

WARNE: A Stalkerware Evidence Collection Tool

Philippe Mangeard

A THESIS

IN

THE DEPARTMENT OF

CONCORDIA INSTITUTE FOR INFORMATION SYSTEMS ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF APPLIED SCIENCE

IN INFORMATION SYSTEMS SECURITY

AT CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

AUGUST 2024

© Philippe Mangeard, 2024

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Philippe Mangeard

Entitled: WARNE: A Stalkerware Evidence Collection Tool

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science in Information Systems Security

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Abdessamad Ben Hamza

_____ Examiner
Dr. Abdessamad Ben Hamza

_____ Examiner
Dr. Anjan Bohwmick

_____ Thesis Supervisor(s)
Dr. Mohammad Mannan

_____ Thesis Supervisor(s)
Dr. Amr Youssef

Approved by _____
Dr. Chun Wang Chair of Department or Graduate Program Director

Dr. Mourad Debbabi Dean of Gina Cody School of Engine

Abstract

WARNE: A Stalkerware Evidence Collection Tool

Philippe Mangeard

Intimate partner violence (IPV) is a form of abuse in romantic relationships, more frequently, against the female partner. IPV can vary in severity and frequency, ranging from emotional abuse or stalking to recurring and severe violent episodes over a long period. Easy access to Stalkerware apps helps foster such behaviors by allowing non-tech-savvy individuals to spy on their victims. These apps offer features for discreetly monitoring and remotely controlling compromised mobile devices, thereby infringing the victim's privacy and the security of their data. In this work, we investigate methods for gathering evidence about an abuser and the Stalkerware they employ on a victim's device. We develop a semi-automated tool intended for use by investigators, helping them to analyze Android phones for potential threats in cases of IPV Stalkerware. As a first step towards this goal, we perform an experimental privacy and security study to investigate currently available Stalkerware apps. We specifically study the vectors through which vulnerabilities found in Stalkerware apps could be exploited by investigators. We then design and implement a tool called *WARNE*, leveraging the identified flaws to facilitate the information and evidence collection process. Our tool generates reports containing all available information about potential Stalkerware present on a victim's device. To ensure a better usability of *WARNE*, we also come up with report parsing mechanisms involving generative AI, especially Large Language Models (LLMs). This genre of Natural Language Processing models is suited for large text analysis and information extraction.

Acknowledgments

I would like to deeply thank my supervisors, Dr. Mohammad Mannan and Dr. Amr Youssef for their continued guidance, patience and support throughout this project. Their motivation, sapience and enthusiasm were of incredible help and I am proud to have been able to work under their supervision. This research was continuously improved by their ever so insightful comments, ideas and suggestions which gave me all the tools needed to bring this project to life.

I am also thankful to all my friends and peers at the Madiba Security Research Group who have shared this journey with me, and fueled me with their experience, knowledge and joy. Thank you to Sajjad Pourali, Bhaskar Tejaswi, Rohan Pagey, Abdelrahman Ragab and Xiufen Yu for everything you have taught me and your precious help during hard times. Concordia University is a wonderful learning environment and I feel lucky to have been a part of this research group.

Finally, I would like to thank my family, who supported me at all times even an ocean apart and my friends, without whom nothing would be possible. Thank you for always reminding me of my worth.

Contents

List of Figures	viii
List of Tables	ix
List of Acronyms	x
1 Introduction	1
1.1 Overview	1
1.2 Problem Statement	2
1.3 Contributions	3
1.4 List of Publications	4
1.5 Thesis Organization	5
2 Background and Related Work	6
2.1 Definition of stalkerware	6
2.2 Related Work	7
3 Methodology	11
3.1 Stalkerware Analysis	11
3.1.1 Ethical Consideration and Responsible Disclosure	11
3.1.2 Experimental setup	12
3.1.3 Stalkerware Vulnerabilities	13

3.2	WARNE Workflow	19
3.3	WARNE - Implementation	22
3.3.1	Prerequisites and Additional Tools	22
3.3.2	Implementation of WARNE Features	23
3.4	LLM Experimental Setup	25
3.4.1	ChatGPT	25
3.4.2	Llama3	26
4	Stalkerware Analysis Results	28
4.1	Stalkerware Apps Capabilities	28
4.2	Security Vulnerabilities	29
5	WARNE Test Results	36
5.1	WARNE Tests on Stalkerware Apps	36
5.2	WARNE Report Parsing	38
5.2.1	ChatGPT	38
5.2.2	Llama3	39
6	Conclusion and Future Work	43
6.1	Key Takeaways	43
6.2	Limitations	44
6.3	Recommendations for Potential Stalkerware Victims	45
6.4	Future Work	47
	Bibliography	48
	Appendix A	54
A.1	ChatGPT report analysis	54
A.1.1	OwnSpy	54

A.1.2	TheTruthSpy	64
Appendix B		74
B.1	Llama3 report analysis	74
B.1.1	OwnSpy	74
B.1.2	TheTruthSpy	77

List of Figures

Figure 3.1	Overview of the stalkerware security analysis methodology	13
Figure 3.2	WARNE’s evidence collection process	20
Figure 3.3	Initial prompt given to both models as a first set of instruction, along with related WARNE generated report. We put additional comments in blue text, which are not part of the prompt.	27

List of Tables

Table 4.1	Features available on tested stalkerware apps.	34
Table 4.2	Vulnerabilities identified within each tested apps.	35
Table 5.1	WARNE-parsed evidence and other relevant values found in stalker- ware apps.	41
Table 5.2	Results comparison of WARNE reports manual analysis and LLM analysis.	42

List of Acronyms

AAPT	Android Asset Packaging Tool
ADB	Android Debug Bridge
APK	Android Package Kit
CSRF	Cross-Site Request Forgery
GUI	Graphical User Interface
IPS	Intimate Partner Stalking
IPV	Intimate Partner Violence
ISDI	IPV Spyware Discovery
JWT	JSON Web Token
LLM	Large Language Model
SSHFS	Secure SHell File System
UUID	Universally Unique Identifier
XML	eXtensible Markup Language
XSS	Cross-Site Scripting

Chapter 1

Introduction

1.1 Overview

Intimate Partner Violence (IPV) has been recognized by the World Health Organization as a major global public health concern, affecting people of all genders, ages, and backgrounds and causing long-term health, social, and economic consequences [13]. IPV can take various forms, from physical violence to psychological abuse. Stalking is one form of harassment involving privacy invasion and remote monitoring, which has been greatly facilitated by programs called *stalkerware* (a.k.a. *spouseware* or *creepware*). Stalkerware apps are generally mobile applications enabling undetected remote control and activity monitoring of the compromised victim's phone by the abuser.

Previous studies have demonstrated the large size of the stalkerware landscape on mobile platforms, with hundreds of dual-use apps available on the Google Play Store [1] and dozens of companies distributing stalkerware apps outside the official Android marketplace [24]. Recent events like the COVID-19 lock-downs increased the use of stalkerware apps (by 93% according to Avast [2] in 2021). According to Fascendini and Fialová [12], technology-enabled violence can be defined by the anonymity of the abuser, the possibility

to perform action remotely and automatically, the propagation and persistence of compromised data as well as the affordable nature of the abusive tools. ESET [22] analyzed 86 stalkerware applications and reported over 18 critical vulnerabilities that could allow an attacker to perform actions such as remotely controlling the victim’s device, hijacking an abuser’s account, capturing victim’s data, and uploading forged data on behalf of the victim. Liu et al. [24] analyzed 14 such Android apps and identified the mechanisms used for spying and hiding from detection, as well as the security failings of these apps (e.g., not using HTTPS).

On the other hand, various online services exist to help victims and raise awareness [35]; see also stopStalkerware.org. Other apps [18] and open source projects [10] aim at mitigating the direct threat of stalkerware apps by detecting or uninstalling known stalkerware from mobile devices.

In short, existing IPV work mainly focuses on detecting and removing suspicious apps and analyzing their security weaknesses. However, to the best of our knowledge, no technical tools exist to help the victim (or an *investigator* who is helping the victim) to gather information, which could potentially be used as evidence against the abuser.

1.2 Problem Statement

To better understand the impact of stalkerware apps and find potential counter-measures, we investigate the capabilities and vulnerabilities of Android applications from outside the Google Play Store that are used by abusers for stalking purposes. With the knowledge gained from this study, we design and implement WARNE, a tool to help victims and investigators detect, remove and collect information/evidence about these harmful apps and their stalker. This requires:

- Designing and implementing a measurement pipeline to find and analyze technological tools (web/app stores) that can potentially be used by IPV abusers.
- Assessing the state of the stalkerware ecosystem, their capabilities and limitations.
- Identifying attack vectors on stalkerware apps that could allow a victim or an investigator to gather information or mitigate the effects of the stalkerware.
- Helping victims of stalkerware apps to gather evidence against their abusers, which can subsequently be used as proof of the abuse.
- Facilitating the evidence collection process by leveraging Large Language Models' capabilities to parse large amounts of data.

1.3 Contributions

We design and implement `WARNE`¹ to collect possibly incriminating and/or identifying information about an IPV abuser by leveraging a set of common security weaknesses in stalkerware apps, their back ends, and the stored data on the victim device. First, we collect 80 well-known, easy-to-find stalkerware apps from the web, and identify 50 unique APKs. Among these 50 apps, 30 are free or offer a free trial. Then, we analyze these free apps and the interfaces they provide to an abuser (e.g., web dashboard) against five common vulnerability types (as identified in past work, e.g., ESET [22]). Through a vulnerability analysis of stalkerware apps, we explore ways in which such security flaws give away information about the stalkerware and the abuser. Our tool `WARNE` leverages these flaws to collect evidence from the victim's device (e.g., locally stored app data from the stalkerware app). In this process, the tool searches for identifying information about the abuser (e.g., account information and email addresses) in the collected data. It provides an

¹The tool is named after Kate Warne (1833–1868), best known as the first female detective in the US (https://en.wikipedia.org/wiki/Kate_Warne).

easy-to-use graphical user interface (GUI) that allows the victim/investigator to navigate through readable files and databases found during the analysis. By default, WARNE uses a list of common stalkerware apps' package names for automated analysis. However, any suspicious app from the device can be selected for analysis. We evaluate WARNE against 30 stalkerware apps, and for 20 of them, the tool is able to gather useful information about the abuser without leaving a trace on the victim's phone.

Data compiled by WARNE is stored in text reports containing the app's metadata as well as a dump of readable text files and databases found in the app's local storage. To improve the investigator's efficiency in reading, analyzing and extracting relevant information from such reports, it is important to make it easily readable and understandable to the user. We put several techniques in place to facilitate this process, such as keyword matching, a graphical in-browser interface for manual inspection and the use of Generative AI, especially Large Language Models like ChatGPT or Llama3 to parse and highlight relevant elements from the report.

1.4 List of Publications

During my master's program, the following papers were published as results from our research work. This thesis presents work from the second publication.

- Mangeard, P., Yu, X., Mannan, M., & Youssef, A. (2023, November). No Place to Hide: Privacy Exposure in Anti-stalkerware Apps and Support Websites. In Nordic Conference on Secure IT Systems (pp. 18-36). Cham: Springer Nature Switzerland.
- Mangeard, P., Tejaswi, B., Mannan, M., & Youssef, A. (2024). WARNE: A stalkerware evidence collection tool. *Forensic Science International: Digital Investigation*, 48, 301677.

1.5 Thesis Organization

This thesis is structured as follows: Chapter 2 introduces contextual background for the research, defining what a stalkerware is and presenting related work in this field. In Chapter 3, we describe the methodology and the structure of our stalkerware analysis framework, WARNE testing process and the setup of our Large Language Model (LLM) report parsing experiment. We present in Chapter 4 our results for stalkerware capability and vulnerability analysis and in Chapter 5 the results of WARNE’s evidence collection tests as well as Large Language Models report processing. Finally, in Chapter 6, we give a set of recommendations to stalkerware victims and add concluding comments about our findings.

Chapter 2

Background and Related Work

This chapter introduces the context of our research. We first give a detailed explanation of what a stalkerware is and how we define it in the scope of our project. We then look at related work that has been done in the field of stalkerware analysis.

2.1 Definition of stalkerware

A stalkerware is a malicious software whose main goal is to monitor and track a victim's online and offline activity. They come in various formats but are most present in the mobile devices ecosystem, like Android or iOS. This harmful technology has become more concerning in the recent years, threatening the security and privacy of millions of victims by facilitating stalking and Intimate Partner Violence (IPV). One could argue that stalkerware and Spyware are the same, however, mechanisms used to detect and deal with the presence of such apps can greatly vary.

The difference between a stalkerware and a Spyware mostly resides in how the malicious app is installed in the first place, as well as the context which it is used in (spyware spread mostly through seemingly legitimate apps that the victim installs by themselves, stalkerware are also more specific to intimate partner contexts). stalkerware apps could

also be considered a subset of Spyware apps, as both share similar mechanisms but can have different purposes. As for Remote Access Trojans, although their main purpose is taking control of a device, they do not necessarily include 'passive spying' features shared by stalkerware and Spyware. As of now, the threat list that we use only allows for known stalkerware to be flagged but other detection mechanisms could be added in the future.

2.2 Related Work

Over the past years, several studies have been conducted on the stalkerware industry [42, 17], revealing the expanding landscape of spyware apps and keeping track of emerging actors in the field [9, 16, 7, 33, 40]. Chatterjee et al. [3] provided one of the first significant studies of the Intimate Partner Stalking (IPS) spyware ecosystem where they identified several hundreds of such IPS-relevant apps. While they found dozens of overt spyware tools, the majority are “dual-use” apps, i.e., apps that have a legitimate purpose (e.g., child safety or anti-theft), but can be easily and effectively repurposed to spy on a partner. They also show how some dual-use app developers are encouraging their use of IPS via advertisements, blogs, and customer support services. The authors analyze existing anti-virus and anti-spyware tools, which mostly fail to identify dual-use apps as a threat. Given the increasing exposure of intimate partner violence, which is further exacerbated by the growing online presence of various actors in the field (such as the distribution of stalkerware apps and the availability of more help services on the internet) [3, 5, 11, 32, 36, 38, 26, 28, 18], there is a growing need for a more in-depth analysis of the mechanisms employed by these new tools.

Freed et al. [14] provide a qualitative study that focuses on how IPV abusers exploit technology to intimidate, monitor, impersonate, and harass their victims. The authors argue that many forms of IPV are technologically unsophisticated from the perspective of IT/security experts. For example, these attacks are often carried out by a user interface-bounded

adversary, i.e., an authenticated adversarial user who can interact with the victim’s device or account via standard user interfaces, or by installing a readily available application that enables remote spying on the victim. Still, such attacks not only harm the victims but are also difficult to counteract because they undermine the dominant threat models considered during the design stage of most systems (e.g., attackers not having physical device access). Thomas et al. [37] argue that security, privacy, and anti-abuse protections are failing to address the widespread threat of online harassment.

Our work relates more with the studies of stalkerware apps’ technical capabilities [7, 8, 23, 31]. These studies considered one or two specific apps and provided insight regarding the poor security state of these apps, highlighting flaws such as inconsistent encryption usage or hard-coded secrets. More recently, Liu et al. [24] investigated the available features of 14 leading Android spyware apps and provided details about their mechanisms.

Security vulnerabilities in stalkerware systems are abundant, with apps like mspy [21], TheTruthSpy [39], Cerberus [30], spyHuman [6] and LetMeSpy [41],¹ leaking data of hundreds of thousands of users through data breaches. Unprotected databases is just one of many other flaws that are found on stalkerware apps and can be potentially exploited. ESET [22] manually analyzed 86 stalkerware apps and reported over 18 critical vulnerabilities that let an attacker perform actions such as remotely controlling the victim’s device, hijacking an abuser’s account, capturing victim’s data or uploading forged data on behalf of the victim. They reported a substantial growth of stalkerware usage in 2020, which correlates with an increase in IPV reports during the COVID-19 pandemic [26].

Regarding stalkerware apps detection and mitigation, notable studies include comprehensive records of known stalkerware apps [10, 22], often used as a baseline for spyware detection tools. Similar to traditional anti-viruses, Android spyware detectors mostly work via package name analysis, therefore requiring thorough and up-to-date spyware package

¹LetMeSpy’s website has been taken down after our first tests and is now unavailable.

databases. Havron et al. [18] developed a tool called IPV Spyware Discovery (ISDI) to detect spyware on Android and iOS devices. ISDI can help remove tracking apps from the phone but only provides limited information such as static app package data and the phone's resources information. New detection techniques such as those using activity analysis with machine learning [29] and traffic examination through external hardware [20] are also emerging.

An analysis of the stalkerware monetization ecosystem has been conducted by Gibson et al. [15] on over 6000 Android apps, sampled from the stalkerware Threat List in 2021. They mainly looked for keywords in the apps' code and evaluated the presence of payment/advertisement libraries used by the stalkerware apps.

A case study by Stephenson et al. [34] assessed the legal implications of Technology-Facilitated Abuse evidences. Through interviews and focus groups, they established that the most commonly used evidences of stalking were screenshots and testimonies, but could also include cell data and account logs in some cases. They also state that collecting evidence is a challenge for most victims, as data can disappear with time or the abuser can access to the victim's to delete the evidence themselves. They also mention that forensic services can be used for more sophisticated evidence collection, but often require access to "scarce resources like an attorney or police services".

We prioritize evidence collection through the security analysis of stalkerware apps and websites. We specifically focus on vectors that could be leveraged to get information about the abuser or the stalkerware distributing company.

Large Language Models have been used a lot for log parsing, Jiang et al. have published a framework to efficiently process chat logs [19]. Michelet et al. also showed that models such as ChatGPT could be used to write forensics investigations reports [25], confirming that they could be of use in a context of evidence collection. In our project, we aim to make the model filter out information that could be relevant to the investigator, rather than

asking it to summarize what has been found. This makes our goal somewhat close to code analysis by LLMs, as researched by Nam et al. [27]. Even though the tool they present is mostly prompt-less, understanding chunks of code and inferring meaning to them is similar to understanding .xml configuration files and SQLite databases.

Our work aims at adding a legal, evidence collection dimension to previously done work. In the context of IPV, victims often know who their abusers are but lack means of proving their wrongdoings. Providing sufficiently convincing evidence that abuse is taking place could help victims take legal actions and see them through.

Chapter 3

Methodology

In this chapter, we present the analysis pipeline we put in place in order to analyze stalkerware apps, identify vulnerabilities and leverage them through WARNE to collect potential evidence about the abuser. We first introduce our experimental setup for stalkerware analysis and describe the security vulnerabilities we primarily look for. We then detail the functioning of WARNE, its implementation and features. Lastly, we explain how our LLM report parsing tests are conducted on the output generated by WARNE.

3.1 Stalkerware Analysis

In this section, we first describe the analysis setup that we used during our testing. Then, we provide a detailed methodology to detect five commonly found vulnerabilities in stalkerwares. Two of these vulnerabilities (i.e., cross-site scripting, and insecure local data storage) are later leveraged in our proposed tool for evidence collection.

3.1.1 Ethical Consideration and Responsible Disclosure

Following the guidelines from Concordia University’s research ethics unit, all testing performed on stalkerware apps is done with our own (test) accounts and with a dedicated

phone as the victim's device. We do not record or save any information from the online dashboard other than our own. The designed tool runs locally on the investigator's machine and no information is shared with any third party. Regarding the ethical aspects of the tool's usage and the admissibility of the evidence gathered in their respective jurisprudence, an investigator is advised to seek legal help. The tool's Graphical User Interface (GUI) displays an explicit warning indicating the same. The tool has been uploaded to a private GitHub repository and will be made available to researchers and IPV clinics upon request. A public repository (with some artifacts) is also available here: <https://github.com/PhilippeMangeard/WarnePublic>.

3.1.2 Experimental setup

We use a Google Pixel 3 phone running Android 12, and a Genymotion virtual device running Android 10, with Google API installed. The Google Pixel 3 phone is rooted to allow superuser rights in the Android Debug Bridge (ADB) shell and certificate pinning bypass for our testing. Genymotion virtual devices are rooted by default. Our setup consists of an analysis device (workstation/laptop) to which the victim's device is connected via ADB (requires enabling developer options on the phone). Whenever possible, we download the stalkerware Android Package Kit (APK) directly on the device; otherwise, we install it from the computer using ADB. In some cases, the website's download link pointed to an installer that must be run first. In this situation, we use ADB to get access to the app's local files once fully installed and pull its APK file from the phone through a superuser shell. Fig. 3.1 illustrates our analysis setup and shows the different steps composing our approach.

We perform dynamic analysis with the help of Frida¹, a dynamic code instrumentation

¹<https://frida.re/>

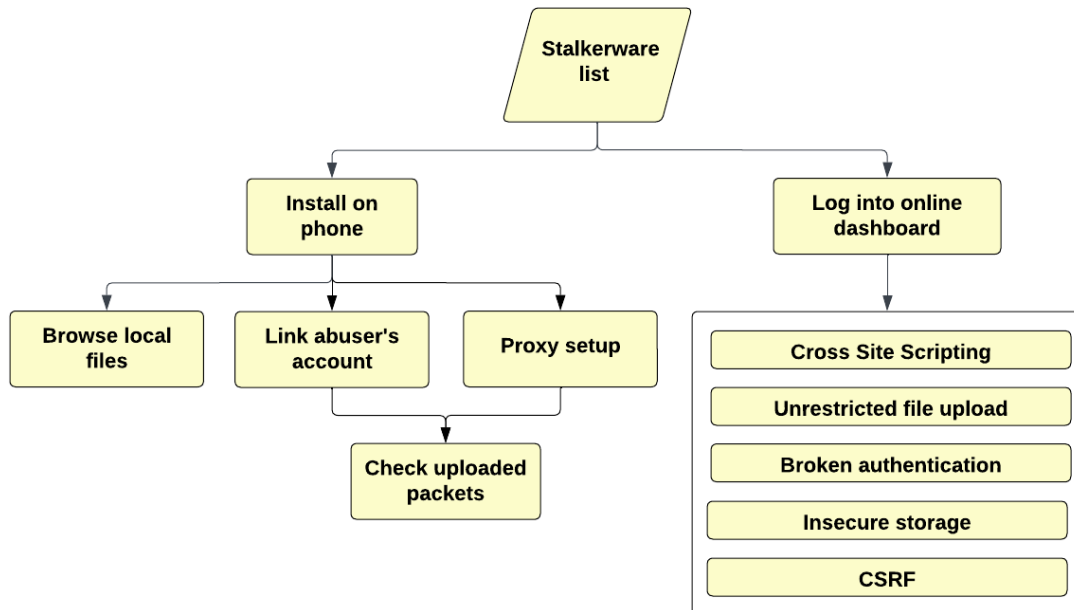


Figure 3.1: Overview of the stalkerware security analysis methodology

toolkit, allowing us, among other features, to inject JavaScript code into apps during runtime. With Frida, we can hook function calls and inspect parameters dynamically. Along with Frida, we use Burpsuite², an integrated platform offering multiple tools for performing security testing of web applications. The tools provided by Burpsuite include a proxy, a repeater and a decoder. With them we intercept HTTPS communications between the compromised mobile device and the stalkerware’s backend servers.

3.1.3 Stalkerware Vulnerabilities

During our study, we identified and tested five main vulnerability types in the stalkerware app ecosystem, which we either found during our tests or were reported in prior work (e.g., [22]). We discuss how each of these vulnerabilities could give away information about the abuser. This ecosystem includes the stalkerware’s mobile app, its backend server, and the web dashboard used to monitor the victim and browse collected data.

²<https://portswigger.net/burp>

Cross-Site Scripting (XSS).

After identifying what data the stalkerware gathers from the victim's phone, we check if the web dashboards of the stalkerware apps lack user-input sanitization, which could lead to XSS vulnerabilities.³ XSS is possible when uploaded information like contact names, text messages, calendar data, or any other user-input field is not verified by the Android app or the backend system. This allows unrestricted usage of special characters in strings, which when displayed on the abuser's dashboard, can trick the abuser's browser into interpreting the unrestricted input as code. XSS would allow an investigator to inject JavaScript code into the web application through the victim's device. We first compile a list of easy-to-edit inputs in the victim's device that are being reflected on the abuser's dashboard (the most common ones are contacts and text messages). We then use XSS fuzzing payloads from a self-hosted instance of XSShunter-express⁴ and manually inject them into our identified inputs. We add a new contact in the victim device's phone book and provide the XSS payload in the contact's name. We also send text messages containing the payload to and from the victim's phone. When the payload is executed on the abuser's machine, the investigator can collect information about the abuser's dashboard, or the machine on which they are logged in.

Note that we need to inject the payloads into the victim's device only once. As soon as the stalkerware app uploads it to the backend server, the custom string will be displayed on the dashboard (when the abuser browses the corresponding page), even if the XSS payload is deleted from the phone afterward. In this case, each time the abuser opens a web page containing the malicious string, the injected script will be executed.

Unrestricted File Upload.

One of the key operations performed by stalkerware applications is to regularly upload

³<https://owasp.org/www-community/attacks/xss/>

⁴<https://github.com/mandatoryprogrammer/xsshunter-express>

data from the phone, including photos, videos, and other files from the victim’s phone storage to a remote server. We observed that 22 of the tested apps can access the device’s internal storage, e.g., downloaded files, SD card storage, and even system files if the app is given admin rights. However, such file uploads lack file content verification⁵ during data synchronization, allowing custom files (with any chosen payload by the victim/investigator) to be transferred from the phone to the backend server and later downloaded by the abuser. An investigator could take advantage of this behavior to collect information about the abuser and their environment by uploading files⁶ with specifically crafted payloads that would be triggered when the abuser downloads those files from the stalkerware dashboard, and opens them on their machine. Using a similar method as for XSS, they could place files in the victim’s phone and wait for them to be uploaded to the dashboard.

With custom updating times ranging from as short as 5 minutes to daily reports, it is up to the abuser to decide how frequently the server should fetch information from the compromised device. In most cases, however, the kind of data uploaded by the stalkerware is not controlled as thoroughly. Even though some apps hide pieces of information behind paywalls (WhatsApp messages monitoring are often provided as a “premium” feature), it does not mean that the related data is not fetched by the app.

One particular feature of stalkerware apps allows the abuser to have access to the device’s internal storage (Downloaded files, SD card storage, even system files if app is given admin rights). While this functionality does not usually allow any editing to be made to the files, the abuser can still navigate through the phone’s storage and download any file they want.

Broken Authentication and Access Control.

Since the majority of stalkerware apps use a centralized platform to store victim’s data,

⁵https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload

⁶<https://canarytokens.org/>

we verify if these platforms are vulnerable to broken authentication and access control.⁷ We first create two accounts (one for the abuser and one for the investigator) on the stalkerware dashboard. We install the abuser app on the victim's phone using the abuser's credentials and browse the abuser's dashboard. We run Auth Analyzer⁸ in the background while we browse, by configuring it to replay requests using the investigator's account session tokens after logging in from another browser. An access control vulnerability is detected in case a replayed request (from the investigator's session) generates the same response as the browsed request (from the abuser's session). Similarly, to test for broken authentication, we configure Auth Analyzer to replay requests using null or blank sessions; a successful response code indicates the presence of a broken authentication.

Furthermore, we notice that JSON Web Tokens (JWTs) are commonly used for authentication, managing user sessions, and controlling access to resources in stalkerware applications. We check if the JSON Web Token (JWT) signature field is properly verified by logging into the abuser dashboard and collecting all corresponding tokens. We then test for all signature-related flaws by supplying collected tokens to an open-source tool.⁹ Note that the signature field allows the server to authorize the request after ensuring that the token has not been tampered with.

For data uploads from the victim phone to the stalkerware backend server, the server needs to identify and authenticate the device and store the transferred information accordingly. To test authentication flaws in this case, we use Burp Suite's built-in proxy to intercept data sent by the mobile device and check the packets sent during data upload. Without proper device authentication, it is possible to upload data to the backend server on behalf of the victim's device. This specific vulnerability can be used by an investigator to send

⁷https://owasp.org/Top10/A01_2021-Broken_Access_Control/

⁸<https://github.com/PortSwigger/auth-analyzer>

⁹https://github.com/ticarpi/jwt_tool

payloads or other information to the dashboard without having to interact with the compromised device. Crafting an upload packet without the victim's phone, however, requires knowledge of the data upload mechanism, which varies from app to app.

Some dashboards also provide features to send remote commands to the phone. Examples of possible commands are locking the phone, taking a picture from the front and/or back camera, recording ambient sounds with microphone, taking a screenshot, displaying alerts on the screen, deleting data, uninstalling the stalkerware app. Whenever the APIs used for issuing such commands share the same API key than the ones called by the mobile device application (for data upload, settings retrieval, license registration), such remote commands can be used as counter-measures if the API key get leaked.

Insecure (Online) Storage.

The victim's multi-media data (e.g., screenshots, images, videos, call recording audios) are generally stored differently compared to text-based data (e.g., social media chats or text messages). Text-based information is often directly displayed on their corresponding dashboard page, while pictures or videos can be shown through previews on the dashboard since viewing them requires generating a URL. It can be stored either on the cloud (e.g., AWS), or on the stalkerware's server itself, albeit in a different directory. In both cases, we check if access to sensitive multi-media content is protected. If access to such files is possible, the investigator could find evidence of data collection by the stalkerware.

First, we collect and store the list of all relevant multi-media URLs. This is done by first syncing the victim's device data to the stalkerware's server and then manually browsing the abuser's dashboard to identify all such URLs. We then make a curl¹⁰ request to each of the collected URLs without providing any authentication token. A successful response (with 200 OK status code and content body) indicates the presence of insecure storage of multi-media data. Second, we check if it is possible to guess the URLs to access the multi-media

¹⁰<https://curl.se/>

data of other victims. For example, the use of high entropy tokens (e.g., Universally Unique Identifier (UUID)) in the URL makes guessing infeasible, whereas the use of short numeric identifiers makes it possible for an adversary to quickly form and test potential URLs that may contain other victims' sensitive information. In the case of high entropy tokens, we make use of the Wayback Machine¹¹ to find any leak of such tokens. Lastly, we repeat the process of triggering curl requests on top of the log file, after deleting the abuser's account from the platform and uninstalling the Android app from the victim's device. This helps us to verify the retention status of the victim's multi-media data. This method could be used by the investigator to check whether the victim's information remains available after the abuser's account deletion.

Insecure (Local) Storage.

Stalkerware apps also use the phone's internal storage to cache data such as collected contact names, text messages, installed apps, key logger history, and app activity. These internal files may contain credentials used for data uploads to the backend servers, as well as information used to link the phone to the abuser's account. Typically, accessing the content of the internal storage of applications requires root privileges on Android. However, by leveraging Android application backup functionality, the same can be done without rooting the phone. In both cases, we use ADB to pull the stalkerware's internal files. If the app sets the "debug protection" parameter to prevent users from tampering with its local directory, we use the Android backup functionality to fetch the app's data. In other cases, we can directly pull the app directory with ADB pull (with a rooted phone) and browse the SQLite databases with an online tool.¹²

Cross-Site Request Forgery (CSRF). In stalkerware apps, it is possible to induce abusers to perform actions that they do not intend to perform (e.g., sending remote commands to the

¹¹<https://archive.org/web/>

¹²<https://inloop.github.io/sqlite-viewer/>

victim’s device) via CSRF.¹³ It can be exploited by sending a link to the abuser and luring them to click on it. We detect CSRF vulnerabilities in all of the state-changing HTTP/s requests that are triggered upon browsing the stalkerware dashboard, from an abuser account. First, we check for the presence of any anti-CSRF tokens in the request body. Second, we check if those tokens are tied to the abuser session. Specifically, we test if the request can successfully be processed by supplying any valid anti-CSRF token. To do this, we log into the investigator’s account and provide their anti-CSRF token to the abuser’s state-changing requests. Successful execution of this request indicates the presence of CSRF.

3.2 WARNE Workflow

In this section, we describe the workflow of our tool, WARNE, for collecting evidence against the abuser. We leverage possible vulnerabilities in stalkerware apps and the unprotected content of the local storage of these apps on the victim’s device (see Sec. 3.1.3). WARNE generates a report consisting of all the data (e.g., IP address, email address) that can be used to infer the identity of the abuser. See Fig. 3.2 for an overview of WARNE’s workflow. The tool’s mechanism is split into eight steps:

1. *Preliminary setup.* Upon starting, WARNE allows the user to configure the analysis through three separate settings by providing: (a) a custom XSS payload pointing to a preemptively configured server; (b) a suspicious package name that may not be in our known stalkerware list; and (c) the device’s backup password/secret if the analyzed phone is rooted or encrypted (i.e., when all backups are password-protected).

2. *Package detection.* WARNE scans the device to list all installed packages and flags suspicious ones: if the package name is present in AssoEchap’s stalkerware Indicators of Compromise list (which is fetched from its GitHub repository to ensure that it is always up-to-date), or if the package name matches the one provided by the investigator.

¹³<https://owasp.org/www-community/attacks/csrf>

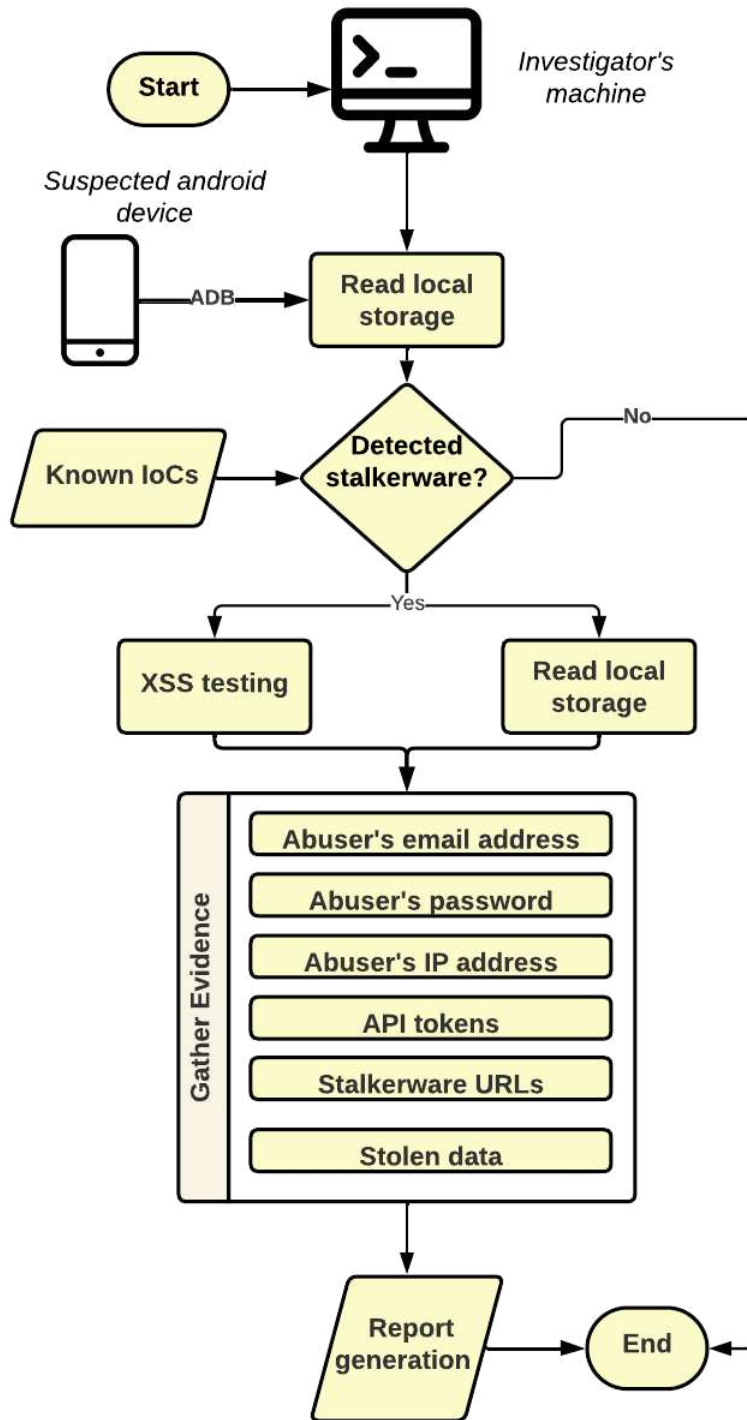


Figure 3.2: WARNE’s evidence collection process

3. *Information dump.* For each flagged app, WARNE dumps information about the package (this does not require root). This information contains the app's installation date, requested permissions, local storage path, actions, and intents. We also check whether the app requires the phone to be rooted to fetch its local files.

4. *Local data extraction.* During this step, WARNE fetches all files related to the suspicious app that are stored locally. This is done differently depending on whether the target device is rooted or not. If the app's manifest explicitly states that backups are not allowed, the app's files cannot be extracted on a non-rooted phone.

5. *XSS injector and listener.* WARNE offers compatibility with XSShunter-express, a self-hosted XSShunter server. We automatically inject a user-given payload in the target device and listen for any payload trigger. WARNE then identifies the trigger source and appends payload information to the related text report. This listener feature requires WARNE to either run in the background on the computer or to be regularly booted up. Note that the XSShunter-express server needs to be configured separately, on an instance with a public DNS record.

6. *Report content collection.* Information gathered in the previous steps is then stored in a readable format. This includes a dump of the package information, all local file names, and their readable content (including eXtensible Markup Language (XML) files, databases, and any other files containing textual information). Files that are empty, unreadable, or cannot be opened, are also logged in the report.

7. *Report parsing.* Once the full report content has been gathered, WARNE parses the collected data for relevant information. This gives users an initial idea of what can be found in the report, which they can then analyze further.

8. *App removal.* This optional step allows the user to uninstall the flagged app. Additional instructions are provided if the removal fails (e.g., due to the app being installed as a device admin). It should be noted that attempting to remove stalkerware apps may be

noticed by the abuser and could lead to reprisal.

3.3 WARNE - Implementation

In this section, we first expand on the set of prerequisites required for WARNE to function correctly. Thereafter, we present details on how specific features (e.g., testing local storage) of WARNE are implemented.

3.3.1 Prerequisites and Additional Tools

It is important for our tool to run outside of the compromised device's scope. This is why WARNE is designed to run on Linux and analyze connected Android devices. The analysis code is written in Python and features an in-browser graphical user interface for readability and ease of use. This interface is handled by the Dash library.¹⁴ Communication with the phone is handled by the Android Debug Bridge (ADB), which must be installed on the host machine as well as enabled on the target phone. The latter can be done by turning developer options on and allowing USB debugging. The ADB tool lets us run commands on the target device and is core to most features offered by WARNE, including package detection, data extraction, app removal, and XSS injection. It relieves us from having to install an app or agent on the phone, which could be flagged by the stalkerware and detected by the abuser.

WARNE uses the Android Backup Extractor¹⁵ to extract data from encrypted backups. WARNE also uses Android Asset Packaging Tool (AAPT)¹⁶ to perform various requests about packages installed on the target phone. AAPT can dump data about a specific app, which is used to fetch information from an app's manifest, such as its common name (i.e.,

¹⁴<https://plotly.com/dash/>

¹⁵<https://github.com/nelenkov/android-backup-extractor>

¹⁶<https://developer.android.com/tools/aapt2>

the one displayed on the phone's interface). Upon startup, WARNE automatically checks for AAPT's presence and downloads it if necessary.

The handling of XSS payloads and payload triggers is done by XSShunter-express, a containerized instance of XSShunter with custom settings and self-hosting capabilities. To allow compatibility with WARNE, we modified the tool to create text reports of triggered payloads and linked them to the machine hosting WARNE using the Secure SHell File System (SSHFS). Our tool then regularly checks for trigger files' creation and updates its corresponding app's text report. To identify the trigger source app, we compare its originating URL to our IOC threat list and find a corresponding package name. In case no related app is found, the payload information is also stored in a separate file with all other recorded triggers. Since this feature requires a substantially complex setup, its setup is not mandatory for WARNE to function. Additionally, as this XSShunter instance is publicly accessible, standard server hardening guidelines (e.g., see [4]) must be followed.

3.3.2 Implementation of WARNE Features

Stalkerware Detection.

The tool fetches information about known stalkerware apps from a GitHub repository [10]. Among other details, the repository provides the package names of known stalkerware apps. The tool queries the device to obtain a list of packages installed on the victim's phone and checks if any such packages are installed on the device. If found, the tool records details such as the name of the detected stalkerware, the date of installation, and the permissions given to the app.

Testing Local Storage.

Typically, accessing the content of the internal storage of applications requires root privileges on Android. However, by leveraging Android application backup functionality, the same can be done without rooting the phone. Depending on whether the analyzed phone

is rooted or not, WARNE uses the most convenient method to fetch local data related to the identified/suspected stalkerware. For each known malicious app detected, the tool issues a backup command and copies the backup to the analysis system. The tool then parses the backup and reads the content of each file in the application's internal storage. For each SQLite database, it reads all the tables in the database and converts them into a readable text format. Note that the backup functionality may not work on every stalkerware, as it can be disabled via the `allowBackup` flag in the app's manifest. Even though it could be disabled through re-packaging the app, doing so would delete previously stored data.

Report Generation.

WARNE is designed to provide a thorough and easy-to-read text report of its analysis. An exhaustive report is created by appending all data (app stats and files that can be displayed as text) to a single text file. The report is then automatically parsed with regular expressions to find relevant information. This includes any email address that is not the phone's primary user address, host names, occurrences of words such as "username" or "password" and recognizable Google API keys/tokens.

XSS Reports.

WARNE sets the XSS payload by creating a contact on the phone whose name is the payload given by the victim/investigator during setup. The user can also choose to clear all the injected contacts from the device's contact list. Simultaneously, a thread is spawned to monitor the local folder linked to the XSShunter-express server. Information gathered by a triggered payload includes the time of the trigger, the source IP address, the URL of the page the payload has been fired on, its referring page, Non-HTTPOnly cookies, HTML data, and User-Agent.

3.4 LLM Experimental Setup

With data collected by WARNE, investigators can find details about the abuser, the stalkerware itself or evidence of data collection. However, the generated report can be long and cumbersome to parse manually. Automatic keyword parsing is already implemented for easily identifiable details such as email addresses, URLs and passwords, but the variety of structures used by stalkerware apps to store data makes this method hard to rely on entirely. Therefore, there needs to be another that could adapt to different dumped files and infer meaning from various text inputs.

Following this direction, it appears that Large Language Models can be highly efficient for such summarization tasks, where no knowledge — external to the report context — is required. We decided to test the accuracy of two different models: (1) ChatGPT through its online GUI and (2) a locally run model of Llama3. We used both models to parse and analyze reports generated by the analysis of 20 stalkerware apps.

Both models were tested by giving them a fine-tuned prompt detailing their role and what information they should look for, before giving them the text report to parse. This first set of instructions can be found in Figure 3.3. Their response to this first prompt allowed us to compare their accuracy to our manual report analysis and list elements which the model was not able to identify. We then ask specific questions about missing details to see if the model can find them. Such questions include: “Double check for the presence of email addresses”, “Double check for SQLite databases”, “Any relevant file or table name?”, “Check the SQLite tables’ content for relevant information”.

3.4.1 ChatGPT

The versatile nature of ChatGPT makes it efficient at handling any kind of natural language processing, text summarizing and data parsing task. We decided to use ChatGPT 4o

to test its ability to analyze WARNE generated reports. We do so through the online API¹⁷. Even though data is shared with openAI through ChatGPT discussions, for our tests, we only upload reports containing our own data. Real investigation scenarios may require the information they are processing not to be shared with any third-parties, thus making locally ran models more convenient. Running our tests with this model, however, allows us to get an idea of what supposedly high-quality results should look like, and what to expect from local models with the right conditions.

3.4.2 Llama3

We used Ollama¹⁸ version 0.1.45 to run a local version of Llama3-8B on a Linux computer system and gave prompts through a python script. Our machine runs on an i7-11700 intel CPU at 2.50GHz, with 16Gb of memory. Other models were considered for local testing, such as roberta-base-squad2¹⁹, which did not handle inputs as large as our text reports, or Llama3-70B, which yields better accuracy but requires the most processing power and thus greatly increases the time required to generate an answer.

¹⁷<https://chatgpt.com/>

¹⁸<https://ollama.com/>

¹⁹<https://huggingface.co/deepset/roberta-base-squad2>

We are investigators trying to help a stalking victim whose phone has been compromised. Our tools generated a text report composed of the stalkerware app's metadata, as well as a dump of the app's local files (readable files, databases). This report is not user friendly and is hard to parse manually. Your goal is to parse the report and highlight all relevant information that could be used as evidence of the stalking.

Instructions for Analyzing a stalkerware Detection Report:

Objective:

Analyze a stalkerware detection report to extract relevant information that proves the phone was being monitored and identify details about the abuser.

Steps:

Identify Application Information: **#Basic app information**

Extract the app package name. Note the app installation path and any associated files. Extract version information, including version code and version name. Note the first install time and last update time.

List Permissions: **#We look for common stalkerware-related permissions.**

Identify and list all permissions granted to the application, focusing on those related to accessing sensitive data (e.g., location, SMS, call logs, installed apps, internet, and network state).

Extract Dumped Files and Content: **#Relevant data can be found in configuration files.**

Identify files dumped by the report. Look for specific markers like "==" indicating the start of a file and "=" indicating the end. Summarize the content of each file, noting any relevant data such as configuration settings, network requests, and temporary files.

Analyze SQLite Databases:

Identify SQLite database files mentioned in the report. Each database is identified by a line of '=' with the label "SQLite database", followed by a dump of all tables in the database, separated by lines of '-' and labeled by table names. Extract and list the tables within each database. Summarize the contents of these tables, focusing on data that indicates monitoring activities (e.g., message history, call history, location tracking). Use the table names to infer their purpose if not explicitly stated.

Search for Email Addresses and Contact Information:

Search for any email addresses or phone numbers mentioned in the report. Note where these email addresses or phone numbers are found (e.g., in configuration files, database tables). **#This can be asked separately for better accuracy/reliability.**

Highlight Evidence of Monitoring:

Provide details on specific files or database entries that indicate the phone is being monitored (e.g., logging modules, Google Analytics usage, detailed configuration settings). Highlight any user activity logs, tracking data, or other surveillance indicators. **#Similarly, this can be asked separately after getting a first answer to get a more detailed response.**

Summarize Findings:

Provide a concise summary of the key findings. Include any crucial information that can help identify the abuser, such as email addresses, phone numbers, or specific monitoring configurations.

Figure 3.3: Initial prompt given to both models as a first set of instruction, along with related WARNE generated report. We put additional comments in blue text, which are not part of the prompt.

Chapter 4

Stalkerware Analysis Results

We present the results of both our stalkerware security analysis and our WARNE tests. We first provide a thorough list of all features offered by the stalkerware apps in our dataset before presenting the security weaknesses found in these features that could be leveraged to efficiently gather data about the abuser. We then show the results of testing our tool on our dataset of 30 Stalkerware apps. Finally, we present our efficiency tests of WARNE report parsing by Large Language Models.

4.1 Stalkerware Apps Capabilities

Most stalkerware apps use two separate systems in parallel: a stalkerware app installed on the phone and a web-based dashboard accessible by the abuser. This platform is linked to backend databases where the collected data can be found and also serves as a control panel through which the abuser can manage their subscriptions, enable/disable features, or send remote commands to the phone.

Table 4.1 compiles a comprehensive list of the data collected by 30 stalkerware apps for Android devices (duplicates excluded). 26 of them gather text messages and phone call logs, 27 of them feature GPS tracking and geofencing (triggering alerts whenever the target

leaves a specified area), these are the most common capabilities available on such apps. Other noteworthy functionalities include secret live recording with the device's camera (17) or microphone (13), a key logger collecting keystrokes, therefore potentially disclosing the victim's passwords to the abuser (14), access to file storage like photos, videos, or documents (16) and social media chat services such as Facebook Messenger, Instagram, WhatsApp or Viber (18).

After being collected by the Android app, the victim's data is sent to the abuser in the following ways. (1) In 29 out of 30 cases, data is uploaded to an online database which can be browsed by the abuser through the web dashboard. The stalkerware database therefore stores all pictures, text messages, contact names, and other collected information from the devices monitored by the platform. (2) When not using a centralized database system, data can be directly sent to the abuser's email address via regular reports. Apps using this approach however tend to offer fewer features than the database approach.

4.2 Security Vulnerabilities

The complete list of vulnerabilities found within each app can be found in Table 4.2. **Cross-site scripting (XSS).** We identified 20 different apps whose online dashboards did not conduct any input validation before displaying the collected victim data on the web page. Depending on where the XSS payload was planted (e.g., contact list, text message, file names), the investigator can see related data displayed on the web page on the XSS report. For 18 out of 20 apps, XSS could be performed through text message injection, by either sending or receiving a message containing a payload. Among these 18 apps, 15 of them feature social media compatibility and were therefore also vulnerable to XSS payload injections through social media chat services. 16 out of 20 apps were vulnerable through the contact list and 11 out of 20 through filenames.

XSS payloads can also be used to verify the presence of a stalkerware app on a phone,

e.g., by sending text messages or adding a contact name containing a payload to the device and waiting for the potentially compromised phone to upload it to the app's backend servers. It should however be noted that this approach relies on the abuser logging into the online dashboard and loading the page displaying the payload. It is also possible that such an attack could raise an abuser's suspicion, if they notice strangely formatted messages on the dashboard. As XSS payloads are customizable, they could be programmed to send a notification when it is activated. This would make it possible to hijack the abuser's session without delay, with increased odds that the session cookies are still valid to the server.

Broken authentication. We found that 8 stalkerware apps are vulnerable to account hijacking or unauthenticated command transmission due to broken authentication. Two apps (CatWatchful and Shadow Spy) use the Google Identity Toolkit for credential verification and account management. It uses a token to identify the abuser on the victim's device, which is exposed inside the */shared_prefs* directory on the phone. This token can be used to issue commands to the monitored device, but also to request API calls through the Google Identity Toolkit (e.g., to delete the abuser's stalkerware account). Another app (Lost Android) uses Google Cloud Messaging (GCM) to upload collected data (using Google's servers as intermediates for data upload and commands). The GCM key can be found unprotected on the victim's phone and could be used to craft data upload packets.

A button in the CatWatchful app also redirects to the abuser's online dashboard and leaks their credentials in the redirection URL. The dashboard of LetMeSpy was only accessible through HTTP, therefore exposing the abuser's credentials. The JWTs used for authentication on Spyc's dashboard were vulnerable to null signature attacks, allowing for easy account takeover.

To authenticate the device to the backend platform during data uploads, stalkerware apps can use multiple identifiers, including a license number entered by the abuser, the phone's International Mobile Equipment Identity (IMEI) number, a fixed session ID, or

the abuser's credentials. Data uploads from the mobile device are poorly secured in four apps, allowing replay attacks on packets sending information about the phone, installed apps, contacts, messages, or GPS location. Authentication of the device is made with the license entered by the abuser and a session ID that stays unchanged even after multiple data uploads.

We also found that none of the stalkerware apps use certificate pinning. This allows for easy interception of the packets during data upload with only a few configuration steps on the phone. This means that any man-in-the-middle¹ attacker could collect the stalkerware authentication credentials with a proxy.

Insecure data storage. We identified issues regarding insecure data storage in 6 tested stalkerware apps. 3 of them failed to curtail access to files such as pictures, that were requested by the abuser on the online dashboard. Such files were accessible via static URLs, allowing unrestricted access to the file to anyone, regardless of authentication. However, the generated links had a limited period of validity (24 hours on average).

This vulnerability was mostly tested with pictures uploaded from the phone to the stalkerware's backend server and then requested from the dashboard. However, as it is a flaw inherent to the backend database configuration, all other data that can be given a URL on request from the dashboard could potentially be accessed by an unauthorized person. For pictures, generated URLs are made up of a mobile device's identifier along with either the time-stamp at which the picture was taken or uploaded, or seemingly random tokens.

Stalkerware apps also keep sensitive data about the abuser on the mobile device itself. 4 different apps store information in easily accessible locations on the mobile device, such as shared preferences. Data such as the abuser's email address, the stalkerware registration license, the application unlocking PIN code, and even the abuser's password can be found in the internal files. Even though some cases require the phone to be rooted, these pieces of

¹<https://www.rapid7.com/fundamentals/man-in-the-middle-attacks>

information can be used to identify the abuser or execute commands that would be reserved for the stalker.

Two apps provide functionalities to uninstall the stalkerware application from the phone, either remotely or from the phone itself. These ways of deleting the app differ from manually removing it from the phone's settings, as mechanisms are used to prevent access to such features (automatically redirecting the user to another legitimate app's settings when trying to access the stalkerware settings). These functionalities require authentication to be used, which can be bypassed by looking for the corresponding password/verification token stored on the phone.

In three stalkerware apps, we also found databases containing a summary of all gathered data, as well as credentials like the abuser's email address and password or the device's identifier to the backend server. These databases could be used as evidence of the collected information in case of account deletion on the abuser's side. We also identified 7 websites that use Google Firebase as their online database service, with 3 being misconfigured, leading to partial or complete leakage of all users' information.

Unrestricted file upload. During our analysis, we have not found a stalkerware conducting any kind of file verification when requesting files from the dashboard. This means that sending malicious files to the backend servers for them to be downloaded by the abuser is easily doable. Anyone knowing the victim's phone number could send a malicious file (e.g., via a text message). The stalkerware will then automatically upload it to the online dashboard for it to be downloaded by the abuser.

The protection provided by the abuser's system is the only variable that could influence the gravity of such a vulnerability. Combined with data transfer presenting broken authentication mechanisms, an attacker could send files containing malicious code to the dashboard without having to download them on the phone. Someone could also send the payload directly to a victim unaware that they are being monitored, as the file only needs to stay on

the phone for a relatively short amount of time for it to be uploaded to the stalkerware's backend server.

Cross-site request forgery. We found 4 apps that are vulnerable to cross-site request forgery (CSRF) attacks. The change password functionality in LetMeSpy is vulnerable to CSRF, making it possible to take over the abuser's account. In Spapp Monitoring and Panspy, it is possible to change the destination email address of the app's notifications to receive all information in place of the stalker.

Name	Text messages	Calls	GPS	Contacts	Camera	Microphone	Notifications	Files	Key logger	Screen	Apps	Social media	Browsing history	Remote cmd.	Wi-Fi
AllTracker	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
Android Monitor	✓	✓	✓				✓		✓		✓	✓			
CallSmsTracker	✓	✓	✓									✓			
CatWatchful	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Cerberus					✓			✓	✓	✓				✓	✓
Couple Tracker	✓	✓	✓												
Easy logger	✓	✓	✓								✓				
Free Android Spy			✓	✓	✓									✓	
i-Monitor	✓	✓	✓	✓	✓						✓		✓		
iKeyMonitor	✓	✓	✓	✓	✓		✓		✓	✓	✓		✓	✓	
LetMeSpy	✓	✓	✓											✓	
Lost Android	✓	✓	✓		✓	✓					✓		✓	✓	✓
Meuspy	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
MobileTrackerFree	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓	
MobileSpy	✓	✓	✓	✓				✓	✓		✓	✓	✓		✓
Mycellspy	✓	✓	✓	✓	✓	✓		✓	✓			✓	✓		
OwnSpy	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Panspy	✓	✓	✓					✓	✓		✓	✓	✓		✓
Remote Audio Rec.						✓									
Reptilicus	✓	✓	✓			✓		✓				✓		✓	
Shadow SPY	✓	✓	✓	✓			✓	✓		✓				✓	
Snoopza	✓	✓		✓	✓					✓		✓	✓		
SpAppMonitoring	✓	✓	✓			✓		✓			✓	✓	✓		
Spy24	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓
Spyhuman	✓	✓	✓	✓		✓		✓			✓	✓	✓		
Spyic	✓	✓	✓					✓	✓		✓	✓	✓		
Spylive 360	✓	✓	✓		✓	✓		✓			✓	✓	✓	✓	
TheTruthSpy	✓	✓	✓	✓	✓	✓	✓		✓		✓	✓	✓		✓
Tispy	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓		
Track My Phone Rem.			✓		✓										

Table 4.1: Features available on tested stalkerware apps.
Legend: A check-mark means that data is collected by the app.

Vulnerability	Apps
XSS	Meuspy, Message Call tracker, OwnSpy, Track My Phone Remotely, Tispy, MobileTrackerFree, Spytomobile, Free Android Spy, iKeyMonitor, Shadow SPY, SpAppMonitoring, Flexispy, Alltracker, A-spy, Mycellspy, Android Monitor, Spy phone labs phone tracker, CallSmsTracker, Lost Android, Reptilicus
Unrestricted File Upload	All apps
Broken Authentication	CatWatchful, Shadow SPY, LetMeSpy, Message Call tracker, Lost Android, SpAppMonitoring, Reptilicus, Spyc
Insecure Storage	Meuspy, CatWatchful, iKeyMonitor, Shadow SPY, AllTracker, Spyc
CSRF	LetMeSpy, OwnSpy, SpAppMonitoring, Panspy

Table 4.2: Vulnerabilities identified within each tested apps.

Chapter 5

WARNE Test Results

We now present the results of our tests on WARNE on known stalkerware apps, and our report processing results by Large Language Models.

5.1 WARNE Tests on Stalkerware Apps

We tested WARNE’s effectiveness on a non-rooted Samsung Galaxy M02 phone running Android 11, and then on the same device after rooting it. We chose to use two different setups for our testing conditions to be closer to real-life situations. We tested our tool on 30 free stalkerware apps (or premium ones offering free trials) and successfully flagged 24 of them. It is still possible to analyze them with WARNE by manually providing the package name to the tool or by enabling non-trusted source detection (flagging apps not installed from the Play Store).

Out of the 30 apps we tested, 12 of them allowed ADB backups of their local data to be performed. After extracting the files from these apps’ local storage and parsing the generated reports, we found occurrences of the abuser’s email address in 15 of them. 5 apps also stored the abuser’s password (in clear text, or as an MD5 hash for OwnSpy) next to the email address. OwnSpy also stores a value called “encryption_key” as an MD5 hash

which is used to communicate with the server. The abuser's password for CallSmsTracker's account was also found in a .xml file, but was labeled as "spin".

Key logger features offered by stalkerware apps can list a large amount of inputs in tables. Tables containing key logger data are the main trigger cause for the detection of certain elements such as email addresses. We manually went through occurrences to verify their relevance and detect false negatives. Five apps (PanSpy, Meuspy, MobileSpy, SpyLive360, and AllTracker) featured key logger tables with all collected inputs.

URL detection can be permissive enough to be able to flag links with no "https://" or "www." prefixes but could sometimes result in longer processing time. However, manual verification of such cases is possible (for example, if the flagged text is the package name). 6 occurrences of the stalkerware dashboard's URL were found in WARNE's generated reports. Other kinds of URLs were also found, like in some of PanSpy's SQLite tables labeled "webWhite", "WebBlack", "webFilterClass" and "traffic" (the first three actually being web filtering lists with specific domain names. Most of the stored URLs pointed towards domains like youtube.com, amazon.com or aol.com, and one of them pointed to a taobao.com domain.

Finally, two apps (ShadowSpy and SpAppMonitoring) triggered the XSS payload we had injected into the phone's contact list. However, we noticed that some online dashboards that displayed the XSS payload without input sanitization were not necessarily firing it back to our XSShunter-express server. It could be possible to avoid this issue by trying other injection media (i.e., text messages, file names, etc.) because some online dashboards use different display mechanisms depending on the data (some of which might be more responsive to XSS payloads). Table 5.1 shows a summary of items found by WARNE in the tested stalkerware apps (unlisted apps provided no relevant result).

5.2 WARNE Report Parsing

Compared to our manual analysis, ChatGPT found 84% of the relevant elements, with 55% highlighted in its first response. Our local Llama3 instance noted 29% of manually found elements in its first answer, and 16% more with separate questions. We now go into the details of what each model was able to identify.

5.2.1 ChatGPT

ChatGPT showed an overall better accuracy than the local model, with 5 instances where the model found as many details as the manual analysis after the first prompt. It was able to identify email addresses in 11 reports out of 15 found manually. With additional questions pinpointing email addresses, GPT4o was able to find all 15 occurrences of relevant email addresses in the given reports. Another kind of information that GPT4o efficiently identified was inferring meaning to the dumped data structures and identifying evidence of data collection. This includes the analysis of SQLite databases, configuration files and other dumped text data that often contains proof that information is being gathered from the phone. These proofs can be found in the form of table entries containing sensitive information, or simply explicit column names for empty data frames. GPT4o gave a satisfying justification in 8 cases out of 11 but did not give any additional results when asked specific questions.

Other notable elements ChatGPT was capable of identifying include:

- URLs pointing towards the stalkerware's dashboard or backend server (found in 6 out of 6 cases, with 3 of them in the first answer). These URLs can provide additional proof of data collection as websites affiliated to known stalkerware apps can be easily identified by their domain name or domain registration.

- Instances of locally stored passwords, which are in clear text in 2 reports and encrypted in 3, but still labeled as "password" or other synonyms ("pass", "pw", etc.). The model

successfully identified 4 of them in its first answer, and a fifth one with additional questions. These passwords can be part of the abuser's credentials (found along their email address) or the app settings (e.g. PIN code to access the app).

A major ChatGPT drawback is its tendency to hallucinate, making false claims using data that does not exist. Such issues occurred occasionally, especially when dealing with bigger reports. Hallucinations can be easily detected by checking if quoted elements do exist in the `WARNE` report or, in some instances, by either asking the model again or resetting the context and starting again from the initial prompt.

5.2.2 Llama3

The first notable difference between both models is the required time to generate a response. As the local model's response delay depends on the hardware on which it runs. ChatGPT's average response time was in the order of seconds, whereas Llama3's local model usually took a minute to process our long initial prompt and generate an answer.

Furthermore, the average Llama3 answer was shorter than ChatGPT's, impeding the response's thoroughness and thus yielding less accurate results.

Even though very few specific details were identified by the local model (only 2 email addresses were found upon first prompt, 3 more after further questioning), it was particularly efficient at identifying evidence of data collection and justifying it with report data. It did so with human readable content and easy to verify claims (by including the file name in which details were found, which can then be manually searched for in the report and parsed for verification). The model gave accurate evidence of data collection in 11 cases, and in 2 more with separate questions.

It appears that the local model handles short, precise prompts better than long thorough ones. It lacks, however, the processing power to be as time-efficient as ChatGPT, albeit no hallucinated information was given. For comparison between ChatGPT and Llama3, all

results from reports analysis can be found in Table 5.2.

We hypothesize that the difference between ChatGPT and Llama3 mainly comes from hardware, as ChatGPT benefits from better resources than our setup. Their capability of quickly processing long text inputs makes them useful for precise information extraction, especially if asked specific questions. Thorough fine-tuned prompts are better suited for streamlined, large-scale analysis, which is usually not required for forensics investigation focusing on one or two suspicious apps.

App	Abuser's email	Abuser's password	Dashboard URL	XSS trigger	API tokens/credentials	Other URLs	Stolen data
AndroidLost	✓				✓		
CallSmsTracker	✓	✓					
CatWatchful	✓	✓	✓				✓
Cerberus					✓		✓
Mobilespy							✓
MyCellSpy	✓		✓				
PanSpy	✓		✓		✓	✓	
OwnSpy	✓	✓			✓		✓
Shadow Spy	✓	✓		✓	✓		
Spappmonitoring			✓	✓	✓		✓
Spy24	✓	✓			✓		
AllTracker	✓					✓	✓
EasyLogger	✓				✓		✓
FreeAndroidSpy	✓						✓
iKeyMonitor	✓						
Meuspy	✓				✓	✓	✓
MobileTrackerFree					✓		✓
SpyLive 360			✓		✓		✓
Spyhuman	✓						
TheTruthSpy	✓	✓	✓			✓	

Table 5.1: WARNE-parsed evidence and other relevant values found in stalkerware apps. Apps in bold font require the phone to be rooted to fetch files from the phone.

App	Manual Inspection	chatGPT		Local Llama3	
		Initial prompt	Separate questions	Initial prompt	Separate questions
AndroidLost	mail, api token	mail	token	token	x
CallSmsTracker	mail, password	mail	x	x	x
CatWatchful	mail, password, dashboard, data	mail, dashboard, token	x	data	dashboard
Cerberus	api token, data	x	x	x	x
Mobilespy	data	data	-	x	data
MyCellSpy	mail, dashboard	x	mail, dashboard	x	x
PanSpy	mail, dashboard, token, url	data	mail, dashboard, token	data (!)	x
OwnSpy	mail, password, token, data	mail, password, token, data		data	mail, password, token
Shadow Spy	mail, password, token	mail, password, token	-	x	x
Spappmonitoring	dashboard, token, data	dashboard, password	-	data	x
Spy24	mail, password, token	x	mail, password, token	data	x
AllTracker	mail, url, data	mail, data	url	x	x
EasyLogger	mail, token, data	mail, data	token	data	data
FreeAndroidSpy	mail, data	mail, data	-	data	x
iKeyMonitor	mail	x	mail	x	x
Meuspy	mail, token, url, data	mail, data	mail	mail, data	mail
MobileTrackerFree	token, data	x	token	x	x
SpyLive 360	dashboard, token, data	mail, data	dashboard	mail, data	mail
Spyhuman	mail	x	mail	token, data	x
TheTruthSpy	mail, password, dashboard, url	mail, password, dashboard, url	-	dashboard, data,	dashboard

Table 5.2: Results comparison of WARNE reports manual analysis and LLM analysis.
Legend: “x”: no relevant data was found, even though there were. “-”: nothing more was found than in the manual analysis.

Chapter 6

Conclusion and Future Work

6.1 Key Takeaways

We presented *WARNE*, a tool facilitating the gathering of information and evidence on stalkerware apps and the responsible abuser with no traceable interactions with the victim's device. To build this tool, we performed a systematic experimental privacy and security analysis of 30 unique stalkerware APKs available online and identified their features and vulnerabilities that could be exploited for this purpose. Many of these invasive capabilities were enumerated and experimentally verified to find ways of using them against the abuser. In this context, we found apps/services vulnerable to various exploitable attacks, including broken authentication mechanisms, insecure storage of sensitive data, and other vectors that could be leveraged to gain information about the IPV perpetrator. These findings were used to develop *WARNE*, which was tested against 30 apps and successfully found relevant data including abuser's credentials, dashboard URLs, and stolen data evidence in 20 of them. Additionally, using Generative AIs like Large Language Models as digital forensics tools, can prove to be useful for investigators as it allow for fast processing of large amounts of data. Even though the field has seen significant improvement with newer iterations of models like GPT 4o, the sensitive nature of digital forensics calls for accuracy

and reliability in its involved mechanisms. With carefully crafted prompts and focused instructions, we can benefit from these models' efficiency to generate specialized, human-readable reports from text files and database dumps. Of course, these models are not meant to replace investigators and practitioners during the report analysis process, but can greatly facilitate this task by highlighting relevant pieces of information among large amounts of data. We believe that this first step towards stalkerware evidence collection can lessen the threat posed by such apps by giving concrete resources that organizations and investigators could use to support IPV victims.

6.2 Limitations

iOS Ecosystem.

WARNE's scan feature leverages the Android Debug Bridge utility and is therefore only compatible with Android devices. As there exists no direct equivalent of ADB for iOS devices that are not jailbroken, performing the same actions on Apple mobile phones is not possible.

Windows and macOS WARNE Hosts.

To run commands through ADB and perform other automatic actions, WARNE has been designed to be used on Linux-based distributions. Ensuring compatibility with Windows-based machines would not change the application's process, but would require modifying most of the tool's issued commands.

Variable Amounts of Information.

Our tool relies on oversights from stalkerware apps (allowBackup, debuggable) to gather data on non-rooted phones and without the need to install anything on the device. In cases where these oversights are not present, a smaller amount of information can be gathered about the app.

XSS Payloads.

XSS payload fires rely on the abuser logging into the dashboard and viewing the injected payload. This can be a rare event and thus might not guarantee the feature's effectiveness. In academic and testing contexts, this is not a problem as the user performing the analysis can also access the stalkerware's dashboard to manually trigger the payload. In real life situations, however, it might not be the case.

6.3 Recommendations for Potential Stalkerware Victims

In this section we provide some recommendations about behaviors that could help IPV victim avoid stalkerware apps from being installed on their devices or detect the ones that could have been installed without their consent.

- Keep your phone up close and under surveillance at all times to prevent any unwanted person from accessing it and potentially installing malicious apps. Stay aware whenever someone else could have potentially used your phone, even with your consent. Use strong passwords or PIN codes and avoid sharing them with other people to prevent unwanted use. Immediately verify any suspicion of the presence of a stalkerware app on your phone. If you find any, seek help from a qualified organization or professional.
- Watch out for potential indicators of compromise, including: abnormal (increased) battery consumption, unexpected pop-ups, performance drops, suspicious app duplicates or with blatantly important name (“Wi-Fi”, “Sync manager”, a second “Settings” apps). If you observe any such behaviors, seek help from a qualified organization or professional.
- A common denominator to all stalkerware apps is that they require Google Play

Protect¹ to be disabled in order to stay undetected. Regularly check that the Protect feature of Google Play is active. If disabled, someone could have tampered with the phone. This feature can also be used to easily detect apps that were not downloaded from the Play Store.

- Keep the phone updated to its latest version, as many stalkerware apps are not updated regularly and could lose compatibility with newer system versions. Using dedicated anti-Stalkerware tool could also help verify the presence of a malicious app, but keep in mind that apps installation can be monitored by the stalkerware itself, meaning that the abuser could be notified that the victim is suspicious.
- Another possible scenario (although one that we have not encountered during our analysis), would be the abuser sending a malicious link to the victim, tricking them into installing a malicious app without knowing that it is a stalkerware app. Always be cautious of links leading to app downloads, especially from unknown sources. Stalkerware apps are often disguised as legitimate apps and can be downloaded from outside the Play Store. Always verify the legitimacy of the source you download apps from.

Fighting against stalkerware websites can also lower the amount of stalkerware apps in circulation. It is encouraged to actively report suspicious apps and websites to responsible authorities when encountered. Advertisement platforms such as Google Ads should establish clear policies or blacklists to detect and block advertisements on stalkerware websites, as well as content promoting such applications. Similarly, domain providers, payment services and web hosting platforms have proven to effectively prevent access to malicious websites when reported.

¹<https://support.google.com/googleplay/answer/2812853?hl=en>

6.4 Future Work

An important area that could be further developed is adding new stalkerware detection mechanisms to `WARNE`'s toolkit, which would allow it to flag apps that might not be already known and compiled in the IOC database we are currently using. Android malware detection is an already deeply studied topic, with behavioral detection mechanisms or permission usage analysis being potentials ways to improve `WARNE`'s reliability.

Expanding our scope to the iOS ecosystem is another priority, even though not as urgent as the first. stalkerware apps are more difficult to install on iOS devices due to restrictions put in place by Apple on their products regarding non-verified installation sources. Since iOS is a closed system, most stalkerware apps require the iPhone to be rooted in order to be installed, which is a rare case for casual, non tech-savvy iPhone users. However, this does not make iOS devices immune to such practices, as it is possible for an abuser to offer the victim an iPhone with a preinstalled malicious app. This scenario is even more plausible when put in the context of intimate relationships. Fortunately, if iOS devices require root access to install such apps, it would make the evidence collection process easier too.

Additionally, implementing an in-app LLM parser to `WARNE` would allow to streamline the report analysis process and make `WARNE` more user-friendly, relieving investigators from having to manually set their model up. Doing so would also ensure that processed data is not stored or shared to any third party, which is the case when using online chat bot services. Testing other models like Gemini or Claude 3 and comparing them could also help finding the best setup for such task.

Finally, comparing `WARNE` to commercially available forensics tools would provide more insight into potential improvements and additional features that would separate our tool from other existing ones, while staying focused on the specific cases of stalkerware apps.

Bibliography

- [1] M. Almansoori, A. Gallardo, J. Poveda, A. Ahmed, and R. Chatterjee. A global survey of Android dual-use applications used in intimate partner surveillance apps. In *Proceedings on Privacy Enhancing Technologies Symposium*, 2022.
- [2] Avast. Use of stalkerware and spyware apps increase by 93% since lockdown began in the uk, 2021. <https://press.avast.com/use-of-stalkerware-and-spyware-apps-increase-by-93-since-lockdown-began-in-the-uk>.
- [3] R. Chatterjee, P. Doerfler, H. Orgad, S. Havron, J. Palmer, D. Freed, K. Levy, N. Dell, D. McCoy, and T. Ristenpart. The spyware used in intimate partner violence. In *IEEE Symposium on Security and Privacy*, pages 441–458, 2018.
- [4] CIS. CIS Ubuntu Linux Benchmarks, 2023. https://www.cisecurity.org/benchmark/ubuntu_linux.
- [5] S. Clevenger and M. Gilliam. Intimate partner violence and the internet: Perspectives. *The Palgrave Handbook of International Cybercrime and Cyberdeviance*, pages 1333–1351, 2020.
- [6] J. Cox. Hacker steals customers’ text messages from Android spyware company, 2018. <https://www.vice.com/en/article/qvm44m/hacker-steals-text-messages-android-spyware-company-spyhuman>.

- [7] J. Dalman and V. Hantke. Commercial spyware-detecting the undetectable, 2015. <https://www.blackhat.com/docs/us-15/materials/us-15-Dalman-Commercial-Spyware-Detecting-The-Undetectable.pdf>.
- [8] S. Desai. Why you shouldn't trust "safe" spying apps!, 2018. <https://www.zscaler.com/blogs/security-research/why-you-shouldnt-trust-safe-spying-apps>.
- [9] S. Desai. A new wave of stalkerware apps, 2019. <https://www.zscaler.com/blogs/security-research/new-wave-stalkerware-apps>.
- [10] Echap. Stalkerware indicators of compromise, Dec. 2022. <https://github.com/AssoEchap/stalkerware-indicators>.
- [11] C. El Morr and M. Loyal. Effectiveness of ICT-based intimate partner violence interventions: a systematic review. *BMC public health*, 20(1):1–25, 2020.
- [12] F. Fascendini and K. Fialová. Voices from digital spaces: Technology related violence against women. *Association for Progressive Communications (APC)*, 2011.
- [13] D. Freed, S. Havron, E. Tseng, A. Gallardo, R. Chatterjee, T. Ristenpart, and N. Dell. "Is my phone hacked?" analyzing clinical computer security interventions with survivors of intimate partner violence. In *Proceedings of the ACM on Human-Computer Interaction*. ACM New York, NY, USA, 2019.
- [14] D. Freed, J. Palmer, D. Minchala, K. Levy, T. Ristenpart, and N. Dell. "a stalker's paradise": How intimate partner abusers exploit technology. In *CHI conference on human factors in computing systems*, pages 1–13, 2018.
- [15] C. Gibson, V. Frost, K. Platt, W. Garcia, L. Vargas, S. Rampazzi, V. Bindschaedler, P. Traynor, and K. Butler. Analyzing the monetization ecosystem of stalkerware. *Proceedings on Privacy Enhancing Technologies*, 4:105–119, 2022.

- [16] R. Gibson. Countering tech abuse together, 2018. https://www.virusbulletin.com/uploads/pdf/conference_slides/2019/VB2019-ZakorzhevskyG.pdf.
- [17] D. Harkin, A. Molnar, and E. Vowles. The commodification of mobile phone surveillance: An analysis of the consumer spyware industry. *Crime, media, culture*, 16(1):33–60, 2020.
- [18] S. Havron, D. Freed, R. Chatterjee, D. McCoy, N. Dell, and T. Ristenpart. Clinical computer security for victims of intimate partner violence. In *USENIX Security '19*, pages 105–122, Santa Clara, CA, USA, 2019.
- [19] Z. Jiang, J. Liu, Z. Chen, Y. Li, J. Huang, Y. Huo, P. He, J. Gu, and M. R. Lyu. Llm-parser: A llm-based log parsing framework. *arXiv preprint arXiv:2310.01796*, 2023.
- [20] KasperskyLab. Tinycheck, 2021. <https://github.com/KasperskyLab/TinyCheck>.
- [21] B. Krebs. For 2nd time in 3 years, mobile spyware maker mSpy leaks millions of sensitive records, 2018. <https://krebsonsecurity.com/2018/09/for-2nd-time-in-3-years-mobile-spyware-maker-mspy-leaks-millions-of-sensitive-records/>.
- [22] L. Stefanko. Android stalkerware vulnerabilities, May 2021. https://www.welivesecurity.com/wp-content/uploads/2021/05/eset_android_stalkerware.pdf.
- [23] A. Langton. Stalking stalkerware: A deep dive into flexispy, 2019. <https://blogs.juniper.net/en-us/threat-research/stalking-stalkerware-a-deep-dive-into-flexispy-2>.

- [24] E. Liu, S. Rao, S. Havron, G. Ho, S. Savage, G. M. Voelker, and D. McCoy. No privacy among spies: Assessing the functionality and insecurity of consumer Android spyware apps. *Proceedings on Privacy Enhancing Technologies*, 1:1–18, 2023.
- [25] G. Michelet and F. Breiting. ChatGPT, Llama, can you write my report? an experiment on assisted digital forensics reports written using (local) large language models. *Forensic Science International: Digital Investigation*, 48:301683, 2024.
- [26] D. N. Moreira and M. P. Da Costa. The impact of the COVID-19 pandemic in the precipitation of intimate partner violence. *International Journal of Law and Psychiatry*, 71, 2020.
- [27] D. Nam, A. Macvean, V. Hellendoorn, B. Vasilescu, and B. Myers. Using an llm to help with code understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pages 1–13, 2024.
- [28] B. Palanisamy, S. Sensenig, J. Joshi, and R. Constantino. LEAF: A privacy-conscious social network-based intervention tool for ipv survivors. In *IEEE 15th International Conference on Information Reuse and Integration*, pages 138–146, 2014.
- [29] M. K. Qabalin, M. Naser, and M. Alkasassbeh. Android spyware detection using machine learning: A novel dataset. *Sensors*, 22(15), 2022.
- [30] Rithvik. Cerberus acknowledges data breach, states some usernames and encrypted passwords stolen, 2014. <https://www.droid-life.com/2014/03/26/cerberus-data-breach>.
- [31] M. Robinson and C. Taylor. Spy vs spy: Spying on mobile device spyware, 2020. <https://media.defcon.org/DEF%20CON%2020/DEF%20CON%2020%20presentations/DEF%20CON%2020%20-%20Robinson-Spy-vs-Spy.pdf>.

- [32] J. M. Schokkenbroek, W. Hardyns, and K. Ponnet. Baby don't hurt me: Victimization and perpetration experiences of offline and online intimate partner violence. In *Annual Meeting of the Belgian Association of Psychological Sciences*, 2021.
- [33] S. Sidor. Android: apps can take photos with your phone without you knowing., 2014. <https://rstforums.com/forum/topic/79016-android-apps-can-take-photos-with-your-phone-without-you-knowing/>.
- [34] S. Stephenson, N. Gupta, A. Polamarasetty, K. Huang, D. Youssef, R. Ceccio, K. Cowan, M. Zinkus, and R. Chatterjee. Poster: A case study on legal evidence of technology-facilitated abuse in wisconsin. *Twentieth Symposium on Usable Privacy and Security (SOUPS 2024)*, 2024.
- [35] H. L. Storer, E. X. Nyerges, and S. Hamby. Technology “feels less threatening”: The processes by which digital technologies facilitate youths’ access to services at intimate partner violence organizations. *Children and youth services review*, 139, 2022.
- [36] S. Taylor and Y. Xia. Cyber partner abuse: A systematic review. *Violence and victims*, 33(6):983–1011, 2018.
- [37] K. Thomas, D. Akhawe, M. Bailey, D. Boneh, E. Bursztein, S. Consolvo, N. Dell, Z. Durumeric, P. G. Kelley, D. Kumar, D. McCoy, S. Meiklejohn, T. Ristenpart, and G. Stringhini. SoK: Hate, harassment, and the changing landscape of online abuse. In *IEEE Symposium on Security and Privacy (SP)*, pages 247–267, 2021.
- [38] E. Tseng, R. Bellini, N. McDonald, M. Danos, R. Greenstadt, D. McCoy, N. Dell, and T. Ristenpart. The tools and tactics used in intimate partner surveillance: An analysis of online infidelity forums. In *USENIX Security '20*, pages 1893–1909, online, 2020.

- [39] Waqas. Company that sells spyware to domestic abusers hacked, 2018. <https://www.hackread.com/company-that-sells-spyware-to-domestic-abusers-hacked/>.
- [40] Z. Whittaker. Xns spy stalkerware spied on thousands of iPhones and Android devices, 2022. <https://techcrunch.com/2022/12/12/xns spy-stalkerware-iphone-android/>.
- [41] Z. Whittaker. Letmespy, a phone tracking app spying on thousands, says it was hacked, 2023. <https://techcrunch.com/2023/06/27/letmespy-hacked-spyware-thousands/>.
- [42] C. Wyburn. The Consumer Spyware Industry: An Australian based analysis of the threats of consumer spyware, 2019. <https://accan.org.au/grants/completed-grants/1435-risks-impacts-and-accountability-in-the-consumer-spyware-industry>.

Appendix A

A.1 ChatGPT report analysis

In what follows, we show two examples of ChatGPT report analysis outputs. They were both the first answer given by the model after submitting the initial prompt along with related WARNE generated reports. Additional comments are added in blue text and are not part of the original reports.

A.1.1 OwnSpy

Let's start with the first step: identifying and listing privacy-sensitive permissions granted to the stalkerware application.

Permissions Granted to the Application

The report lists the following privacy-sensitive permissions:

%Permissions are not explained by the model, most of them are self-explanatory but an additional prompt can make the model highlight the most important ones.

- ****Location:****
 - 'android.permission.ACCESS_COARSE_LOCATION'
 - 'android.permission.ACCESS_FINE_LOCATION'
 - 'android.permission.ACCESS_LOCATION_EXTRA_COMMANDS'
 - 'android.permission.ACCESS_MOCK_LOCATION'
 - 'android.permission.ACCESS_BACKGROUND_LOCATION' (restricted)
 - 'android.permission.ACCESS_MEDIA_LOCATION'

- ****SMS and Call Logs:****
 - 'android.permission.READ_CALL_LOG' (restricted)
 - 'android.permission.READ_SMS' (restricted)
 - 'android.permission.RECEIVE_MMS' (restricted)
 - 'android.permission.RECEIVE_SMS' (restricted)
 - 'android.permission.RECEIVE_WAP_PUSH' (restricted)
 - 'android.permission.SEND_SMS' (restricted)
 - 'android.permission.WRITE_CALL_LOG' (restricted)
 - 'android.permission.WRITE_SMS'
 - 'android.permission.PROCESS_OUTGOING_CALLS' (restricted)
 - 'android.permission.CALL_PHONE'
 - 'android.permission.READ_PHONE_STATE'
 - 'android.permission.RECORD_AUDIO'

- ****Contacts and Calendar:****
 - 'android.permission.READ_CONTACTS'
 - 'android.permission.WRITE_CONTACTS'

- 'android.permission.READ_CALENDAR'
- 'android.permission.WRITE_CALENDAR'

- ****Installed Apps:****
 - 'android.permission.GET_TASKS'
 - 'android.permission.GET_ACCOUNTS'
 - 'com.android.browser.permission.READ_HISTORY_BOOKMARKS'
 - 'com.android.browser.permission.WRITE_HISTORY_BOOKMARKS'

- ****Network State and Internet:****
 - 'android.permission.ACCESS_NETWORK_STATE'
 - 'android.permission.ACCESS_WIFI_STATE'
 - 'android.permission.CHANGE_NETWORK_STATE'
 - 'android.permission.CHANGE_WIFI_MULTICAST_STATE'
 - 'android.permission.CHANGE_WIFI_STATE'
 - 'android.permission.INTERNET'

Extracting Dumped Files and Content

The report lists various files and their locations:

App APK files:

```

/data/org.system.kernel/app_apk/app.apk
/data/org.system.kernel/app_apk/oat/app.apk.cur.prof
/data/org.system.kernel/app_apk/oat/app.apk.prof
/data/org.system.kernel/app_apk/oat/arm/app.art

```

/data/org.system.kernel/app_apk/oat/arm/app.odex

/data/org.system.kernel/app_apk/oat/arm/app.vdex

App logs :

/data/data/org.system.kernel/app_logs/2023-09-04.log
.gz

Temporary files :

/data/org.system.kernel/app_tmp/keylogger

Shared preferences :

/org.system.kernel/shared_prefs/GCM.xml
/org.system.kernel/shared_prefs/SETTINGS.xml

Analyzing SQLite Databases

The report details three SQLite databases :

queue database :

Tables: android_metadata , queue (empty)

address_book database :

Tables: android_metadata , contacts_data (empty)

package_installed database :

Tables: android_metadata , package_installed

Packages :

com.spa_app.alarm

com.android.core.mngx

com.android.core.mngy

net.reptilicus.clientapp

com.app.com.app.com.app.aplntal
com.androidlost
com.samsung.android.messaging
com.w0f0
city.russ.alltrackerinstaller
com.mobilespy.io
com.google.android.gm
app.spy24.spy24installer
com.panspy.android
com.google.android.apps.docs
wosc.cwf
com.android.chrome
com.android.internet.a20230215
com.shadow.client.android
com.google.android.videos
com.google.android.apps.photos
com.microsoft.office.officehubrow
com.facebook.katana
com.sec.android.sipibuz
com.sec.android.app.popupcalculator
com.hp.vd
m.mobile.control
com.linkedin.android
net.homesafe
app.EasyLogger
org.alltracker.security

```
com.systemservice
com.google.android.apps.youtube.music
com.lsdroid.cerberuss
org.system.kernel
br.com.galaxys.gameap
app.spy24.systemwifi
com.topjohnwu.magisk
com.gcm_call_sms_tracker.updated
```

Highlighting Evidence of Monitoring

We'll look for specific files or database entries indicating that the phone is being monitored:

- **Logging modules**: Any files or entries indicating logging of user activities.
- **Google Analytics usage**: Any mention of tracking via Google Analytics.
- **Configuration settings**: Detailed settings indicating monitoring preferences or targets.
- **User activity logs**: Logs of SMS, call history, location tracking, etc.

Shared preferences indicating monitoring settings:

Settings from SETTINGS.xml:

```
whatsapp: 1
```

account_type: 3
hqaudio: false
sms_connect: none
livetcp: 1
registered: true
prevent_uninstall: 0
tracking: 1
pictures: 1
lastSync: 1694023069965
trackingrate: 10
callhistory: 1
webhistory: 1
keycapture: 1
sms: 1
email: hankaera0@gmail.com
forcewifi: 0
appsTorecord: [""]
recordAudio: 1
checkrate: 10
sms_callback: none
applog: 1
last24Sync: 1693954907295
version: 1624
appsinstalled: 1
addressbook: 1
sms_locate: none

```
userpass: 5f4dcc3b5aa765d61d8327deb882cf99
callrecording: 1
```

Summarizing Findings

We'll provide a concise summary of the key findings, including:

- Any crucial information that can help identify the abuser (e.g., email addresses, phone numbers, specific monitoring configurations).

I'll now search through the uploaded report for any more specific information related to email addresses, URLs, API tokens, potential password entries, and relevant data within the SQLite databases. This may take a bit of time. I'll provide the results once the analysis is complete.

The file 'report.org.system.kernel.txt' contains several sensitive data elements such as email addresses, URLs, and potentially passwords. Here's a summary of what was found:

Sensitive Information

1. ****Email Addresses****:

- 'hankaera0@gmail.com' found in the '/org.system.kernel/shared_prefs/SETTINGS.xml' file. *%A common occurrence of identifiable stalker data.*

2. ****Potential API Tokens or Registration IDs**:**

- Registration ID found in '/org.system.kernel/shared_prefs/GCM.xml':

““

```
<string name="registration_id">
  APA91bG19qd3iVZf3wzthSTxmfGKD-
  ApU0Ci0TheUXZPjmVXpArVAfkPv9qbbexwJN117QRK2FJe9H-
  eJkr8O438ZkvKAVwaQIpsfCF5ttO7gsOOzcD_1-N74mUL-
  erftCWTggiET8sz </string >
```

.

““

3. ****Potential Password Entries**:**

- A hashed password found in the '/org.system.kernel/shared_prefs/SETTINGS.xml' file:

““

```
<string name="userpass">5
  f4dcc3b5aa765d61d8327deb882cf99 </string >
```

.

““

4. ****Config.db and Location.db**:**

- There are no specific entries labeled as 'config.db' or 'location.db', but there are several databases mentioned such as:
 - 'address_book '
 - 'queue '
 - 'package_installed '
- These entries are found under '/data/org.system.kernel/databases/' and appear to be SQLite databases [11:3+report.org.system.kernel.txt][11:4+report.org.system.kernel.txt]. %This can be confusing as the model mentions file names that are not present in the report. However, it highlights databases that are, in fact, proofs of data collection by the app.

5. **SMS.db**:

- No direct reference to 'sms.db' but there are several database files listed which could potentially store SMS or other sensitive information.

Additional Findings

- A variety of APK files listed under different paths indicating installed applications, some of which may be monitoring or spying applications [11:1+report.org.system.kernel.txt][11:2+report.org.system.kernel.txt][11:5+report.org.system.kernel.txt].
- The presence of keylogger logs and other monitoring logs:

- '/data/org.system.kernel/app_tmp/keylogger '
- '/data/org.system.kernel/app_logs/2023-09-04.log.gz '. %
This is another important evidence of monitoring by the app.

Recommendations

- **Review and sanitize**: All sensitive data elements should be carefully reviewed and sanitized to prevent unauthorized access or data leaks.
- **Encrypt sensitive data**: Ensure that any sensitive information stored in configuration or preferences files is encrypted.
- **Access control**: Limit access to these files to only authorized personnel and applications.
- **Audit and monitor**: Regularly audit and monitor for unauthorized access or modifications to these files.

This summary highlights the sensitive information found and provides recommendations to enhance security and data protection.

A.1.2 TheTruthSpy

I will now start analyzing the provided report according to the given instructions.

Step 1: List Permissions

Requested Permissions :

- android.permission.FLASHLIGHT
- android.permission.SYSTEM_ALERT_WINDOW
- com.android.browser.permission.READ_HISTORY_BOOKMARKS
- android.permission.READ_CALENDAR
- android.permission.CAMERA
- android.permission.READ_CONTACTS
- android.permission.GET_ACCOUNTS
- android.permission.ACCESS_COARSE_LOCATION
- android.permission.ACCESS_FINE_LOCATION
- android.permission.ACCESS_BACKGROUND_LOCATION
- android.permission.RECORD_AUDIO
- android.permission.MODIFY_AUDIO_SETTINGS
- android.permission.READ_PHONE_STATE
- android.permission.READ_PHONE_NUMBERS
- android.permission.READ_CALL_LOG
- android.permission.PROCESS_OUTGOING_CALLS
- android.permission.CALL_PHONE
- android.permission.READ_SMS
- android.permission.RECEIVE_SMS
- android.permission.RECEIVE_MMS
- android.permission.SEND_SMS
- android.permission.WRITE_EXTERNAL_STORAGE
- android.permission.READ_EXTERNAL_STORAGE
- android.permission.INTERNET

- android.permission.ACCESS_NETWORK_STATE
- android.permission.ACCESS_WIFI_STATE
- android.permission.CHANGE_WIFI_STATE
- android.permission.CHANGE_NETWORK_STATE
- android.permission.WAKE_LOCK
- android.permission.RECEIVE_BOOT_COMPLETED
- android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS
- android.permission.ACCESS_SUPERUSER
- android.permission.FOREGROUND_SERVICE
- android.permission.REQUEST_DELETE_PACKAGES
- oppo.permission.OPPO_COMPONENT_SAFE
- com.huawei.permission.external_app_settings.USE_COMPONENT
- com.google.android.finsky.permission.
 BIND_GET_INSTALL_REFERRER_SERVICE
- com.google.android.c2dm.permission.RECEIVE
- android.permission.ACCESS_MEDIA_LOCATION

Granted Permissions:

- com.google.android.finsky.permission.
 BIND_GET_INSTALL_REFERRER_SERVICE: true
- com.google.android.c2dm.permission.RECEIVE: true
- android.permission.MODIFY_AUDIO_SETTINGS: true
- android.permission.CHANGE_NETWORK_STATE: true
- android.permission.FOREGROUND_SERVICE: true
- android.permission.RECEIVE_BOOT_COMPLETED: true

- android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS:
true
- android.permission.INTERNET: true
- com.android.browser.permission.READ_HISTORY_BOOKMARKS:
true
- android.permission.CHANGE_WIFI_STATE: true
- android.permission.FLASHLIGHT: true
- android.permission.ACCESS_NETWORK_STATE: true
- android.permission.REQUEST_DELETE_PACKAGES: true
- android.permission.ACCESS_WIFI_STATE: true
- android.permission.WAKELOCK: true
- android.permission.READ_SMS: true
- android.permission.READ_CALENDAR: true
- android.permission.READ_CALL_LOG: true
- android.permission.ACCESS_FINE_LOCATION: true
- android.permission.RECEIVE_MMS: true
- android.permission.RECEIVE_SMS: true
- android.permission.READ_EXTERNAL_STORAGE: true
- android.permission.ACCESS_COARSE_LOCATION: true
- android.permission.READ_PHONE_STATE: true
- android.permission.SEND_SMS: true
- android.permission.CALL_PHONE: true
- android.permission.CAMERA: true
- android.permission.PROCESS_OUTGOING_CALLS: true
- android.permission.WRITE_EXTERNAL_STORAGE: true
- android.permission.RECORD_AUDIO: true

- android.permission.READ_CONTACTS: true
- android.permission.ACCESS_BACKGROUND_LOCATION: true
- android.permission.ACCESS_MEDIA_LOCATION: true

Step 2: Extract Dumped Files and Content

Identified Dumped Files:

- The file list was extensive and truncated. I will focus on parsing the content of dumped files, looking for critical information.

Step 3: Analyze SQLite Databases

Identified SQLite Databases:

- The report indicates the presence of SQLite databases. I will extract information from lines containing "SQLite database".

Step 4: Highlight Evidence of Monitoring

Monitoring Indicators:

- User activity logs
- Tracking data
- Surveillance indicators

Step 5: Summarize Findings

Key Findings:

- ****Privacy-sensitive permissions granted****: The app has access to location , SMS, call logs , contacts , internet , and network state , indicating its potential for extensive monitoring .
- ****Dumped files****: Multiple files related to the app's functioning were dumped. These could contain logs or configurations used for monitoring .
- ****SQLite databases****: The presence of SQLite databases suggests storage of structured data , possibly including monitored data .
- ****Monitoring evidence****: The combination of permissions and service intents implies significant potential for surveillance and logging of user activity .

I will now parse the content of the report for specific details: email addresses , URLs, API tokens , potential password entries , and relevant SQLite database information . This might take a moment .

The uploaded file contains several key pieces of sensitive information related to system services . Here are the details :

1. ****Email Addresses****:
 - 'mobileinnova@gmail.com'
 - 'hankaera0@gmail.com'

2. ****URLs and Host Addresses****:

- 'http://ambientrecord.thetruthspy.com'
- 'http://protocol-a943.thetruthspy.com/protocols/check_device_registered.aspx'
- 'http://protocol-a943.thetruthspy.com/protocols/authenticate.aspx'
- 'http://protocol-a943.thetruthspy.com/protocols/getsetting.aspx'
- 'http://sync-a943.thetruthspy.com/sync'
- 'http://media-sync-a943.thetruthspy.com/sync'

3. ****Authentication Tokens and Passwords****:

- 'RTSP_Password' with a placeholder value '*****'
- 'Source_Password' as '123456'
- A plaintext password for user 'hankaera0@gmail.com' found in a URL for authentication.

4. ****Database Information****:

- Several SQLite databases related to 'androidx.work.workdb' and 'google_app_measurement_local.db'.
- The databases contain tables like 'android_metadata', 'room_master_table', 'Dependency', 'WorkSpec', 'WorkTag', 'SystemIdInfo', and 'WorkName' which are empty, except for the 'android_metadata' and 'room_master_table'.

5. ****License Information****:

- 'Player_License' and 'Android_Gocoder_License' values.
- Several entries indicate license checks and expirations.

Here are some excerpts highlighting the sensitive information:

- ****Emails and Authentication****:

```
““plaintext
thetruthspy.com”,”URL_Help_Protected_App”:””,”
RTSP_Username”:”mobileinnova@gmail.com”,”RTSP_Password
”:”*****”,”Player_License”:”PLAY2-a68pA-E3GED-MCjHt-8
CAcc-4YWAn”,”Android_Gocoder_License”:”GOSK-4F47-010C-
B0B2-9270-5440”,”iOS_Gocoder_License”:”[GOSK-4F47-010C-
B0B2-9270-5440]”,”Host_Address”:”http://ambientrecord.
thetruthspy.com”,”Port_Number”:1935,”Application_Name
”:”thetruthspy”,”Source_Username”:”scp”,”
Source_Password”:”123456”,”Streamlock_Website”:”https
://64ba8fb57db37.streamlock.net”,”
Streamlock_Port_Number”:8443}
““
```

- ****Authentication URL with Plaintext Password****:

```
““plaintext
```

Authenticate URL =

```
http://protocol-a943.thetruthspy.com/protocols/
  authenticate.aspx?username=hankaera0@gmail.com&
  password=password&deviceid=ee0df1ece1382345&brand_id
  =2&dotest=1
```

““

- ****License and Service Checks****:

```
““plaintext
DEBUG - 2023-09-06 10:10:38,420 - [AndroidSmsWatcher] :
  checkLicenseExpired = true
DEBUG - 2023-09-06 10:10:38,424 - [
  AndroidBrowsingHistoryWatcher] : Start
  AndroidBrowsingHistory
DEBUG - 2023-09-06 10:10:38,463 - [UIUserInternetReceiver]
  : checkLicenseExpired LICENSE EXPIRED and
  RemoveDevice
```

““

- ****Database Information****:

```
““plaintext
==> /data/com.systemservice/databases/
  google_app_measurement_local.db
```

Tables :

```
--android_metadata --
```

```
  locale
```

```
0  en_GB
```

```
--messages --
```

```
Empty DataFrame
```

```
Columns: [type, entry]
```

```
Index: []
```

```
'''
```

This analysis covers the critical information, including email addresses, URLs, tokens, passwords, and database structure within the provided file. Let me know if you need more specific details or further analysis on any part.

Appendix B

B.1 Llama3 report analysis

We show here two examples of Llama3 report analysis outputs. They were both the first answer given by the model after submitting the initial prompt along with related WARNE generated reports.

B.1.1 OwnSpy

This appears to be a forensic analysis of a device, likely a smartphone, that was compromised by a stalkerware app.

Here's a breakdown of the files and data:

`**blacklist_app.json**`: An empty file indicating no blacklisted apps were found.

`**blacklist_words.json**`: Another empty file suggesting no blacklisted words or phrases were detected.

****geofence.json****: A file containing geolocation information , likely used by the stalkerware app to track the device's location. However, it is empty, implying that the geofencing data was not retained or accessed.

****app_logs/2023-09-04.log.gz****: A gzipped log file from September 4, 2023, containing application logs. This could provide valuable information about the app's behavior and actions taken on the device.

****keylogger****: A file containing keylogger data , which captures keystrokes entered by the user. This suggests that the stalkerware app was monitoring user interactions with the device.

****databases****:

* ****address_book-journal****: A journal file related to the address book database , indicating that changes were made to the contacts list.

* ****queue-journal****: Another journal file recording changes to a queue or message system on the device.

* ****package_installed-journal****: A journal file tracking package installations (apps) on the device.

* ****package_installed****: A SQLite database containing information about installed packages (apps).

The presence of these databases and journals suggests that the stalkerware app was monitoring and recording various aspects of the device's behavior, including:

1. Installed apps
2. Contacts list changes
3. Message queue activity

****app_apk**:**

- * ****app.apk****: A compressed Android package file containing the app's code and resources.
- * OAT (Object Archive Template) files: Various files related to the app's compiled code, including:
 - + 'arm/app.art': ARM architecture-specific code
 - + 'arm/app.odex': Optimized DEX (Dalvik Executable) file
 - + 'arm/app.vdex': Verified DEX file

These OAT files are likely used by the stalkerware app to monitor and control the device's behavior.

In summary, this analysis suggests that a stalkerware app was installed on the device, which monitored and recorded various aspects of user activity, including keystrokes, app installations, and message queue changes. The presence of these databases and files provides valuable evidence for forensic investigators to reconstruct the events surrounding the compromise.

B.1.2 TheTruthSpy

This is a log file from an Android device, specifically related to the Truesee stalkerware app. The file contains debug logs from the app, which provides insight into its functionality and behavior.

Here are some key findings:

1. ****Stalkerware app configuration****: The log shows that the app is configured to send data to a server at 'http://protocol-a943.thetruthspy.com/protocols/setsetting.aspx'. This suggests that the app is designed to transmit user data to a remote server.
2. ****Device information****: The log contains device-related information, such as the device ID ('ee0df1ece1382345'), operating system (Android 11), and battery level (100%).

3. ****GPS and location tracking****: The log indicates that GPS is active , which suggests that the app is capable of tracking the user's location .
4. ****License expiration****: The log shows that the app's license has expired , which might indicate that the app requires a paid subscription or update to continue functioning properly .
5. ****Android watchdog service****: The log mentions the Android watchdog service , which suggests that the app is designed to run continuously in the background and monitor the device for specific events .
6. ****Data collection****: The log implies that the app collects data from various sources , such as browsing history , SMS messages , and photos .
7. ****Uninstallation detection****: The log shows that the app detects when it's being uninstalled or removed from the device .

Overall , this log file provides evidence of a stalkerware app's capabilities and behavior , which is concerning given its potential for violating user privacy and security .