chat (same in Figure 5), extracted using the sequence of codes describe above.

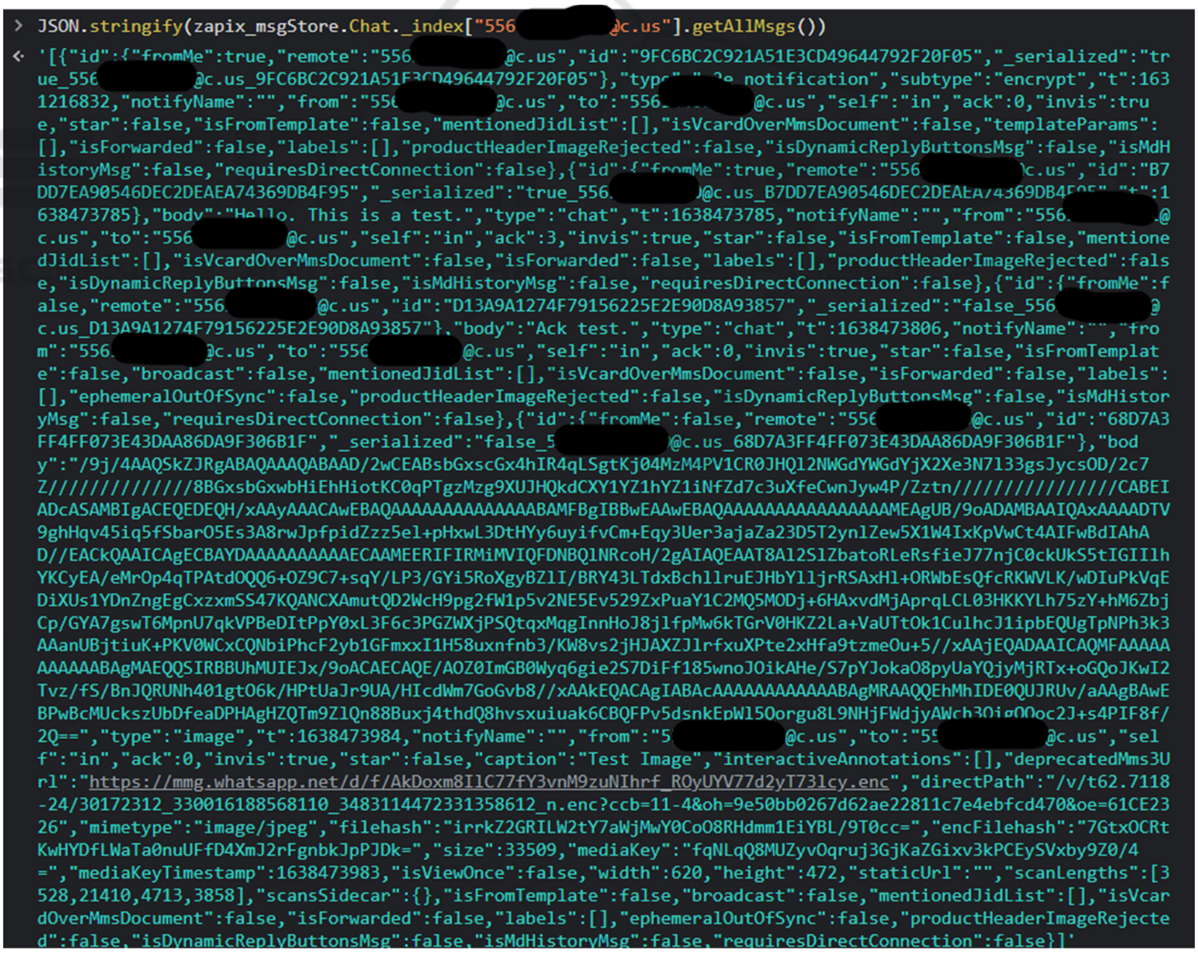

Figure 4: Example of simple chat extract. Source: Author.

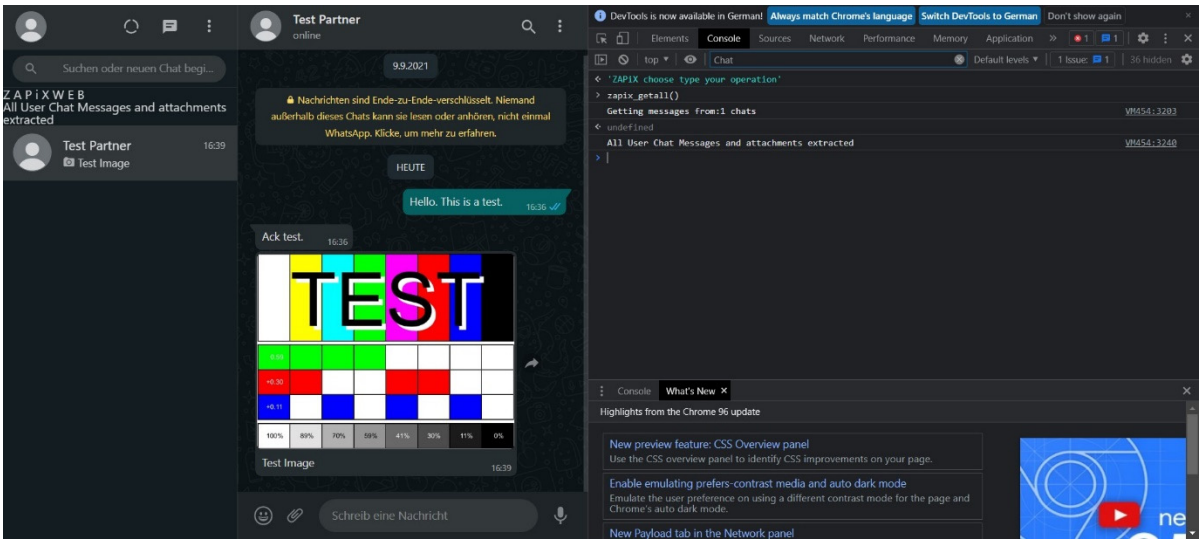

Figure 5: View of WhatsApp Web and ZAPiX commands execution on console. Source: Author.

6 ZAPiXWEB

The ZAPiXWEB is a WhatsApp Web live response technique based on the execution of script in browser's JavaScript console, in forensic acquisition phase, with injected code to expose Whatsapp Web internal modules built with Webpack framework.

The developed code was based on project Open WhatsApp (OpenWA, 2021), that provides an API to access the WhatsApp Web internal functions hooking handles to functions inside Webpack modules.

The code also uses library AXIOS HTTPClient (Axios, 2021) for better use of promise browsers functionality, JSZip tool (JSZip, 2021) to compact all data into a single ZIP file and FileSaver to save the final zip file on client-side.

The ZAPiX is a monolithic JavaScript-based file that must be executed in the command console of the web browser, accessible through development mode. For each browser, it is necessary to activate the developer mode and paste the script that will take action immediately after pressing "enter" key. The script is designed to work on all standard ECMAScript version 6 browsers such as the latest versions of Chrome, Firefox, Edge, Opera and Safari browsers. The script works in all browsers, except Safari, that need adaptations not implemented yet.

After execution, all basic WhatsApp user data is collected as associated phone number, contacts profiles, and sync tokens and mobile device data, if not using the WhatsApp Web beta version.

In standby state, the following commands available for extraction operations: `zapix_getall()`, for extraction all available chats; `zapix_getchat()`, for extracting a specific open conversation, and `zapix_takeout()`, to package what was extracted to ZIP file.

If the system is off-line, the forensic examiner can extract just chats available in memory, if online, it is possible to scroll each chat until the messages of interest, loading them in browser memory area. To grab all chat data in browser memory, the forensic examiner can call the `zapix_getall()`, as shown in Figure 4, and finally call `zapix_takeout()` to generate packed zip file. For just specific chats, it also possible interact with WhatsApp Web UI, opening each chat with mouse click and call `zapix_getchat()` in JavaScript browser's console. To finish, it's necessary to call `zapix_takeout()`, in which all data is packaged into a single zip file and open to save in any local storage device. Figure 6 shows an example of zip file containing data extracted.

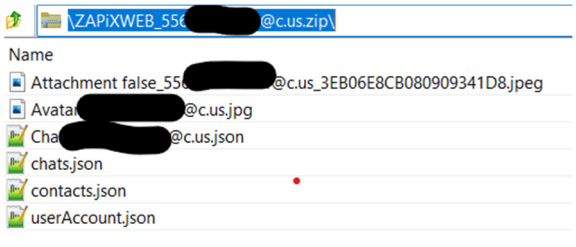

Figure 6: Example of a ZAPiX extract data into a zip package. Source: Author.

After data acquisition, the forensic examiner can calculate hash and load the data in a forensic tool for analysis and report. For use with Cellebrite UFED Physical Analyser, a specific plugin was developed that is capable to load all extracted data and load all data using the vendor API.

7 ASSUMPTIONS AND LIMITATIONS

As the presented technique is based on JavaScript to be executed in the Web Browsers console, it is necessary that the code be built in a portable way. For the present time, it has been adopting code compatible with ECMAScript version 6, compatible with the most popular Web Browsers.

Textual data is exported in JSON format and media data is decoded in the same formats used by WhatsApp Web (jpg, mp4, ogg, pdf, docx, vcard, webm), therefore, to analyze this data, a tool capable of decoding these data is needed.

Since the code was built for technique, changes have been detected in the code of the WhatsApp Web Webpack modules that required changes to the built code in order for it to remain operational. Thus, it is necessary to keep a constant review of the WhatsApp Web code to keep the constructed code working.

8 CONCLUSIONS

This paper presents a forensic technique for acquiring WhatsApp Web data from browser and making it ready to analysis phase. The work includes the study of concepts and structures of the most used browsers and architecture of single-page web applications. The proposed technique is performed using JavaScript code developed for the direct use in browser's JavaScript console and packing all extracted files into a single file.

The proposed technique contribution comes from its ability to extract and prepare the obtained data as JSON files to be useful in most of the forensic analysis tools, as Cellebrite Physical Analyser, thus overcoming earlier works and other traditional techniques based on post-mortem analysis or RAM dumps. An additional contribution concerns the console browser use as a forensic technique that can also be useful to other chat web applications.

As future work, the author intend to carry out of the technique with data retrieved from other web chat client applications, as Telegram Web (Telegram, 2021), that in preliminary analysis shows uses a similar architecture based in Webpack as WhatsApp Web.

ACKNOWLEDGEMENTS

The author is thankful to Police Agents and Forensic Experts members of SPI/SPCAT (IT and Advanced Computer Forensics sections) from Criminalistics Institute of the Civil Police in Federal District of Brazil (IC/PCDF) for their support in accomplishing this research.

REFERENCES

Axios (2021). Promise based HTTPClient for the web browser and nodeJS [Online]. Available: https://axios-http.com

Bouzid, M. (2020). Webpack for Beginners: Your Step-by-Step Guide to Learning Webpack 4. Apress.

Flanagan, David, "JavaScript - The Definitive Guide", 5th ed., O'Reilly, Sebastopol, CA, 2006, p.497 (6)

Hay, B., Nance, K., and Bishop, M. (2009). Live analysis: Progress and challenges. Digital forensics. IEEE Security and Privacy, 7:30–7. https://doi.org/10.1109/MSP.2009.43 (4)

Johansen, G. (2020). Digital forensics and incident response: Incident response techniques and procedures to respond to modern cyber threats. Packt Publishing Ltd.

JSZip (2021). JavaScript library for creating, reading and editing zip files [Online]. Available: https://stuk.github.io/jszip

Kloeze L. (2017), "Collecting huge amounts of data with WhatsApp," [Online]. Available: https://www.lorankloeze.nl/2017/05/07/collecting-hugeamounts-of-data-with-whatsapp/.

Ligh, M. H., Case, A., Levy, J., Walters, A.(2014). The art of memory forensics: detecting malware and threats in windows, linux, and mac memory. John Wiley & Sons.

OpenWA (2021). WA-Automated-Node [Online]. Available: https://docs.openwa.dev

Paligu, F., & Varol, C. (2020). Browser Forensic Investigations of WhatsApp Web Utilizing IndexedDB Persistent Storage. Future Internet, 12(11), 184.

STJ Revista Eletrônica (2021). Superior Tribunal de Justiça [Online on Superior Court of Justice website]. Available: https://processo.stj.jus.br/processo/revista/documento/mediado/?componente=ATC&sequencial=127283032&num_registro=202002175828&data=20210607&tipo=5&formato=PDF

Scott Jr, E. A. (2015). SPA Design and Architecture: Understanding single-page web applications. Simon and Schuster.

Telegram (2021). Telegram Web client [Online]. Available: https://web.telegram.org

Volatile Systems website. [Online]. Available: https://www.volatilesystems.com/default/volatility#overview

Vukadinovic, N. V. (2019). WhatsApp Forensics: Locating Artifacts in Web and Desktop Clients (Doctoral dissertation, Purdue University Graduate School).

WhatsApp Web - WhatsApp Blog. (2015, January 21). WhatsApp Web [Online]. Available: https://blog.whatsapp.com/whats-app-web

Wijnberg, D., & Le-Khac, N. A. (2021). Identifying interception possibilities for WhatsApp communication. Forensic Science International: Digital Investigation, 38, 301132.

Yudha, F., Luthfi, A., & Prayudi, Y. (2017). A proposed model for investigating on web WhatsApp application. Advanced Science Letters, 23(5), 4050-4054.

Zammetti, F. (2020). Modern Full-Stack Development: Using TypeScript, React, Node. js, Webpack, and Docker. Apress.