Security and Privacy Analysis of Employee Monitoring Applications

by

Adam Campbell

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2023

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Workplace surveillance is not a new issue; however, recently there has been increasing adoption of Employee Monitoring Applications (EMAs) that observe employees' digital behaviour. This trend was advanced by the increase of remote work due to the COVID-19 pandemic and the ease of deployment of EMAs with the accelerating cloud computing industry. EMAs allow employers to monitor their workers' behaviours remotely, resulting in privacy concerns.

EMAs use highly privileged functions to achieve their features, such as web browsing monitoring, key-logging, microphone monitoring, webcam monitoring, and remote takeover of the device. EMA vendors claim to protect company security and employee privacy. Our research challenge is to assess how well the vendors uphold their claims of protecting security and privacy.

We develop a framework to assess security and privacy issues related to EMAs. Our framework applies dynamic and static analysis techniques to ten popular Windows EMAs. EMAs typically have a monitoring app, which is installed on an employee computer. The app collects and sends data to the backend server, which aggregates the data and displays it in a dashboard. The employer has access to the dashboard to view the collected data and configure monitoring settings.

Our app-centred analysis is focused on issues such as insecure data transmissions, lack of certificate pinning, residual vulnerabilities after app un-installation, security vulnerabilities due to use of a proxy, anti-keylogging, conforming to Windows privacy permissions, effectiveness of EMA privacy features, and determining a general monitoring profile. The app-centred analysis informs us whether EMAs are secure at the local and network levels. We also assess whether EMAs uphold their promises in regards to privacy.

Our backend analysis focuses on issues like password security, lack of input validation, open cloud storage, insufficient access control, server geolocation, and insecure security configurations like no HSTS enforcement and out-of-date TLS versions. Analysing the backend infrastructure tells us on EMAs' vulnerability posture in regards to a remote attacker threat. We assess whether EMA vendors adequately protect the data they collect about employees.

Our analysis reveals a number of security and privacy vulnerabilities. These vulnerabilities include issues like data creep, where apps collect metadata about employees and their devices, but do not display this data on the dashboard to an employer. We also notice that one app does not use TLS for data transmission, so it sends private employee data over the public Internet for anyone to eavesdrop. One app offers a GDPR mode,

which claims to stop collecting highly sensitive data like web browsing history and screenshots. However, we see that this app still collects and sends web browsing history while this mode is turned on. Backend security misconfigurations we observe include open cloud storage, weak password requirements, lack of password guess rate limiting, and no HSTS enforcement.

Overall, we find that each app in our analysis is vulnerable to at least one threat we assess in our framework. Our study aims to provide data for legal analysis to assess the need for legal protections for employees against this kind of monitoring.

Acknowledgements

This thesis draws on research supported by the Social Sciences and Humanities Research Council.

I would like to express my gratitude to my supervisor, Urs Hengartner, for all his guidance, patience, and support throughout my program.

I would also like to thank the rest of the collaborators on the project, Understanding the Risks and Regulation of Workplace Surveillance in Canada's Digital Economy. In particular, I would like to thank Adam Molnar for leading and directing the project and Xavier de Carné de Carnavalet for help setting up mitmproxy and advice on how apps may collect browser history.

Finally, I would like to thank Yousra Aafer and Diogo Barradas for serving on my thesis committee and for providing insightful comments and feedback.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Employee monitoring has long been a controversial issue. From punch cards to CCTVs, now we have network monitoring and keystroke logging. This controversy is accelerated by advances in computing technology, which provide the ability for more invasive surveillance and easy-to-deploy solutions. Employee monitoring applications (EMAs) can now monitor employees while they are working remotely. EMAs are capable of keystroke logging, monitoring web usage, emails, and printings, screen recordings, and even observing the user's webcam. The shift towards remote work brought on by the COVID-19 pandemic has accelerated the adoption of EMAs [28]. Large companies have seen a 100% increase in adoption from the pre-pandemic number [30]. Employee monitoring has increased in both adoption and capabilities. However, there is little legal protection for employees from employer monitoring. In Ontario, employers can electronically monitor employees as long as they disclose the surveillance under the Working for Workers act [5]. Quebec, British Columbia and Alberta also require disclosure of monitoring under provincial privacy laws and workers in other provinces are left without legal protections [47].

Working remotely, our homes have become our office spaces. This blurs the line between work and home. Many use their devices for work and personal usage. This complicates the issue of privacy. Employees are in a vulnerable position relative to their employer. Employees may be informed of the monitoring but have little say in what aspects of their behaviour can be tracked. Workplace surveillance can widen the trust gap between employee and employer [48].

A survey reports that 2/3 of companies track internet usage, and 45% log keystrokes [55]. The EMA industry is robust, with a doubling in size from 2021 to 2030 [39]. These rapid increases in adoption and capabilities are not met with an increase in regulation. Many EMAs [**?**, **?**, **?**] claim that they aim to protect security and personal privacy. How well do they protect the security of the data they collect and adhere to their privacy policies?

Some commonly cited benefits of EMAs are project management, productivity, accountability, attendance, payroll, performance view, and data security [52]. One vendor claims that being monitored improves performance [**?**]. Insider threats are the largest risk to manage in enterprise security [1]. So, employers wish to monitor their employees so they can reduce the risk of an insider threat. Also, EMAs are increasingly easy to deploy with cloud-based solutions. Of course, EMAs also have disadvantages like the risk to personal privacy, the risk of data security, increasing distrust in employees, and breaking legal boundaries. Even though EMAs claim to improve data security, they are implicitly adding another layer in the technology stack, which will always come with another opportunity for an attack vector. EMAs must collect and store employee data in a secure manner so as not to risk breaching employee privacy.

In this thesis, we design and develop an analysis framework to assess security and privacy issues related to EMAs. We choose this domain of applications due to the accelerating nature of the industry. The industry is accelerating because of the increasing trend towards remote work and the ease of deploying EMAs in the cloud. These apps are designed to help managers and employers watch their employees for insider threats, improve productivity, and help with business administration such as payroll or attendance. EMAs use invasive surveillance techniques like screen recording and keystroke logging to achieve this end. Our research question is: *how well do EMA vendors protect the data they collect about employees?* We detail a list of security and privacy issues that are important for protecting the security and privacy of employee data (i.e., proxy vulnerabilities and backend security configurations). We apply static and dynamic analysis techniques to assess 10 Windows EMAs with a variety of tools. We choose Windows as our analysis platform due to its prevalence in workplace settings.

1.2 Contributions

Our contributions are as follows:

1. We develop an analysis framework to assess security and privacy issues related to EMAs. We apply static and dynamic analysis techniques. We intercept the apps' web

traffic to understand what data the apps are collecting and whether it conforms with the app's claims. For our app-centred approach, we assess each app for insecure data transmission, residual vulnerabilities, whether it raises malware alarms, check for proxy vulnerabilities, and assess the efficacy of privacy features. For our back-end analysis, we assess password strength, password brute force attack protections, input validation, open cloud storage, access control of private media, server geolocation, TLS versions, and HSTS enforcement.

2. We assess EMAs for security and privacy issues, which is the first technical analysis of EMAs. We apply our framework to 10 Windows EMAs. We find that each app is vulnerable to at least one of the privacy and security issues we assess.

3. We provide novel data sets to social and legal teams to clarify discrepancies between how EMAs work and what is permissible under Canadian law.

4. We find a number of vulnerabilities in EMAs. We find 3/10 of apps have weak password requirements. So, password brute-force attacks on these apps are computationally easier. 2/10 of apps have no password brute-force protections. An attacker can try passwords repeatedly without penalty. 1/10 of apps do not encrypt network traffic, so private employee data passes through the public internet for anyone to see. 9/10 of apps lack HSTS enforcement, so users visiting their site are vulnerable to SSL-stripping attacks. 7/10 of apps do not inform the user of suspicious activities on the account. This allows attackers to compromise accounts more conveniently. 1/10 of apps have known vulnerabilities with their web proxy, compromising network security. 5/10 of apps have improper access control to private media, so those without credentials can access the user's private data. 4/10 of apps have open cloud storage, so anyone with the URI can access the information stored there.

1.3 Thesis Structure

This thesis is organised as follows: In Chapter 2, background knowledge is provided for the rest of this thesis. We discuss the security and privacy issues of EMAs, our threat model, and related work. In Chapter 3, we propose the analysis framework used to assess EMAs. We apply static and dynamic analysis techniques at both the application and back-end levels of the technology stack. In Chapter 4, we present the experimental results of our app-centred analysis. In Chapter 5, we present the experimental results of back-end analysis. Lastly, in Chapter 6, we provide discussion, limitations, recommendations for developers, and our concluding remarks.

Chapter 2

Background

In this chapter we will discuss the concepts, techniques and questions we will apply to our analysis of employee monitoring applications. We will borrow high level strategies from malware analysis work, and use research questions from security and privacy analysis papers. Both types of papers will suggest techniques and strategies we can apply to the analysis of employee monitoring apps.

2.1 Malware Analysis and Reverse Engineering

Malware analysis typically involves reverse engineering techniques to peel back the layers of complexity to understand how malware works. These techniques fall into two broad categories: static analysis and dynamic analysis. Static analysis is an analysis of the program without it running. Opening and reading the program's code is a static analysis technique. Dynamic analysis is the analysis of a program while it is running. Network analysis is a dynamic analysis technique where network traffic coming from the program is logged and analysed. Analysis techniques and strategies introduced in malware analysis can be applied in the context of security and privacy analysis. We analyse the state-of-the-art in malware analysis to borrow techniques to analyse EMA's privacy and security issues.

2.1.1 Low-level Malware Analysis

Carnavalet et al. [17] show the security and privacy risks of the adware, Wajam. The authors study the adware for six years and evaluate the anitivirus evasion techniques used by Wajam. They demonstrate an infected user is subject to plaintext leaks of browser histories and keyword searches. Their paper provides analysis on how Wajam has reached widespread use. Adware was considered the same as spyware, but now there is a change to tolerate adware, and it is now considered "optional", or "not a-virus". Malware analysis research has not focused much on adware in recent years. This paper addresses three research questions: 1) Is Wajam simply displaying untargeted advertisements? 2) Does Wajam pose any serious security and privacy threats? 3) Are all strains of Wajam limited in complexity and reliably detected by antiviruses? Wajam has been installed hundreds of millions of times. The authors evaluate 52 samples from Wajam, which were released over a six-year period. They reverse engineer the samples to characterise how the Wajam product has evolved over the years. The Wajam product used 4 different techniques for content injection, 23 techniques for AV evasion, and 332 domains to serve injected scripts. Their analysis shows the important PII (Personal Identifiable Information) leakage and security risks. Wajam uses a rootkit to hide itself. The authors show the privacy leaks and security risks. Wajam leaks unique identifiers during installation. Wajam was shown to leak the files installed on the OS, the browsing history, and whether the program was running in a VM. Wajam looks for the presence of major antivirus software and sends out a list of installed AV software. The authors describe the update mechanism of Wajam and the security risks involved. Security risks include: downgraded TLS from proxies, private key generation and common name generation, downgrading website security by removing CSP headers. Their analysis method involves two main steps: 1) Download Wajam on a fresh VM snapshot with ProcMon and Wireshark running. They snapshot the filesystem and registry. 2) Debugging: they set breakpoints at Windows API calls to load files and use I/O events from ProcMon to identify relevant functions from the call stack at that time. They find that Wajam collects and leaks large amounts of data about the user. Wajam also displayed antivirus evasion strategies.

Knockel et al. [36] examine the security risks of the QQ browser by Tencent. Their threat model is based on a MITM attacker with nation-state control over the network. They describe and provide examples of three classes of threats to QQ. The authors use reverse engineering to examine the encryption protocols used by the browser. They find a significant security risk in the browser's encryption scheme. There is no padding added to the ciphertext before encryption. The authors describe three sets of attacks. The first set is based on exploiting QQ's poor random number generation, hard-coded symmetric

keys, and use of a 128-bit RSA key in earlier versions. The MITM threat would be able to decrypt and read all secure communication with passive, offline eavesdropping. The second set of attacks is based on the use of textbook RSA. An attacker can crack the session key with 128 connections of their own to the QQ server. This attack can guess the session key, one bit at a time. This attack is not a passive, offline attack and only scales to breaking the encryption of one user's session. The third category of attacks involves an attacker gaining full access to a client device through a MITM. The attacker exploits the update system to replace the APK and hash with their own malware instead of the update. QQ sends WUP updates regularly to QQ servers with IMEI, WiFI MAC address, URLs of webpages visited, and more sensitive data. The only entropy source for choosing a session key scheme is the time since epoch. The session key is the 128 least significant bytes in the decrypted plaintext. The authors present a novel attack against textbook RSA. The downside is that the security of millions of people has not been adequately protected by Tencent developers. The paper does not include details on how they performed the reverse engineering. What tools did they use? What were the steps in this analysis? The authors seem to gloss over this aspect of their findings. This lack of detail makes the paper less useful if one wants to repeat their experiments. If we find that EMAs use the same method of updating, they may face a similar vulnerability as the QQ browser.

2.1.2 High-level Malware Analysis

Egele et al. [21] examine techniques and tools used to help us understand the behaviour of malware. Malware is using defences such as self-modifying code that will make signature detection difficult with static analysis alone. The authors discuss tools and techniques that run the malware and monitor its behaviour. Most dynamic analysis tools (DAT) will monitor API and system calls. Function calls can be categorised based on the parameters passed in the function call. Some DATs monitor how sensitive data is collected and sent within and beyond the malware-infected device. Automated dynamic analysis reports actions that the malware took while under observation. Such reports can be used to group malware into families based on similar behaviours. Novel malware can be identified when its behaviour does not fit into the previously identified malware families. This is an improvement over static analysis, which may not be able to detect signatures if attackers generate many new malware samples with polymorphic encodings and binary packers. This paper introduces several techniques. One such technique is hooking. Hooking is the process by which function calls are intercepted. The Windows native API sits between the system call interface and the Windows API. Often, the native API is called by the Windows API. Implementing function hooking involves several steps: 1) If the source code is available,

calls to hook functions can be inserted in the source code at appropriate locations. 2) If the program is only available in binary, binary rewriting is used to insert hook function invocations. 3) Rewrite the monitored function to invoke the hook before normal operation of the function. 4) Modify all locations of the "call" instruction that will invoke the call to the hook instead of normal execution. The Detours library [29] helps with function hooking. This program redirects control from the hook function to the analysis function. It can modify binary before it is loaded, or it can also manipulate binary while it is loaded in memory. An instrumented debugger can also be used by placing breakpoints at the call site or the monitored function. Then the memory contents and CPU state will be visible. Function parameter analysis allows for the correlation of function calls operating on the same object. Information flow tracking is another technique that operates by establishing taint sources and taint sinks. Then we can create direct data dependencies where taint is transferred directly for assignments and arithmetic operations with the tainted value. Address dependencies are made when taint is transferred if the tainted value is referenced as a pointer or as an array index. Control flow dependencies find the first instruction (re-convergence point) executed regardless of which branch is taken. Before re-convergence is reached, each target of an assignment is tainted. The paper describes implementation techniques for malware analysis. Emulated environments allow the analysis component to control every aspect of program execution and the ability to run malware without fearing how it impacts the system. VM-based analysis offers the privileged state of the physical machine, which is not reachable from the VM (analysis environment). Network simulation offers no internet but a simulated network, and allows filtered internet access. The authors describe the malware analysis arms race in which malware creators use self modifying code, packers, detection environments, and logic bombs.

Galloro et al. [23] document 92 evasion techniques used by malware to avoid detection and create a taxonomy of these techniques. They evaluate 45K malware samples belonging to 2867 different malware families. They perform this evaluation over 10 years to document how evasion techniques change over time. They compare the evasion techniques of malware with legit Windows apps and software. They find a slight increase in the prevalence of evasive techniques. Such example techniques are: memory fingerprinting, which examines the memory location of a running process to find debuggers, exception handling, CPU fingerprinting; table descriptors; Traps, Timing, which identifies analysis by a slowdown in CPU clock ticks, Stalling, Registry, System environment, and human interaction, which assumes the environment is instrumented if the mouse is not moving. The malware may also process its environment by analysing the file system (i.e., a Python-based agent, agent.py in the home directory), listing processes, listing services, and listing drivers (emulated or virtualized devices).

Yong et al. [68] provide insight into the goals, workflow and thought processes of reverse engineers (REs). The authors broadly describe two types of analyses: Static analysis, which happens not during run time and dynamic analysis, which occurs during run time. The method the authors use is as follows: Recruitment of subjects, interview subjects about malware sources, analysis workflow, and dynamic analysis system configuration, and ask about data analyses. The authors find there are three categories of malware analysts: Tier 1: those who focus on string-based intrusion detection (i.e., hashes, IPs, and domain names) Tier 2: those who identify potential threats in network host artefacts or with tools. Tier 3: those who track TTPs: tactics, techniques, and procedures. The malware analysis workflow generally starts with dynamic analysis. The dynamic analysis set-up starts with the decision of bare metal vs. virtualization, and usually virtualization is used. Then the analyst will install the Web browser, Microsoft, Java, and Adobe Libraries. The analyst will mimic a real user with browser history, files, documents, directories, and usage. The analyst will configure settings such as libraries, timezone, language, usernames, and user privileges. The paper lists commonly used dynamic analysis monitoring tools and techniques such as ProcMon, hooking, syscalls, registry, files created, network activity monitoring, PCAP, and logs. This paper provides useful, high-level strategies to analyse EMAs. Although the paper is focused on malware, the strategies mentioned are effective in our analysis. The question we are focused on answering is different from typical malware analysis, however.

Votipika et al. [65] ask three research questions. Their first question asks what high-level process do REs follow when examining a new program? Second, what technical approaches (i.e., manual and automated analyses) do reverse engineers use? Finally, how does the RE process align with traditional program comprehension? How does it differ? Reverse engineering practises are different from traditional program comprehension because there is a lack of access to code and documentation and it occurs in an adversarial environment. The authors outline a three phase model used in reverse engineering: starting with an overview, going into sub-component scanning, and then performing focused experiments to answer specific hypotheses about the respective sub-component. During the overview phase, a reverse engineer will list strings and APIs, run a program with basic parameters, review metadata, and find specific functions to focus on. Then, in the subcomponent scanning phase, a reverse engineer scans beacons, comes up with specific hypotheses that require concrete information, and constructs data flow and control flow paths. Finally, in the focused experiments phase, a reverse engineer executes under inspection (debugger, file monitor), compares to the reference function (i.e., compare encryption outputs from the reference and in the program under investigation), and reads code line by line. The subcomponent scanning and focused experiment phases require a lot of back and forth,

to look at a subcomponent, create a hypothesis, and run experiments to evaluate their hypothesis. Some cross-phase trends are that the first two phases use static analysis, while phase 3 uses dynamic analysis. In phases 1 and 2, the REs choose a focus area. In phase 2, the RE recognises behaviours or vulnerabilities. The experience of reverse engineers guides where to look in the code. This paper provides guidelines for reverse engineering tool design, provides a framework to evaluate said tools, and gives insight on reverse engineering automation.

Botacin et al. [10] do static and dynamic analysis of 40k malware, over a 10 year period to track how financial malware in Brazil changes. They perform static analysis with SSDeep [32] to discard samples with repeated SHA1 values. They look for executables embedded in normal files with Foremost [45]. To extract portable executables, they use pyew [37] and peframe [7]. They use VirusTotal [53] to understand what malware family a sample belongs to. As for dynamic analysis techniques, they monitor changes to the registry, process creation and termination, and use tcpdump to capture network traffic.

2.1.3 Reverse Engineering Tools

In this subsection we discuss reverse engineering tools that represent the state-of-the-art in computer science research. We describe the tools to illustrate that there are a number of reverse engineering tools available but it is difficult to find one to suit our specific needs. Reverse engineering involves looking into very specific questions about the application in question, and often researchers develop tools to fit their use case but may be difficult to use in other applications. Instead we use well-known tools in our analysis. We use IDA Pro [26] for static analysis of application code, in particular when analyzing DLLs injected into a web browser process as we discuss in Sec. 3.2.9. For local dynamic analysis we use ProcMon [51].

PIITracker [8] can be used by reverse engineers to track PII data and capture any processes sending PII over the network. The techniques the authors use to accomplish this are, dynamic information flow tracking (DIFT) and monitoring specific functions and system calls. Their evaluations reveal that 12 of the 15 apps evaluated collect the users' PII in some form. The DIFT method uses two methods to track data. "Direct flows are data-flow based; indirect flows are control-flow-based. There are two direct flows: copy and computation dependencies. There are two types of indirect flows: address and control dependencies." The system plugs into PANDA [20], which is an open-source platform for dynamic analysis. PIITracker interacts with three plugins: taint2, syscalls2, and OSI/Win7x86intro. They use Windows API function calls and system calls as hooks, so

they get a callback when a specific function is called, with the callback containing the memory address. They taint the address with taint2. The authors have made this tool publicly available for download. The evaluations show how users' PII is not being kept safe by Windows application developers, who frequently gather and transmit the users' PII. PIITracker provides memory addresses for leaked PII and network socket data. However, it is difficult to understand how all the technology fits together. The authors do not explain how PIITracker fits into PANDA. It is not made clear what PANDA does or how it works. The PII that the authors investigate in this paper are (1) MAC address, (2) hard drive serial number, (3) hard drive model name, (4) volume serial number, (5) host name, (6) computer name, (7) security identifier number (SID), (8) CPU model, and (9) Windows version and build.

PyPANDA [15] provides a single script that controls both analysis and guest behaviour. Their interface provides full access to core emulator structures, and users can use generic images as starter templates. A common challenge is that the analysed system must be controlled manually while analysis scripts are run. PyPANDA addresses this challenge. Their design goals for PyPANDA are: high performance, providing unified analysis, easy integration with other tools, making C structures available as Python objects, and making them easy to use. The authors describe an example workflow for different use cases, such as dynamic heap monitoring, and provide the steps to do this analysis with PyPANDA. Overall, the tool provides an easy-to-use interface for PANDA. This tool appears to be useful for users of PANDA but does little to convince non-users of PANDA to try it.

PIITracker and PyPANDA both work as a plugin for PANDA but provide little background on what PANDA can be used for and how to implement the analysis.

2.2 Privacy and Security Analysis of Applications

Many applications today monitor user behaviour; this may be the purpose of the app or it may be a side effect. Apps with uses such as elderly care, e-scooter rentals, child monitoring, proctoring, and stalkerware have been studied to analyse the security and privacy issues raised by such apps. To our knowledge, this analysis has not been done on employee monitoring applications. So, we will pull useful research questions and analysis techniques to apply in our work.

Much of the work in this domain is done on the Android platform. Our analysis focuses entirely on Windows applications. So, exact techniques may not transfer to our work. However, we can adapt the techniques in order to answer our research questions.

Table 2.1: Summary of Security and Privacy Analysis Papers

Papers	Local Analysis Techniques	Network Analysis Techniques	Backend Analysis Techniques
Vinayaga-Sureshkanth et al. [64]	Android permissions requested, 3rd party identification, data flow analysis, control flow analysis	generic network monitoring	none
Ali et al. [6]	static analysis, identify 3rd party SDKs	tcpdump, mitmproxy, PII and secrets leakage, identify trackers, insecure authentication	password policy, online brute force attack, hsts enforcement, uniformed suspicious actvities, firebase scanner
Gruber et al. [25]	static analysis (permissions, libraries, SDKs), certificate pinning check	mitm, ip geolocation	SQL injection, access control, cloud enum, sslscan, Firebase scanner
Yang et al. [67]	API monitor, binary analysis	Microsoft Network Monitor	none
Burgess et al. [13]	encryption at rest, screenshots, virtual machine detection, clipboard management	encryption in transit, network access restrictions	none
Carnavalet et al. [16]	private key extraction	certificate validation, TLS versions, known attacks	none
Liu et al. [38]	manual analysis of each feature	tcpdump, mitmproxy	none
Kapoor et al. [33]	app permissions, 3rd party libraries,	TLS interception (BurpSuite), identify 3rd party trackers, API authentication	SQL injection, cross-site scripting XSS, firebase scanning

Common areas of analysis in this domain are local analysis, network analysis, and back-end analysis. Local analysis involves looking at the app itself and searching for security and privacy vulnerabilities. For example, static analysis of code is a local analysis technique. Network analysis of applications concerns the investigation of network traffic coming from the application under investigation. Intercepting encrypted traffic is a common technique that allows researchers to read what data the app is sending over the network. Researchers study the back-end infrastructure of applications to understand their security posture against potential attacks. A summary of the papers discussed in this section is viewable in Table 2.1.

2.2.1 Android

Kapoor et al. [33] look at the security and privacy issues in mobile apps marketed towards elderly populations. Their analysis includes local analysis, network analysis, back-end analysis, and privacy policy analysis. For static analysis techniques, they look at the app permissions and which third-party libraries an app uses. This informs them if the

app is using any suspicious permissions or third-party libraries. For network analysis, they use BurpSuite [49] for TLS interception, so they can read encrypted traffic coming from the apps. They analyse the network traffic to identify third-party trackers. They also modify the authentication headers in the network packets they collect to assess the security strength of the apps APIs. To analyse the backend of elderly apps, they try SQL injection and cross-site scripting to assess input validation. They use the Firebase Scanner to assess access control for the Firebase database. In this work, they also use an automated privacy policy analysis tool, Polisis [27], to assess these apps' privacy policies.

We also use TLS interception techniques but with mitmproxy [14] instead of BurpSuite. We also assess the input validation of the back-end against SQL injection attacks. As for database scanning, we use cloud_enum [43] instead of the Firebase scanner because Windows apps do not use Firebase as often as Android apps. We employ some of the same strategies, but our tools will differ because this work focuses exclusively on Android.

Liu et al. [38] investigate consumer spyware apps features. They investigate each feature to see how it works under the hood. This includes features like screenshots and hiding the app icon. They use static and dynamic analysis techniques for their investigation. This paper also looks at the data collected from these apps and how well it is protected. Some issues they look at are whether the data is encrypted in transit, cross-account request forgery, unauthenticated access to the victim's data, and data retention practises. They use tcpdump and mitmproxy in their network analysis to collect and intercept traffic coming from the spyware apps.

In our work, we assess the privacy issues this paper brings forth. We analyse network traffic with Wireshark to see if the data is encrypted in transit. We assess the risk of unauthenticated access to the user's data by looking at the URLs used to host the user's data. We assess if access to the URL is feasible (e.g., if the URL can be easily guessed) and whether it is accessible publicly. Finally, we also assess the EMA's data retention practises. When a user requests their data be deleted, we test whether media access becomes unavailable.

Vinayaga-Sureshkanth et al. [64] look at the privacy issues related to mobile e-scooter rental apps. They use local analysis techniques like static analysis to analyse the control flow of an app, permissions requested, and third-party libraries used. Dynamic analysis techniques are used to monitor how data flows through the device and onto the network. They use generic network monitoring techniques to capture the traffic leaving the device. This paper also analyses the privacy policies of these apps with automated and manual processing.

In our work, we also record what data EMAs collect about users, but we rely on network

traffic interception instead of monitoring API calls. The Android platform allows for insight into the Android API and permissions requested by applications. On the Windows platform, information on system calls and OS procedures are not accessible at the user level. Much of the low-level permissions are obfuscated from the user on Windows.

Gruber et al. [25] investigate privacy and security issues of child care apps. They use common local analysis techniques for Android analysis. Such techniques include using static analysis to see what permissions, third-party libraries, and SDKs are used by an application. For network analysis, they use tcpdump [57] for recording traffic and mitmproxy to intercept encrypted traffic. They record whether the app trusts a generic certificate from mitmproxy or whether this app is using certificate pinning. With traffic recorded from tcpdump, they geolocate all IPs the app communicates with. Their back-end analysis techniques are input validation testing, access control, TLS versions, checking for open cloud buckets, assessing access control security, and scanning Firebase.

Our work borrows their network analysis techniques for intercepting encrypted traffic, but we will use Wireshark instead of tcpdump for recording traffic. We employ most of their back-end analysis strategies, like testing input validation.

2.2.2 Windows

Yang et al. [67] use reverse engineering techniques to evaluate the security and privacy of video conferencing applications (VCAs). They use runtime binary analysis tools to achieve this end. With such tools, the authors trace raw audio in VCAs as it goes from the audio driver to the network. The authors develop a classifier that can infer what type of activity the user is doing based on the stats sent to the server. This classifier realises 82% accuracy across six categories of user activities. The authors analyse the mute button of VCAs for potential privacy risks. For analysis of the mute button in Windows, they use the system registry to track microphone access. Then they use the Windows API Monitor tool to instrument the userland API with hooks and log pointers to the inputs and outputs of microphone-related API calls. They use X64dbg [66] to read the contents of the buffer and Scylla-Hide [3] to hide the debugging from the app to prevent crashing. This paper [67] implements reverse engineering techniques across three platforms to assess the privacy strength of mute buttons in VCAs. This paper shows how answering the same question requires different techniques for each platform. Further, this paper shows how a user's privacy may be violated when they use a mute button. Even when the mute button is used, audio data is collected and sent over the network. Their classifier is able to use the audio data collected to infer what activity the user is doing.

We use static analysis and network monitoring, but the tools and techniques differ from this work. This work investigated specifically how the mute button works for VCAs, but in our work, we are building a more general profile of the security and privacy issues related to EMAs.

Ali et al. [6] propose a system to evaluate security and privacy risks in parental control software and hardware. The authors identify potential issues and develop a threat model. They develop a framework to evaluate parental control apps for security and privacy issues and apply this framework to a number of Windows and Android applications. This paper provides useful insight on security and privacy issues that affect monitoring applications. Their evaluations are conducted with eight network devices, eight Windows apps, ten Chrome extensions, and 29 Android apps. They analyse the tools on these platforms for leaking PII, insecure API authentication, and using third parties or known trackers. Their analysis reveals 135 vulnerabilities among the parental control solutions tested. Such solutions fail to protect the security and privacy of both the children and their parents. The parental control solutions are achieved with network monitoring, Android apps, Windows apps, and Chrome extensions. Network devices can use ARP spoofing to fool devices on the local network to confuse the network device for the local router. This way, the network device can read the outgoing messages and act as a MITM. The threat model includes an on-device attacker, a local network attacker, an on-path attacker (MITM), or a remote attacker who gains access through the backend. Their method uses dynamic analysis to see if the traffic is plaintext or encrypted and what data is sent. They use static analysis on binaries and scripts. They look in the binaries for APIs and URLs. In their dynamic analysis, they capture network traffic on the test device and the router using Wireshark and tcpdump. Their evaluation setup uses a full Linux distribution with mitmproxy and tcpdump installed in each environment with a Linux deployment. When using Android, they configure the network setting to proxy all traffic through the WiFi adapter, which routes it to the mitmproxy server. They analyse network traffic for PII and authentication leakage, improper access control, and identifying trackers. To identify improper access control, the authors use Postman [2] with requests to the API with stripped authentication headers to see if the API has weak authentication. They identify known trackers with Easy List, Easy Privacy, and Fanboy. The authors identify challenges in dynamic analysis, like network traffic attribution. Their solution is to use mitmproxy to call netstat to report the process name for each packet. They had challenges with traffic interception as well, where the apps were using certificate pinning to stop mitmproxy from intercepting the traffic. The authors use SSLunpinning [62] to overcome this challenge. The author's static analysis techniques involve identifying vulnerable services by scanning network devices with tools such as OpenVas [24], Nmap [40], Nikto [56], and Routersploit [60]. They

use the Firebase Scanner to detect security misconfigurations for apps that use Firebase. They use LibScout [19] to find third-party libraries embedded in the apps. They analyse the back-end of parental control apps by assessing the password rules, protecting against brute-force attacks, attempting SSLstripping attacks, recording responses to suspicious activities (like logins from new devices), and scanning the Firebase database. They also provide a framework that can be applied to similar apps to assess the security and privacy risks of those applications. Some of their techniques are applicable to Android apps, like SSLunpinning, which only works for Android apps, and Firebase Scanner, which is more effective with Android apps because of how prevalent the use of Firebase is with Android developers.

In this work, we also use mitmproxy [14] for network traffic interception, but instead of tcpdump, we use Wireshark. We also apply their techniques in back-end analysis: recording password rules, protection from brute force attacks, checking HSTS enforcement with SSLstrip attacks, and whether the apps inform users of new logins or password changes.

Carnavalet et al. [16] design a framework to analyse client-end TLS proxies that are seen in antivirus and parental-control software. This work highlights the risks created by such TLS proxies. They evaluate 14 antivirus and parental control apps (content control apps). They analyse whether the apps generate root certs dynamically and to what extent they protect the private keys using reverse engineering and deobfuscation techniques. They find flaws in the certificate validation process and test the TLS proxies against known attacks.

Applications that wish to monitor the user's traffic install a MITM proxy on the user's device. This breaks the client-server connection into two connections: client-proxy and proxy-server. The proxy analyses the traffic to determine whether it is safe. This raises security concerns. If the proxy's root certificate is pre-generated, users may be vulnerable to a MITM attacker who has access to a signing key if the proxy accepts external site certificates issued by its own root certificate. The proxy must also verify certificates, which may be vulnerable to several threats. Verifying certificates is difficult for tested TLS libraries. The proxy introduces a new TLS client, which must have all the latest updates to protect against new vulnerabilities. The new proxy connection may not match the parameters of the original connection, weakening the security and declaring the connection to be more secure than it is. The authors describe techniques to locate the private keys. To locate private keys in files and the Windows registry, they use ProcMon during installation to monitor the actions, check the Windows certificate manager for new certificates, and identify the process that created the certificate. Then they examine the registry and associated files that may contain the private key. For app-protected private keys, the authors identify the process filtering traffic, dump the memory of such processes, search memory for private keys and root certificates, and identify the process filtering traffic by noting which process

holds the private key in memory. Some protect their private keys with a passphrase. To retrieve passphrases, the authors extract strings of printable characters and try that as the password, disassemble (using IDA Pro to find OpenSSL functions related to private keys), execute (loading binary into the Immunity debugger), or break an SQLCipher encrypted database in some cases. Their TLS proxy testing framework involves certificate validation testing, proxy-embedded trust stores, TLS versions, and known attacks against TLS. They implement this framework in a number of antivirus and parental control applications that use a TLS proxy. They find that not one app implemented the TLS proxy that defended against all the tests the authors did.

In our work, we check the TLS versions of EMA servers as well as test them against known proxy vulnerabilities with howsmyssl.com [4] and ssllabs.com [50] to test the security parameters of the proxy. Only one app in our study uses a proxy for interception, so we avoid the in-depth analysis this paper made because it only covers 1/10 of the apps in our analysis.

Burgess et al. [13] look at any potential security or privacy risks of remote proctoring apps. The authors also examine any racial bias introduced by the apps' facial recognition system. They look at four proctoring apps for their security and privacy issues. They release a tool [12] to extract the security and privacy properties of proctoring apps. The authors ask whether the app provides the claimed security. Further, they ask what security and privacy the user must give up for the app to provide the security it claims. The authors try to investigate claims of procedural fairness, security, and integrity with reverse engineering techniques. They find that cheating measures can be easily subverted. To analyse these apps, they monitor the network with standard techniques (presumably Wireshark or tcpdump). They ask whether the traffic is TLS or plaintext. If TLS is used, they examine how the programme responds to a certificate with an incorrect name or a certificate that is not recognised by any authority but bears the correct name. The authors use reverse engineering techniques to examine VM detection, clipboard management, screenshot capture, application restrictions, and network interception by the apps. Further, they analyse exam content protection (encryption in transit and encryption at rest). Finally, they assess ID verification and authentication by evaluating login, general interaction fingerprinting, and facial recognition.

In our work, we try the tool released with this paper to see what it reports on EMAs. But this tool does not work accurately for us. For example, this tool reports on whether an app records screenshots, and the tool reports back that this binary does not record screenshots. But we know this to be false based on the functionality we report as well as what is marketed on the EMA website. We do not use the results of the tool in our analysis to gain insight into how these apps work. The only strategy we borrow from this paper is

network analysis; we also check to see that the apps are encrypting data in transit.

2.2.3 Our Techniques

The local analysis techniques we use in this work are checking for residual vulnerabilities, checking whether the apps comply with Windows privacy permissions, using an anti-keylogger to see if their keylogging is detectable and running the apps through malware analysis suites.

Network analysis techniques we use are standard network imposition with Wireshark, using mitmproxy to intercept encrypted traffic, checking proxy for known vulnerabilities, server geolocation and checking for certificate pinning.

Back-end analysis techniques we use are checking password requirements, preventions of brute force password guessing attacks, input validation, searching for open cloud storage, HSTS enforcement, checking TLS versions, assessing access control of private media, and verifying that deleting private user data works as expected.

2.3 Security and Privacy Issues

From related work and our project goals, we decide on the following list of issues to analyse employee monitoring applications.

1) Insecure private data transmission: is private user data sent over the network unencrypted (HTTP)?

2) Residual vulnerabilities: does the application leave behind any file containing private data recorded about the employee after uninstallation?

3) Privacy features claims upheld: do these apps stop recording when requested?

4) What is the general monitoring profile of these applications? Do they send more data than they claim?

5) Network vulnerabilities: do the apps use secure techniques to collect users' web traffic?

6) Back-end security configurations: HSTS enforcement, TLS versions, password security, input validation and open cloud storage buckets are all potential vulnerabilities of a back-end web-server.

7) Where are the servers located? What legal jurisdiction may be applicable to EMA data collection?

8) Does delete mean delete? When a user or manager requests their data to be deleted, is it really deleted and is it done in a timely manner?

9) Access control: is private user data viewable only by those with proper credentials?

2.4 Ethical Concerns and Responsible Disclosure

When we assess for vulnerabilities, we ensure we are running these attacks against our own accounts. We do not use any vulnerabilities we find to breach the systems. We stop our assessment once we collect enough data to confirm the existence of a vulnerability.

In this work, we have identified a number of privacy and security risks. We hope to disclose this information to the developers of the vulnerable apps. However, we are still waiting on legal support from the university to name the apps we have looked at. So, until we have legal support, we will not publicly name the apps we have assessed, nor will we contact the developers.

Chapter 3

Approach

3.1 App Selection

We choose 10 EMAs for this analysis. We are focusing on popular apps that are widely used and recommended for employee monitoring. This software has similar feature profiles to any spyware, but we choose these apps based on the marketing material. If the tool is marketed towards managers, then it fits our inclusion criteria. We choose apps based on popularity [42, 22]. We select apps that offer a free trial. The apps we assess offer the full feature set in the trial version so we are able to test all of the features. The exception being SmartAuthority's facial recognition feature which requires a subscription. When we began our analysis SmartAuthority also offered webcam monitoring in the trial version but later limited this feature to paying customers only. We were able to assess this feature before SmartAuthority changed the trial version. We study these apps for months and would be downloading them many times, so this was an economical choice.

These apps have a variety of feature sets. The more powerful apps offer advanced monitoring capabilities like keylogging, webcam monitoring, screen recording, remote takeover, GPS tracking, instant messaging monitoring and more. The less powerful apps may just track time spent on the computer and web browsing history. The general app structure is based on the client-server model. A client-side application collects data on the employee device and transmits the data to the server. The server-side collects, aggregates and displays the data in a dashboard format. Generally the server is hosted in the cloud. However, JoltDrone is the exception to the rule where the server is hosted on premises of the employer.

The apps we choose are: SmartAuthority, CerebralMega, EmployHour, IndustryVibe, CounterCube, JoltDrone, Oversightio, CoreCrew, DateExpert, and AliveRecord.

For each app, we create an account, download the required installers, and install the application in Windows Sandbox. We use Windows Sandbox [46] for experimental repeatability; each time we install the application, it is in a fresh sandbox. Further, Windows Sandbox helps protect our devices against the advanced tracking capabilities of these apps.

Our analysis of the data these apps collect relies heavily on network analysis. We are able to see what data these apps collect based on the network traffic. Because most of the apps we choose use TLS for communicating with the server, we use a man-in-the-middle (MITM) to break TLS encryption and read the traffic coming from the EMA. More details are to follow in section 3.2.3.

3.2 App-centred Analysis

This analysis approach is focused on the app itself and includes both static and dynamic analysis techniques. For each app, we install it in a sandbox and monitor the application with various tools to inspect how it runs and any vulnerabilities in its security or protection of privacy. We employ static and dynamic analysis techniques in our investigation. A high level overview of the app-centred issues we assess is shown in Fig 3.1.

3.2.1 Malware Analysis

We use a set of anti-virus engines in the VirusTotal system to assess whether these apps would raise any flags for a curious employee or a corporate security system. We provide VirusTotal the app executable (not the installer), and the tool runs the app through up to 71 anti-virus engines, and we tabulate the results.

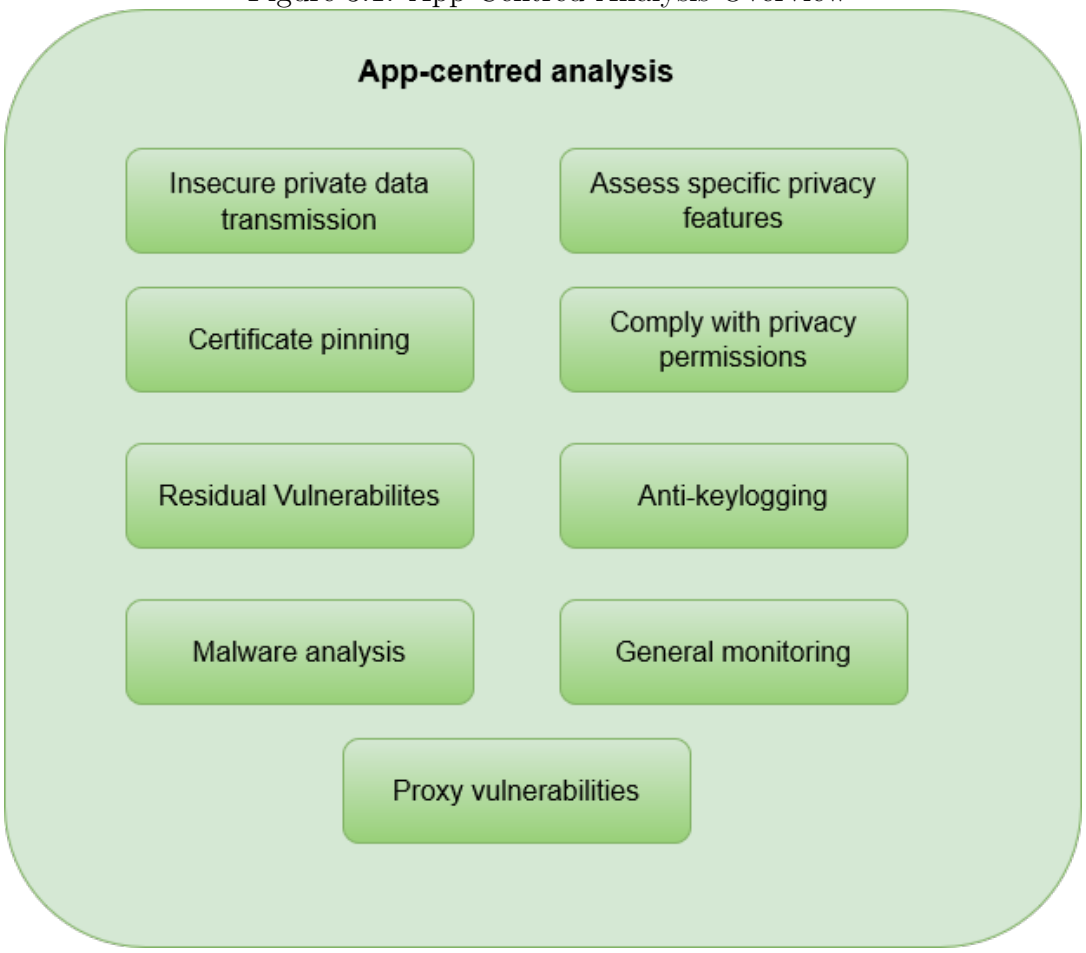
To get the application file, we install each application and use the application executable in the installation directory.

VirusTotal uses a suite of anti-virus engines, including popular AV tools like Avast, McAfee, Fortinet, and others.

3.2.2 Anti-Keylogging Analysis

We want to see if keylogging is detectable by a curious employee and whether keylogs are stored on disc before being sent over the network to the server. We use the KL-

Figure 3.1: App-Centred Analysis Overview



detector tool [9], which monitors changes to the filesystem during a period. We, the user, type into a notepad app during the monitoring process. The KL-detector tool will record changes to filesystems that are potentially suspicious (ignoring windows folders, which often are changing something while the OS is running) and point the user to these potentially suspicious folders if the tool noticed changes. We use this approach to see if there is evidence of keylogging from the user's perspective.

3.2.3 Traffic Interception

For each app, we run the app in the sandbox and run Wireshark on the host system to log the traffic. But because most modern apps encrypt their traffic, we need a man-in-the-middle (MITM) to intercept the traffic and see what data these apps are collecting and sending. We use mitmproxy [14] as our MITM. To use this program, we install a certificate in the Windows Trusted Root Certification Authorities Certificate Store so that apps trust the MITM as a server. This way, we can break the encryption and read what the apps are sending.

We connect the sandbox device to a router that has mitmproxy running. We run the proxy in transparent mode and are breaking encryption at the router. This is the simplest way to make sure we are intercepting the traffic. The traffic must flow through the router, and we are ensuring that the router redirects the traffic to the mitmproxy process.

Once we see the device running the sandbox is having traffic intercepted, we apply an evaluation protocol to assess the data collection options provided by EMAs. Variable monitoring options include screen capture, microphone access, webcam access, web traffic monitoring and website blocking. With each setting we collect the data transferred by the EMA with this setting on and off.

Our evaluation protocol steps are as follows:

1) Install the application and wait 5 minutes.

2) Once the app is installed, enable the variable monitoring option at the server-side and wait 5 minutes.

4) Disable the variable monitoring option at the server-side and wait 5 minutes.

5) Enable the variable monitoring option at the server-side and wait 5 minutes.

We initially tried PolarProxy[44] and Proxifier[31] to intercept TLS traffic entering and leaving the Windows sandbox. Both tools are installed on the Windows sandbox. Proxifier is configured to redirect all traffic to PolarProxy, except traffic coming from PolarProxy (to avoid an infinite loop).

We installed PolarProxy's certificate in Windows Trusted Root Certification Authorities Certificate Store so that the EMAs trust PolarProxy as a server and we can decrypt the TLS traffic. This method worked as expected for some EMAs, where we were able to decrypt and read the TLS traffic. But other EMAs ignored our proxy rules and bypassed the Proxifier/Polar Proxy setup.

Because some apps ignore windows proxy settings, we choose to use a transparent proxy setup. A transparent proxy eliminates any trouble with apps ignoring proxy rules by putting the proxy on the router.

Wireshark

In Wireshark, we gather and decode all traffic arriving at the router from the interface to which the sandbox machine is connected. With the DNS requests, we can see all domains contacted and get the IP addresses for server geolocation.

Wireshark allows us to see what protocols the apps are using for data transmission. If the traffic is unencrypted, we can read it in Wireshark.

We analyse the DNS traffic to see all hosts contacted by the monitoring application, and we use these hostnames for geolocation analysis.

Certificate Pinning

Certificate validation by a Windows app generally checks the certificate chain by going up the hierarchy until they arrive at the root certificate; if the root certificate is in the Windows trusted certificates, then the app trusts this certificate.

Certificate pinning differs from this general validation technique in that apps will only trust a pre-defined certificate or only certificates signed by the pre-defined certificate.

This means that without access to the pre-defined certificate, the application will not trust a certificate we put in the Trusted Root Certification Authorities Certificate Store.

We install a certificate into the Trusted Root Certification Authorities Certificate Store with a mitmproxy certificate on the sandbox machine. This means if the app trusts any generic certificate, we can break encryption with mitmproxy. However, if this technique does not work, it is indicative of certificate pinning.

Typically with certificate pinning, an app will have a certificate embedded in the program such that only this certificate is trusted. Meaning you can only break encryption with access to this certificate.

One approach we use to break certificate pinning is to see if we can find the certificate and replace it with our own or just make it empty. So, we search for certificates in the file folder in which the application was installed.

For CerebralMega we find two security certificates in the installation directory. However, these certificates do not match with the certificates we get from the servers that CerebralMega app communicates with. We get the server certificates with an OpenSSL [59] command: "openssl s_client -connect host:port -key our_private_key.pem -showcerts".

Proxy Security

CerebralMega uses a proxy for intercepting traffic. This proxy runs on the client machine and breaks all the encrypted traffic from the user. We check this proxy for any security vulnerabilities it may introduce. These may be related to its cryptography, interception certificates, and validation.

We use badssl.com and howsmyssl.com to check the security configuration of this proxy. We install CerebralMega in the sandbox and ensure the program is using the proxy by checking the configuration and that the web traffic is visible on the dashboard. Then we access the web-based tool through the browser, with the CerebralMega proxy running to run tests on the proxy.

howsmyssl.com checks the TLS version, cypher suites, support of ephemeral keys, session tickets, whether there's TLS compression, and whether there's a vulnerability to a BEAST (Browser Exploit Against SSL/TLS Attack), which is only possible with TLS 1.0.

badssl.com [34] checks certificate validation, broken cryptography, interception certificates, legacy cryptography, and domain security policies. The site checks whether the browser connects to sites with the previously stated vulnerabilities, and if it does, this means that an attacker could modify websites that a user visits.

3.2.4 Residual Vulnerabilities

We want to know if, after uninstalling these apps, any artefacts are left behind in the file system. These could potentially be a security or privacy vulnerability if they contain the information these apps log about the user. To do this analysis, we install FolderChangesView [54], which monitors the Windows file system for changes. We install this tool and run it. Then we install the app and wait 5 minutes. Next, we uninstall the app and stop monitoring folder changes. We can see all files modified, added, or removed during this time and see if any files from the app are left behind. On the Windows platform, there are many files being changed in the background. Because of this, we will not manually assess each file modified during this period. Instead, we focus on folder that are likely to

be related to the app and monitoring data such as "Program Data" or "Program Files" folders.

3.2.5 Hidden App Installation Directory

Many apps include the option for a "hidden" app, where there is no icon displayed in the taskbar. We investigate where the program files go in this instance. We assess how difficult it is for an employee to see if there is a monitoring program on their device.

3.2.6 Privacy Features

Several of these apps implement features to give the employees control over when they want to be monitored. We inspect these features individually to investigate whether they work as expected. SmartAuthority, Oversightio, CoreCrew, DateExpert and IndustryVibe offer employee-end privacy controls such that on the client end, an employee can turn off and turn on monitoring. AliveRecord does not have client-end controls; instead, an employer can add an employee to a do-not-track list. EmployHour, CerebralMega, and JoltDrone have no client-end privacy controls nor a do-not-track list option on the server-side. If the app gives the option, we install the app with the non-hidden mode enabled and enable the toggling of turning monitoring off and on. This way the employee can control when monitoring occurs, and we assess the feature for effectiveness.

For each app that offers client-end privacy controls, we employ the following protocol:

1) Install the app with the non-hidden mode settings and options for turning off and turning on monitoring.

2) Once the app is installed and running, wait 5 minutes with default monitoring settings.

3) Turn off monitoring at the client-side and wait 5 minutes with monitoring off.

4) Turn monitoring on at the client-side and wait 5 minutes.

5) Turn monitoring off at the client-side and wait 5 minutes.

We repeat steps to confirm our observations. If we find that the same observation is made repeatedly, we can conclude that the finding is not a one time quirk in the system. If there is a server-side restart button, we apply the same steps as above but with toggling the restart button instead, of the monitoring setting. We assess the restart button to understand if an employers' controls can override the employee's privacy controls.

3.2.7 Windows Privacy Permissions

Windows provides privacy permission options for the user to control. The permission options govern what sensitive data an app can collect, such as data from the webcam or microphone. The apps requesting permissions can be viewed and changed from "Privacy & Security" settings page, searchable from the Start Menu. The user can select what permissions a given app has access too. Further, an icon can be viewed from the taskbar indicating that an app is using the webcam or microphone.

For apps that record webcam and microphone data, we want to know whether these apps comply with the permission controls set forth by Windows. Malware has been shown to override Windows privacy permissions and record without informing the user [63]. We will investigate whether EMAs evade Windows privacy permissions or conform with the OS privacy and security settings.

3.2.8 Proctoring Suite Analysis Tool

Burgess et al. [13] developed a tool [12], which they used on Windows apps to analyse the security and privacy properties of Windows proctoring apps. We use this tool on EMA executables to analyse them for their respective security and privacy properties. This tool is available on Github and runs as a command-line tool, with the path to the binary as an input parameter.

3.2.9 How do EMAs access browser history?

There are four possible methods an EMA can access an employee's browser history [18]. The most simple method is for the app to gather DNS requests and take the hostname from the requests to see which site the employee visits. This method, however, will not show the full URIs, only the domain name. The second approach is to use a proxy as a MITM and intercept the traffic at the proxy. The third approach is to look in specific folders for browser history, for example, \localappdata\Google\Chrome\User Data\Default\History. The fourth potential method is to inject a DLL that will hook into the browser process and record the full URI.

We analyse the EMAs to see what browser monitoring techniques are used by EMAs. We can rule out DNS traffic recording if they show the full URIs visited by an employee. We use ProcMon [51] from the Windows System Internals tools to see if a DLL is injected into the browser process by the EMA. This would indicate that the EMA is using the fourth

method for web browser monitoring. We use IDA Pro [26] to perform static analysis on the DLL for insight on how it functions. Only one app in our selection uses a proxy for web browsing monitoring, CerebralMega. If an app does not use one of the previous methods, we infer that the EMA is using the third method. This will be confirmed if the EMA provides the browser history prior to installation of the EMA, because the EMA is gathering browsing history from files stored on the employee's device.

3.2.10 QUIC

With IndustryVibe, we identified that the EMA uses TLS as well as QUIC for network communication. To try and break QUIC encryption we use the developer version of mitmproxy, which has support for QUIC. The released version does not yet have QUIC support.

We download the newest development version of mitmproxy, and run it in transparent mode. We can run the program and decrypt TLS traffic as with the fully released version of mitmproxy. However, despite the development version supporting QUIC traffic, we still see QUIC packets passing through the proxy and are not intercepted. So, this method for breaking QUIC encryption failed.

3.2.11 SmartAuthority's Facial Recognition Feature

SmartAuthority offers a premium feature, facial recognition feature that records from the employee webcam with snapshots every 5 seconds [**?**]. Because this feature is considered an extra, we purchase a subscription, webcam monitoring and the facial recognition feature. The photos are taken and processed by machine learning models to locate the face in the picture and identify the face. The prediction model compares to any other face in the database, which is comprised of any people identified through this feature. So, the database grows as this feature is deployed on more employee machines. SmartAuthority claims this feature is designed to improve discipline, reduce employees working under false identities, and improve productivity. Then SmartAuthority reports photos of all people who worked on the computer, and each photo has the name of the person (if previously identified) and a timestamp.

We wish to evaluate the accuracy of the model by testing it with simple tricks. We install SmartAuthority, enable the facial recognition model, and allow SmartAuthority to record faces. We intercept the traffic with mitmproxy, with our previously mentioned experimental setup. By intercepting the traffic, we can understand where the model makes inferences and other data sent with the webcam photos.

We show the webcam pictures of the thesis author and someone else to see how the model responds to photos and to introducing a new face. We also show part of the face (i.e., slowly moving from the edge of the frame), change the angle of the face, and take glasses on and off. Our goal here is to see if any of these methods can fool the model. If we can fool the model with such tactics, we conclude that the model has obvious flaws in identification. For a feature as intrusive as this one, we would expect it to be highly effective with its supposed claims.

3.3 Back-end Analysis

This portion of analysis focuses on the back-end infrastructure of EMAs. EMAs send the client data to a server, which stores, processes and visualizes the data in a dashboard. In this portion of our investigation, we focus on the analysis of the web servers and cloud computing systems that EMAs employ. A high level overview of the backend issues we assess is shown in Fig 3.2.

3.3.1 Input Validation

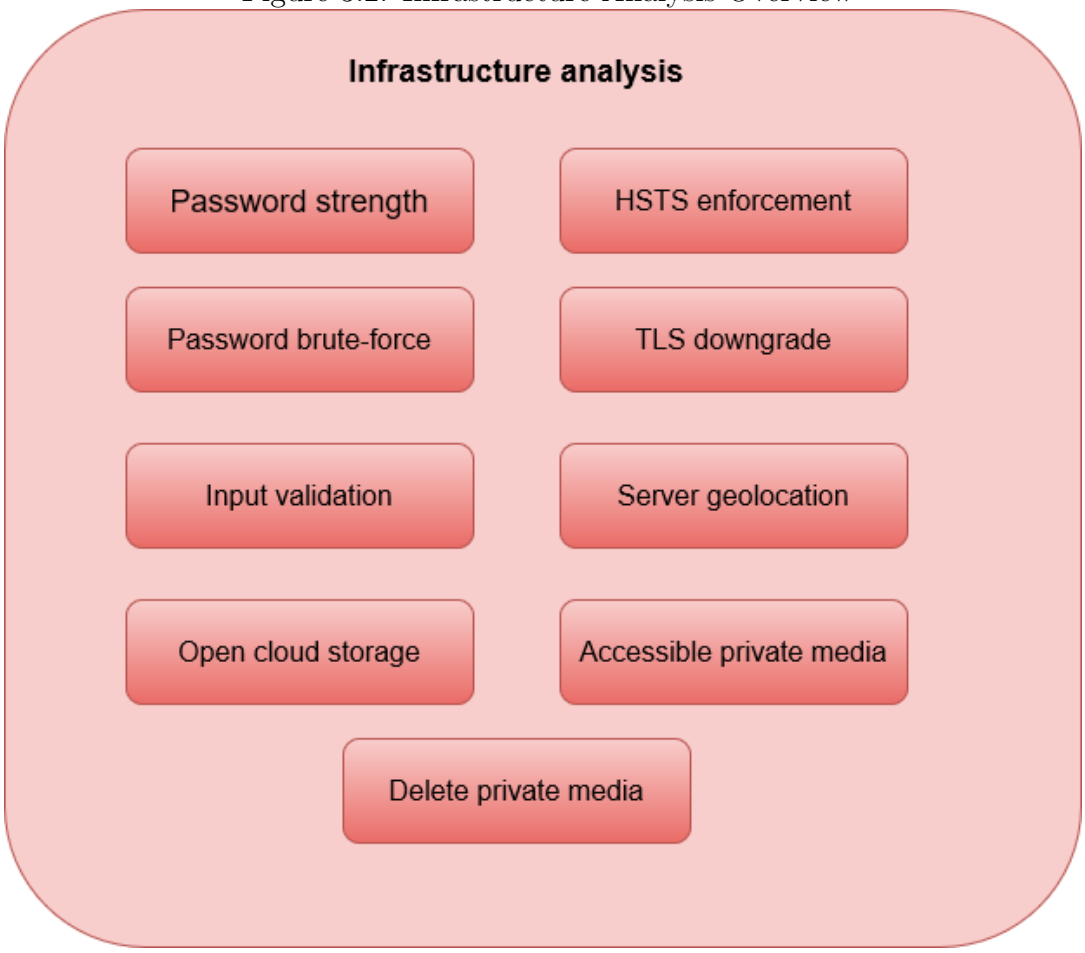
Any input field on a website has the potential to be an attack vector. An attacker could use such input fields to launch an SQL injection attack. Such attacks can be mitigated if developers use rules to allow only the expected types of input. Such rules include blocklisting certain characters or enforcing formatting rules for an email input field.

In our investigation we try to use a dummy injection attack to see if the website has input validation rules. Such an attack adds: " 'or 1=1;—" to the tail end of an email in the email field. Which would always return true, so we can use our password to see if we can gain access.

3.3.2 Password Security

Strong password requirements are an easy method to improve security against brute-force password guessing attacks. We investigate each EMA's password requirements on the vendor website to determine the relative strength of their password requirements. We expect to have requirements of at least eight characters and two types of characters or more (i.e., uppercase and lowercase, or letters and numbers).

Figure 3.2: Infrastructure Analysis Overview



3.3.3 Cloud Resource Enumeration

Many modern web apps use cloud infrastructure for storing media. This design principle has the potential to be a security risk if the cloud storage buckets lack proper access control. If private client data is stored in cloud storage buckets, for example, screenshots from their personal devices, and those buckets lack secure access control, there is a privacy vulnerability.

We employ cloud_enum [43] to enumerate over a large set of possible web URIs that an EMA may use. Such URIs could be formatted like: "dev.CerebralMega.ap-northeast-3.amazonaws.com". cloud_enum checks thousands of possible URIs to see if they exist

and are publicly accessible. This example of URI format comes from fuzzing over a list of common words used in cloud servers. The enumeration returns a list of URIs that are publicly accessible, and is often a small subset of all URIs checked. Once the enumeration completes, we manually check the URIs corresponding to open storage to see if they are owned by the EMA product and see what kind of data is stored in the cloud drives.

3.3.4 Private Media Access Control

Most of the EMAs in our investigation gather data in image, or video formats. The data may come from the screen capture or from the webcam. We investigate each EMA to see how they store this media on the web. We record screenshots from the user's device, and open the screenshot in a new tab to access the media URI. With the URI for the media file, we try to access the URI from another device that is not logged into the EMA dashboard. If the media can be accessed without logging in, this can be a security risk. We investigate further to see if there are any patterns in the URIs on which the media is hosted. If the URI is long and seemingly random, this poses less of a security concern.

To investigate the privacy risk, we see if, when an employer or employee requests their data be deleted, it is done in a timely manner. To do so, we access the cloud URI that hosts the employee's media (e.g., a screenshot of their device) and request that their media be deleted. Then we monitor the URI to see if and when the media is removed.

3.3.5 Server Geolocation

The geographical location of the web server will have an impact on which legal jurisdiction applies to the recording and storage of private employee data. To determine the geolocation of each server contacted by EMA clients, we record the web traffic in Wireshark and parse the DNS requests for all domains contacted by the EMA client. Once we have a list of the domains, we use the MaxMind API [41] and Linux whois commands to verify the location. We use two methods to cross-validate the results from one tool with those from the other. The results from these tools may differ, so we look into the result returned from the tools to give a better indication of where the server is located.

3.3.6 TLS Versions

We look at the TLS versions each EMA webserver has enabled to see whether they have older versions of TLS enabled. TLS versions 1.2 and 1.3 are considered secure and up-to-

date. Older versions are often accepted by most web servers for clients who are using old, outdated browsers.

3.3.7 HSTS Enforcement

A web server can avoid HTTP downgrade attacks with HTTP Strict Transport Security (HSTS). This means the server strictly enforces that you visit the site over HTTPS and not HTTP. We use mitmproxy to launch an SSL-stripping attack against a client trying to access EMA web servers. If the client can access the EMA website via HTTP, this indicates the server lacks HSTS enforcement.

Chapter 4

App-centred Analysis Results

We follow the method discussed in Chapter 3 between May 2022 and June 2023. For dynamic analysis, we run the apps on a Windows 10 Thinkpad P14s, running the app in a Windows Sandbox environment. We perform this analysis with 10 employee monitoring applications.

In this chapter, we present our results from analysing the security and privacy issues of EMAs at the application and local network levels.

4.1 Employee Monitoring Features

CerebralMega is one of the most powerful EMAs we have look at in our assessment. CerebralMega's feature set includes remote takeover, screen recording (live and playback), internet usage monitoring, file transfers, printings, emails, keystroke logging, instant message monitoring, and online meeting monitoring. CerebralMega uses a cloud model in that the server is hosted on cloud resources leased by CerebralMega. The employer installs the CerebralMega monitoring program on the employees' devices, and data is sent back to a cloud-hosted backend. Employers can access employee data through a web-based dashboard.

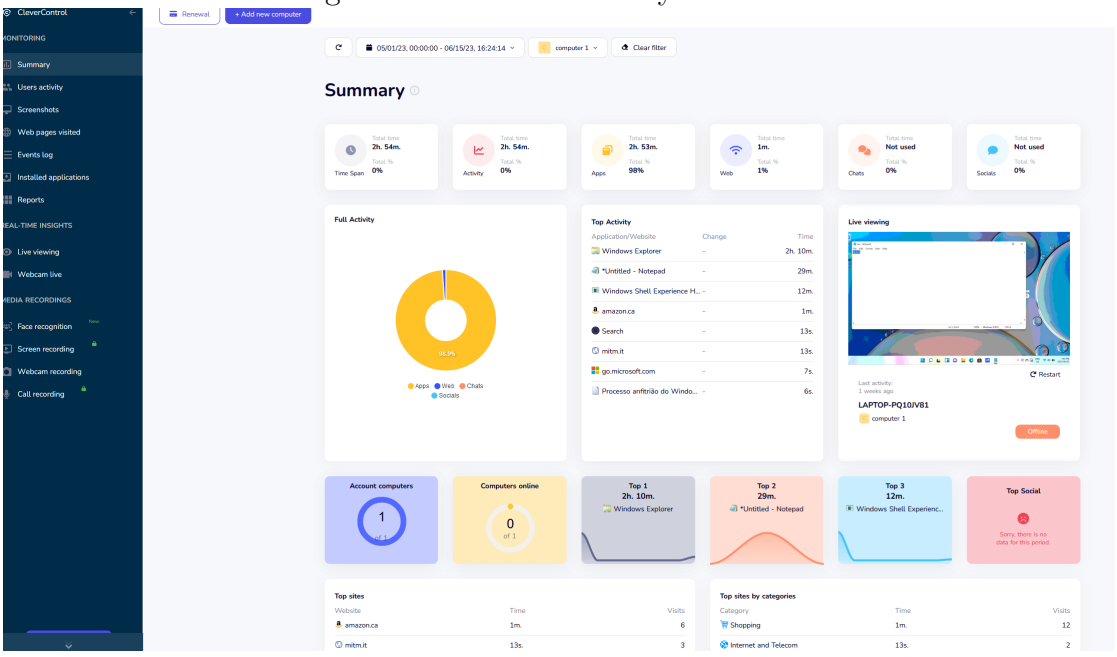
EmployHour has relatively few features compared to other apps we look at. EmployHour can monitor attendance, internet usage, online meetings, and active and idle times. EmployHour has on-premise and cloud-based deployment solutions. We elect to use cloud-based deployment because of the simplicity of setting it up in our analysis. Employers can see employee data on the web-based dashboard.

AliveRecord can track app and internet usage, provide productivity reports, and take screenshots with the “alarm feature”. AliveRecord provides activity logs that record each action an employee takes (e.g., opening a new tab or the computer going to sleep). The alarm can be triggered when the employee performs an action, such as visiting a URL containing“ebay.com”. AliveRecord uses a cloud model, and employers can see employee data and settings on the dashboard.

DateExpert provides features like time tracking, website, app, and chat monitoring, screenshots, screen recordings, and productivity analyses. DateExpert uses a cloud deployment, and employee data is accessed through the dashboard.

SmartAuthority has features such as key-logging, screenshots, live viewing of the screen, tracking instant messages, tracking internet usage, emails, social media, website blocking, recording from a webcam, live streaming from a webcam, tracking external media, printing, voice recording, and program activity monitoring. SmartAuthority has the most privileged features we look at in this analysis. SmartAuthority uses a cloud deployment, and employers can see employee data from the online dashboard, which we have pictured in Figure 4.1.

Figure 4.1: SmartAuthority Dashboard



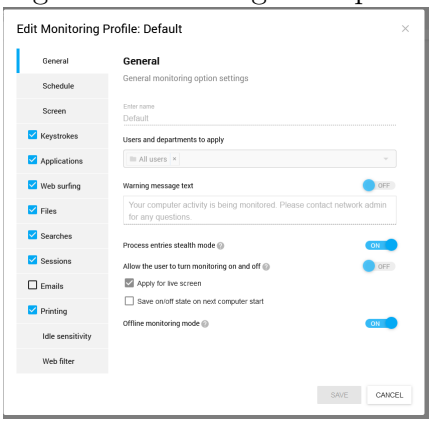
CoreCrew can record app usage, URLs visited, GPS location, time tracking, and screen-

shots. CoreCrew uses a cloud deployment with an online dashboard to display tracked data.

CounterCube tracks URLs, app usage, window titles, and screenshots. This feature set is relatively compact compared to others, but it still provides invasive features like screenshots. CounterCube deploys their server on the cloud and makes employee data available through the dashboard.

Oversightio offers monitoring features like attendance tracking, screenshots, key logging, internet usage monitoring, file accesses, printing usage monitoring, and email monitoring. Figure 4.2 shows Oversightio's monitoring options. Oversightio uses a cloud model and displays employee data on the online dashboard, viewable in Figure 4.3.

Figure 4.2: Oversightio Options

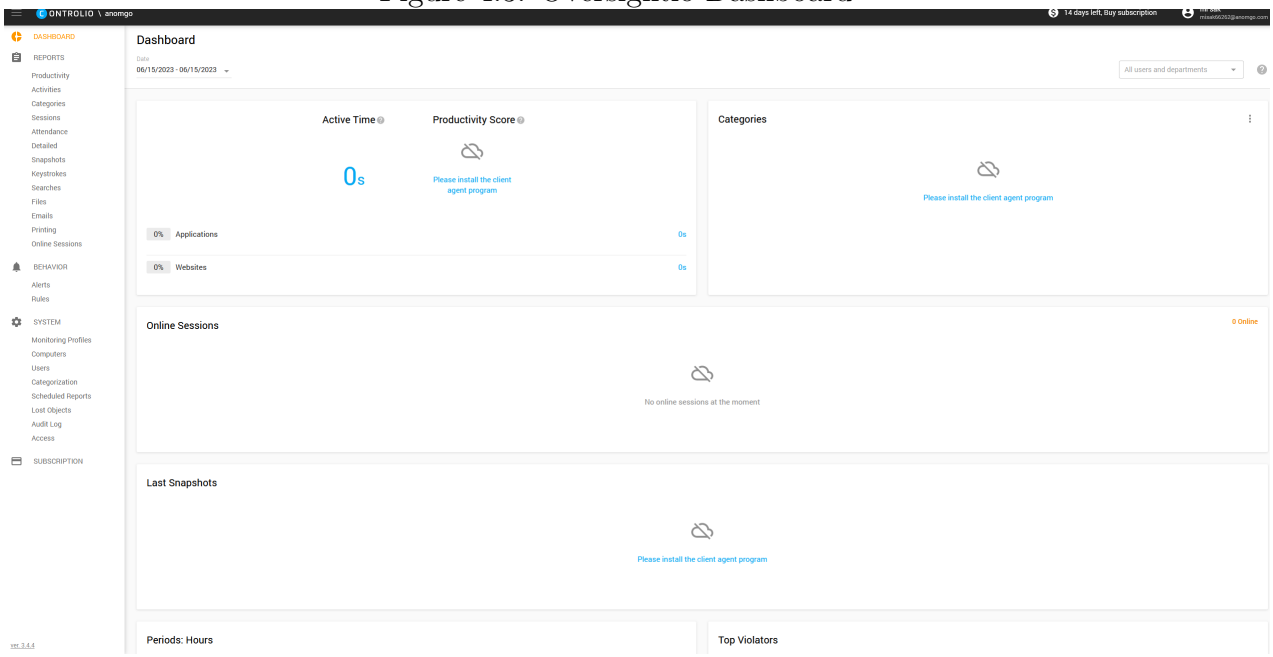


IndustryVibe offers standard EMA features such as screen recordings, time tracking, app and internet usage, and productivity analysis. They also use a cloud deployment model and display employee data on the online dashboard.

JoltDrone provides features such as online screen monitoring, screen recordings, keystroke logging, remote access, time tracking, productivity analysis, and violation detection. JoltDrone is the exception when it comes to deployment models. They use an on-premise server model, so the employer installs the JoltDrone server on their machine, and data from the employees is stored on that computer. The employer can view the data with a program called Viewer provided by JoltDrone, which pulls the data from the server program running on the same machine.

Table 4.1 shows which versions of apps we use in our app-centred analysis and when we downloaded the apps. We are missing the versions for some apps because the app did

Figure 4.3: Oversightio Dashboard



not have the version embedded in the file, nor were we able to recover a client app version from network traffic.

4.2 General Monitoring Profile

AliveRecord sends its data to a Google Cloud server in the form of a POST request. The requests include information such as AliveRecord account ID, OS, OS version, public IP address, device hardware ID, desktop username, list of open applications, titles of open windows, URLs accessed, and the user's timezone. The OS version, IP address, hardware IDs and other metadata collected and transmitted is not displayed on the dashboard. Collecting minute details about the observed machine falls into data creep, which builds a comprehensive profile of the monitored user. AliveRecord sends screenshots to the same Google Cloud server with a hex encoding.

SmartAuthority sends updates by opening a WebSocket, which is a full-duplex communication channel over a single TCP connection. The data sent in WebSocket messages

Product	Downloaded Filename	Date Obtained	Version
DateExpert	DateExpert2-setup-3.6.51-windows.exe	September 19, 2022	3.6.51
IndustryVibe	IndustryVibe.msi	July 25, 2022	NA
CoreCrew	CoreCrew-1.6.7-5c6fee47.exe	September 19, 2022	1.6.7
AliveRecord	ATAcct664059_1MjsrXbQTh_G_411674668850.msi	September 19, 2022	8.2.16.0
CerebralMega	CerebralMega_agent_x64_s.msi	September 21, 2022	NA
CounterCube	CounterCubeSetup.exe	September 19, 2022	1.3.613
EmployHour	wta_silentinstall.exe	September 19, 2022	11.0.0.0
Oversightio	wec_launcher_54hph9pm_.exe	September 23, 2022	2.5.0.0
SmartAuthority	SmartAuthority.NET	September 23, 2022	11.5.1029.3
JoltDrone	grabberEM.x64.msi	September 19, 2022	NA

Table 4.1: EMA Download Dates and Versions

are the open apps, window titles, desktop username, and screen dimensions. There are also other updates sent in Multipurpose Internet Mail Extensions (MIME) format. Data in the MIME updates include processor name, timezone, OS architecture, OS install date, webcam name, SmartAuthority program version, total physical memory, OS serial number, and more meta-data collected about the monitored computer. The information about the computer is not displayed on the dashboard, nor is it mentioned on their website. This metadata collection also falls under data creep.

Oversightio sends updates in a WebSocket, too. Messages in the WebSocket include the open apps, window titles, desktop username, account ID, and tracking options. The tracking options detail inactivity sensitivity (60 seconds) and directories monitored for file changes. Web servers for collecting data are hosted on Amazon Web Services (AWS).

CounterCube sends updates in JSON and MIME formats to their own webserver. The JSON updates show what apps the user has open and the path to the app's executable. The MIME updates include URLs accessed, window titles, and screenshots encoded in hex.

CoreCrew sends both JSON and MIME-formatted updates to an AWS URL. JSON updates include open apps and information on the mouse and keyboard, represented by an integer. MIME updates include information on the connection (i.e., expiration, signature, encryption, and credentials) and hex-encoded screenshots.

DateExpert sends updates in a WebSocket to a Google Cloud URI. WebSocket messages include open apps, URLs, mouse clicks, keystrokes, and mouse movements. DateExpert also sends screenshots in a PUT request to the Google Cloud server.

IndustryVibe also sends updates in WebSocket messages. These messages have the URLs accessed, filepaths to open apps, window titles, active/inactive state, timezone offset, and a gateway list (e.g., 00155D2CDB20), which could be a MAC address.

EmployHour uses no network encryption so we can read the data with Wireshark. EmployHour sends data that includes workstation ID, customer ID, active or idle, username, open applications, window titles, URLs accessed, EmployHour installation directory, local timezone, time tracked today, idle time today, key press count, and number of mouse clicks.

CerebralMega and JoltDrone both use certificate pinning so our network interception technique fails for these apps. Our analysis of these apps relies on the data we can see from the dashboard. CerebralMega offers powerful features like webcam monitoring, instant message monitoring, screen recording, keystroke logging and remote takeover of the device. JoltDrone also has a powerful feature suite, with keystroke logging and screen recording.

Full messages for each app are included in Appendix A.

4.3 Unprotected Private Data Transmission

For each app, we collect the network traffic with Wireshark. The majority of apps use TLS, meaning they encrypt the data transmission. EmployHour is the exception to the rule. EmployHour does not send the data it collects in TLS but instead in plaintext TCP.

EmployHour transmits PII over the network, visible to any eavesdroppers. The most sensitive information includes web browsing history, mouse usage, keyboard usage and last time since last input. This data should be kept private from any eavesdroppers, but it is transmitted over the public internet without protection.

4.4 Certificate Pinning

We find that 7 out of 10 apps do not use certificate pinning. These apps are SmartAuthority, IndustryVibe, CounterCube, Oversightio, CoreCrew, DateExpert, and AliveRecord. These apps trust any generic certificate, so we are able to intercept and remove the network encryption with mitmproxy. EmployHour does not use encryption, so the problem of certificate pinning is not relevant.

We find that 2/10 apps do use certificate pinning. CerebralMega and JoltDrone use certificate pinning, which means we cannot intercept traffic originating from these apps. We were unable to remove encryption for these two apps. There are certificate unpinning tools available, but these are built for the Android platform. To bypass certificate pinning for these apps would require significant reverse engineering, which falls outside the scope of this work. We did find a certificate within CerebralMega files, and we try removing it or replacing it with mitmproxy's certificate, but both attempts did not allow us to see CerebralMega traffic.

4.5 Residual Vulnerabilities

We find that 10 out of 10 apps left no files behind after installation or uninstallation. We use FolderChangesView [54] to monitor changes from install to uninstall. It could be a potential vulnerability if an app records private data, stores it somewhere on the computer, and does not remove the file when the app is uninstalled. This is of particular concern when the computer is a shared work device. We find that all apps were secure against this potential threat.

4.6 Hidden App Installation Directory

Oversightio, IndustryVibe, JoltDrone, SmartAuthority, CerebralMega and DateExpert allow for a hidden app. For each of these apps we investigate where the program files are stored in the filesystem. The most common method for obfuscating the install directory was by using a serial number in the ProgramData folder, which is hidden by default. The user can enable viewing of hidden folders in Windows File Explorer and find the program files. CerebralMega, Oversightio and SmartAuthority all use this method. The second most common method is to store program files under an alias. JoltDrone stores program files in the folder "TeleLinkSoftHelper" within "Program Files" directory. DateExpert stores program files in "SFproc" folder also within Program Files directory. Finally, IndustryVibe stores program files in ProgramData directory without an alias, simply "workpuls".

4.7 VirusTotal Analysis

We run each app's executable through the VirusTotal analysis system to find out how an antivirus engine would react to the EMAs in our assessment. We find that 4/10 apps test positive in at least one anti-virus engine. Each app is tested on up to 71 antivirus engines. Potential listed threats in CerebralMega were its netfilter, which is likely related to the app's proxy for intercepting network traffic. CounterCube and EmployHour are classified as potential grayware. Grayware is a classification of software that blurs the line between legitimate software and malware. Typically spyware is considered as malware or grayware. SmartAuthority is identified as malware by four AV engines.

4.8 Proxy Vulnerabilities

CerebralMega is the only app in our assessment that uses a proxy to intercept user traffic, record it, and send it to the back-end server. We analyse CerebralMega's proxy against known proxy vulnerabilities. While using the proxy, we check the security with howsmyssl.com [4] and badssl.com [34].

We find that CerebralMega's proxy has several vulnerabilities. Badssl.com checks certificate validation, interception certificates, broken cryptography, legacy cryptography, domain security policies, and some security settings. CerebralMega's proxy connects to untrusted-root.badssl.com, which means the certificate issuer is unknown and the proxy

Product	Positive Count	Negative Count	Engines Used	% Positives
DateExpert	0	66	66	0
IndustryVibe	0	56	56	0
CoreCrew	0	68	68	0
AliveRecord	0	60	60	0
CerebralMega	7	54	61	11
CounterCube	1	44	45	2.2
EmployHour	1	70	71	1.4
Oversightio	0	71	71	0
SmartAuthority	4	59	63	6.3
JoltDrone	0	61	61	0

Table 4.2: Summary of VirusTotal Analysis

should stop the connection, so CerebralMega has a high risk with this flaw in certificate validation. CerebralMega is also vulnerable to interception certificates as it connects to preact-cli.badssl.com and webpack-dev-server.baddssl.com. CerebralMega is vulnerable to interception attacks where an attacker modifies the webpage a user is visiting. CerebralMega also has a risk of broken cryptography by connecting to dh1024.badssl.com, which is considered a weak cryptographic scheme.

Howsmyssl.com checks the TLS version, ephemeral key support, session ticket support, TLS compression, TLS downgrade, and insecure cypher suites. CerebralMega passes all checks on howsmyssl.com. So, this means it is using up-to-date TLS versions, supports ephemeral keys and session tickets, does not use TLS compression, and uses up-to-date cypher suites.

CerebralMega's proxy is used to filter and intercept user traffic. We can see that this proxy introduces many vulnerabilities, increases the risk of interception, and accepts outdated cryptographic schemes. CerebralMega's web proxy is a risk to user privacy and security.

4.9 Anti-key-logging Analysis

4/10 apps in our analysis use key-logging as a feature to track user behaviour. CerebralMega, Oversightio, SmartAuthority, and JoltDrone use key-logging. We use KL-detector [9] to detect this activity with each of the 10 apps. We find that 6/10 apps are

Product	Key-logging feature?	Detectable by an anti-key-logger?
EmployHour	no	no
AliveRecord	no	suspicious
DateExpert	no	suspicious
CerebralMega	yes	no
CoreCrew	no	no
CounterCube	no	no
Oversightio	yes	suspicious
IndustryVibe	no	suspicious
SmartAuthority	yes	suspicious
JoltDrone	yes	suspicious

Table 4.3: Summary of key-logging analysis

identified as potentially suspicious by KL-detector. The suspicious apps are AliveRecord, DateExpert, Oversightio, IndustryVibe, SmartAuthority and JoltDrone. AliveRecord, DateExpert and IndustryVibe are false positives in this assessment. KL-detector also points to the files that changed during analysis and may contain the key logs. For each of the suspicious apps, we perform a manual analysis of the potentially suspicious files. In each case, there are no key logs to be found in files on the computer. So, we conclude that the apps that are key-logging are storing the key-logs in program memory and sending them over the network rather than storing them in local files and transmitting the file over the network. The suspicious files held general logging data but are not related to the user's keystrokes.

4.10 Privacy Feature Assessment

7/10 of the apps in our assessment have the option of client-end privacy controls. These control features allow the employee to start or stop monitoring. 2/7 of the apps with client-end privacy control have the feature implicitly built into the app. The employee starts monitoring when they want and ends it when they want. 5/7 of the other apps add privacy controls as an additional feature, so a manager can choose whether or not to enable privacy controls. 1/10 apps, AliveRecord has privacy controls available to the manager on the dashboard. 2/10 apps do not have privacy controls: CerebralMega and JoltDrone.

3/8 of apps that implement privacy controls have risks associated with privacy control features. For each app with privacy controls, we apply the protocol from Section 3.2.6.

We turn off monitoring at the client side, wait a few minutes before toggling monitoring off again at the client side, and repeat this process. Our aim is to see whether the privacy feature works as expected and whether data is not transferred when monitoring is disabled.

Oversightio offers two privacy features: GDPR mode and client-end privacy controls. The GDPR mode is described as a disabling of some data collection. This mode can be enabled or disabled by the manager on the dashboard; the employee does not have control. Oversightio's website and dashboard claim to no longer collect window titles, URLs, screen recordings, keystrokes, searches, files, emails, or printings in GDPR mode. However, we still see Oversightio sending window titles and URLs. The window titles and URLs are not displayed on the dashboard, though. We also see screenshots and videos sent, but these were recorded before we enabled GDPR mode. Meaning the data was not collected while GDPR mode was on.

Oversightio also offers control over what media is collected, such as web browsing or screen captures. This control is available on the dashboard for the employer. We notice that when the employee turns off monitoring and while the monitoring is disabled, their employer turns off screen captures and the employee starts monitoring again, a single frame of video is sent by Oversightio. This appears to be a bug; Oversightio tries to record video before checking what the monitoring options are. This bug is especially concerning because the frame of the video was visible from the dashboard. So, an employer may inform their employee that they will not record their screen. But, because of this bug, their screen is recorded for an instant.

SmartAuthority offers client-end privacy controls for employees. They also offer a server-side restart button to the employer. We find that when an employee has disabled monitoring, an employer can hit the program restart button, which will start SmartAuthority monitoring. After a few moments (less than one minute), the monitoring halts and data collection stops. It appears that SmartAuthority will record data when restarted without first checking whether the employee has enabled monitoring. This violates the employee's privacy because they would think that while they have monitoring turned off, there would be no data collection. But, their employer still has access to the program restart button, which can start data collection for a short interval. The data collected in this interval was not displayed on the dashboard, however. But we can see it was collected by intercepting the network traffic.

AliveRecord offers server-side privacy controls in that an employee can be added to a do not track list and they will not be monitored. This do-not-track list works mostly as expected but takes up to five minutes to update. This finding is not overly concerning except for the case when an employer tells their employee they have been added to a do-

not-track list immediately after adding them. In this case, the employee is monitored for a few minutes before the list updates.

4.11 Windows Privacy Permissions

CerebralMega and SmartAuthority use the microphone and webcam data for surveillance. For both apps we toggle Windows privacy permissions for both microphone and webcam. When the webcam or microphone is in use, the user is made aware with an indication in the taskbar. Upon the user changing the privacy settings to deny access to the microphone and webcam, the data collection halts. Once the data collection halts, the dashboard view stops displaying webcam footage or microphone audio. We find that both apps comply with Windows privacy permissions. This indicates that they are using the camera and microphone permissions in the standard method and are not overriding Windows privacy permissions.

4.12 How do EMAs Access Web Browsing History?

There are three main categories in which an app may collect browsing history from a user. 1) A TLS proxy; 2) Collect browser data from local files (i.e., %localappdata% \Google \Chrome\User Data \Default \History); 3) Inject a DLL into the browser and hook into functions that handle the request URLs.

We assess the apps in our study to see what strategies they use to access browsing history. We know CerebralMega uses a proxy for web traffic filtering and interception. We use ProcMon to see if Oversightio injects DLLs into the browser for traffic collection. We find Oversightio does inject DLLs into the browser. With those DLLs, we open them in IDA Pro to read the binary and gain insight on what is happening. We find hooking functions, which we conclude it uses to gather the URLs accessed by the user. SmartAuthority has the ability to import web browsing history prior to installation. Because of this feature, we conclude that SmartAuthority is using the local browser files to record web traffic. Also, we find that the web browsing history on the server side takes several minutes to load. This finding makes sense given that these local files take a few minutes to update.

We find that EMAs are using all three strategies to collect web history.

4.13 Instant Messaging Monitoring

2/10 apps in our assessment have features specific to monitoring instant messages (IM), CerebralMega and SmartAuthority. We install and use Slack, Skype, and Signal while the apps are monitoring our behaviour. The apps are only able to capture Skype messages. Even though CerebralMega claims to explicitly support Slack IM monitoring, we observe that CerebralMega does not track those messages.

With keystroke monitoring turned off, this feature still worked. This has the potential to be confusing if a manager tells their employees that their keystrokes are not logged but their Skype messages are still recorded verbatim.

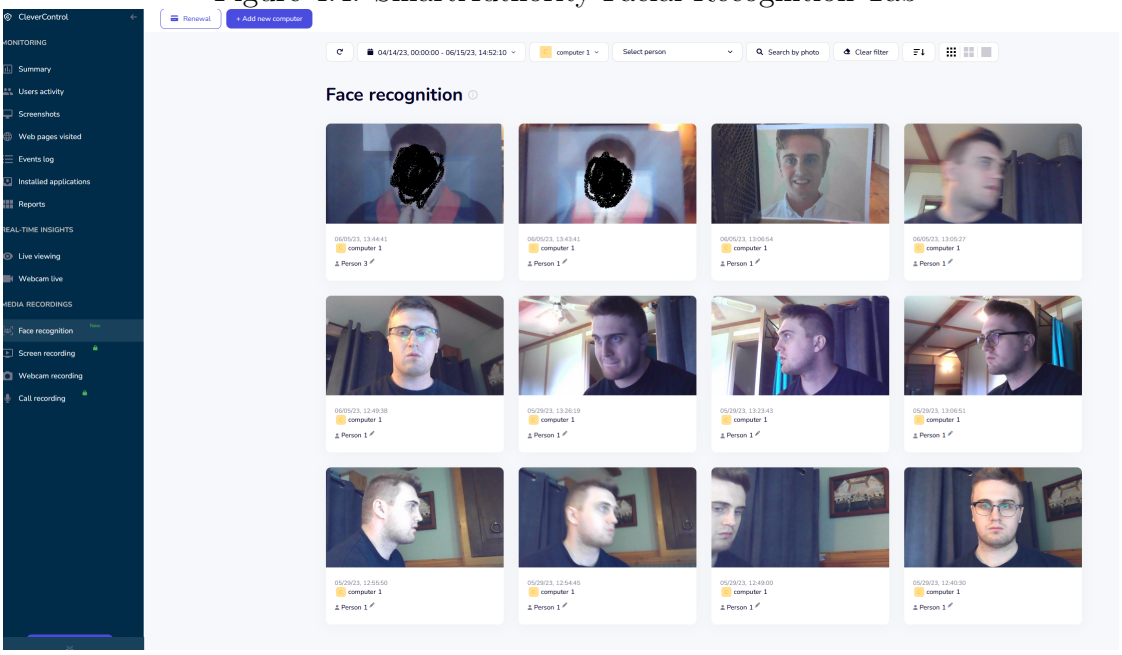
4.14 SmartAuthority's Facial Recognition Feature

We want to see how easily facial recognition can be fooled and whether the user is informed that they are being observed by a facial recognition model. A claimed benefit of this feature is that it can reduce the number of employees working under false identities. If we can fool the model with simple tricks, this feature fails to meet its claims. For instance, the author took a picture of himself and held it in front of the webcam. The facial recognition model identified this as the author. An employee could leave their desk with a picture of themselves to fool the model into thinking they are still there. Further, we tried using a picture of another person in the same setup. The model both identified this as a new person and identified it as the author. Changing the angle of the face, putting on glasses, and showing only part of the face did not fool the model into thinking the author was someone else. In Figure 4.4, it is shown that changing the face angle, putting on glasses, a picture of the author, and a picture of someone else were all identified as person 1. The only photo where a new person was identified is where the author put a photo of someone else in front of the webcam.

The intercepted messages did not include who was in the picture (i.e., person 1); they just contained the picture without saying who was in it. This is surprising because we found the ML models in the local SmartAuthority program files. The model we found was the DLib open-source facial recognition model [35]. So, it seems the inference is done server-side. The employee is not made aware that the software is using facial recognition; they can only see the webcam light on. So, the employee only knows if the employer explicitly tells them.

This facial recognition feature is flawed, identifying photos as people and identifying a photo of another person as the author.

Figure 4.4: SmartAuthority Facial Recognition Tab



4.15 GDPR Compliance

CoreCrew provides a short description; they claim that they work hard to be GDPR compliant and are privacy shield certified. They also offer an option for an employee to ask for their data to be deleted by asking their employer.

AliveRecord provides a relatively in-depth explanation. There is a table with GDPR requirements and recommended actions for the organisation or agency to take in order to comply with GDPR. They also briefly explain what GDPR is.

CerebralMega describes GDPR and how it can help organisations conform with the requirements. They describe features such as selective recording to protect privacy, categorising PII to apply privacy, and exfiltration rules, continuous enforcement of policies, identity-based authentication, and segregated access control to prevent PII leakage, and screen recording only during policy violations. However, it is not clear how to set this up. It seems the employer must manually configure each setting to be GDPR-compliant.

CounterCube claims it complies with GDPR. They claim data is secured on secure cloud servers in the EU. Any third-party service providers also comply with GDPR, and more detailed data protection information is viewable on the privacy policy page. Overall,

CounterCube gives a short description, but CounterCube has fewer monitoring capabilities compared to some other apps (e.g., CerebralMega.)

EmployHour gives a legal disclaimer that their GDPR advice is general only and that a lawyer should be consulted for legal advice. They describe the principles to follow in order to conform to GDPR and say how EmployHour can conform with those principles. But as per the disclaimer, their advice is general. They claim EmployHour is non-invasive and focused purely on productivity.

Oversightio provides a GDPR mode, which is described by: "In this mode, some data Collection functionality will be disabled on the agent, server, and console sides (windows titles, web URLs, screen recordings, keystrokes, searches, files, emails, and printings). GDPR-related data collection and display will be stopped.". An employer can also configure a warning message for employees to tell them that they are being monitored. Overall, this is a short description, but the user only needs to activate GDPR mode instead of configuring each setting. With this mode, PII will not be leaked, so Oversightio claims. Despite Oversightio's claim that collection of window titles and web URLs will be disabled with GDPR mode on, we observe collection and transmission of windows titles and web URLs as discussed in Sec. 4.10.

IndustryVibe provides a legal disclaimer as well, saying that this advice should not be used as legal advice. They lay out the regulations in each EU country, saying some are stricter than others. They also mention how monitoring remote employees introduces different legalities and it is important to be aware of the law. They mention that GDPR states that employees must be informed that they are being monitored.

SmartAuthority also states that the main aspect of GDPR is informing employees that they are monitored. They do say that GDPR explicitly prohibits keystroke logging and screen recording, which no other app states.

JoltDrone describes GDPR, what the principles are, and how it can be ensured. They generally describe how employee monitoring can be GDPR-compliant. They say it is important to make monitoring as transparent as possible, to make monitoring as unobtrusive as possible, and that JoltDrone you can restrict some of the more powerful features (i.e., keystroke logging and screen recording).

DateExpert describes how users have additional rights over their data in the EU. They describe the rights of access to your data, the right to correct it, and the right to restrict data collection. DateExpert provides contact information for their EU representative to exercise their data rights. DateExpert also describes the California Consumer Privacy Act, which provides the right to know what data is collected and the right to delete data. They claim that the data requested to be deleted will be done within 30 days.

Chapter 5

Backend Analysis Results

In this chapter, we focus on the back-end infrastructure EMAs use to host their websites and store and display the data they collect. Our analysis consists of checking the infrastructure against known vulnerabilities and attacks.

5.1 Password Strength and Protections

For each app, we check the password requirements for the app's dashboard as well as assess protections against brute-force password guessing attacks. We attempt to login to the account up to fifty times to see if we encounter any protection mechanisms. We tabulate the results in Tables 5.1 and 5.2.

3/10 apps (CerebralMega, EmployHour and SmartAuthority) have password requirements of just 6 characters and no other requirements on character types. The other apps have more sophisticated requirements, either in terms of the number of characters or more complex character types, or a combination of both.

2/10 apps have no protection from a brute-force password-guessing attack (DateExpert and EmployHour). Notably, EmployHour has weak password requirements and no protection from brute-force attacks. EmployHour's password security is the most concerning of the apps in our assessment.

We recommend app developers use password requirements of at least 8 characters and include uppercase, lowercase, numerals, and special characters. We also recommend app developers use some protection mechanisms against password brute-force guessing attacks.

Application	Password Rules
DateExpert	8 characters min., uppercase, lowercase, number and a special character
IndustryVibe	8 characters min., lowercase, uppercase, number
CoreCrew	6 characters min., uppercase, lowercase, number
AliveRecord	8 characters min., lowercase, uppercase, number, special character
CerebralMega	at least 6 characters long
CounterCube	at least 8 characters
EmployHour	at least 6 characters long
Oversightio	8-64 characters, one digit, one uppercase, one lowercase
SmartAuthority	at least 6 characters
JoltDrone	8 characters min, digits, uppercase and lowercase letters

Table 5.1: Summary of Password Requirements

Such protection mechanisms could include a captcha to make automating the task more difficult or rate limiting guess attempts at 5-10 attempts and locking the account out for a period upon reaching this limit. Otherwise, any password can be broken with enough guessing.

5.2 Uninformed Suspicious Activities

For each of the 10 apps in our assessment, we perform password changes and a login from a device different from the one currently logged in. In each case, we record any notifications or updates.

We find that 3/10 apps warn the account owner about a login from a new device. The updates for a login from a new device are not sent by email but instead are visible from the server dashboard. AliveRecord displays a warning banner to the user on the dashboard about a new login. SmartAuthority shows the logins from a "Latest Actions"s tab in the dashboard. Oversightio shows logins to the dashboard from the "Sessions"tab in the dashboard.

5/10 apps warn the account owner about a password change. CoreCrew, CounterCube and JoltDrone all send the password change warning via email. CounterCube's warning is implicit because the only way to change the password is via the email link. CoreCrew and JoltDrone warnings are explicit because an account owner can change the password without an email link. AliveRecord and SmartAuthority display a warning about the password change the same way they display a warning about a login from a new device.

Application	Password Brute Force attack results
DateExpert	no warnings, no lockouts
IndustryVibe	5 minute lockout after 10 attempts
CoreCrew	1 hour lockout after 10 attempts
AliveRecord	account locked after 5 attempts, must reset password
CerebralMega	5 minute lockout after 10 attempts
CounterCube	30 second lockout after 5 attempts
EmployHour	no warnings and no lockouts
Oversightio	account locked (must reset password) after 10 attempts, with email warning
SmartAuthority	1 hour lockout after 2 attempts
JoltDrone	Uses a CAPTCHA

Table 5.2: Summary of Password Protections

4/10 apps have no warnings at all for suspicious activities. These apps are CerebralMega, EmployHour, DateExpert, and IndustryVibe. We consider these apps to be potentially vulnerable to an attacker gaining access to the backend. Because these apps collect and display highly sensitive data, we suggest that EMAs show warnings about password changes and logins from new devices.

5.3 Input Validation

We check each app's login form for basic input validation. We use a benign SQL injection attack with email= "ouremail@emailserver.com'or 1=1;–" and password="some-password". Where we have an account with the password "some-password". That way if the injection attack succeeds we only access our account.

All 10 apps deny access with this simple SQL injection attack. 9/10 apps tell us that it is an invalid email format. CoreCrew says that the reason our login was blocked may be because of SQL injection.

5.4 Server Geo-location

We collect all DNS requests while EMAs are on. We check each domain name in the MaxMind API [41] to check the geolocation in which the server resides. We tabulate the

Product	Login from New Device Warning	Password Change Warning
CerebralMega	no	no
EmployHour	no	no
AliveRecord	yes, visible from dashboard	yes, visible from dashboard
DateExpert	no	no
SmartAuthority	yes, visible from "latest actions" tab	yes, visible from "latest actions" tab
CoreCrew	no	yes, email warning
CounterCube	no	yes, via password reset link
Oversightio	yes, viewable from sessions list	no
IndustryVibe	no	no
JoltDrone	no	yes, email warning

Table 5.3: Summary of Uniformed Suspicious Activities

results in Table 5.4. The majority of servers are located in the USA. The exceptions are a CerebralMega server located in Germany, SmartAuthority servers located in Canada, and a CounterCube server located in India. It is important to note that the German CerebralMega server and Indian CounterCube server are not the main servers with which the apps are communicating. Instead, the traffic sent to these servers contains data on connectivity. This connectivity data is a check that all the servers are reachable and could be connected to.

5.5 Access Control of Private Media

With each app, we assess whether the private media EMAs collect (screenshots, webcam snapshots, and screen recordings) are stored in authenticated buckets. We allow the app to record screenshots and use the dashboard to access the media. We find that 5/10 apps do not have access control for private media as shown in Table 5.5. This means that even someone who is not logged in can access the media if they have the URL. However, the URLs used by each app are long in number and seemingly random. We find that AliveRecord is encoded as a token in base64. The decoded token is in Appendix **??**. We were unable to decode any other URLs.

App	Country
EmployHour	USA
AliveRecord	USA
DateExpert	USA
CoreCrew	USA
CerebralMega	USA
CerebralMega	Germany
CounterCube	USA
CounterCube	India
IndustryVibe	USA
Oversightio	USA
SmartAuthority	Canada
JoltDrone	USA

Table 5.4: Geo-location of EMA webservers

Product	Publicly Accessible Media
EmployHour	N/A
AliveRecord	yes
DateExpert	yes
CerebralMega	no
CoreCrew	no
CounterCube	no
Oversightio	yes
IndustryVibe	yes
SmartAuthority	yes
JoltDrone	no

Table 5.5: Publicly Accessible Media

5.6 Does Delete Mean Delete?

For each app in our analysis that provides the option to delete screenshots, this works as expected. We check the URL hosting the private media before and after the deletion is requested. In each case, the URL is inaccessible within minutes or seconds. We conclude that the media is no longer stored and has been successfully deleted.

DateExpert gives the employee the option to request their data be deleted. This was done as expected and in a timely manner.

5.7 Server-side Security Configurations

In this subsection we discuss our findings on back-end security configurations of EMA web-servers.

5.7.1 HSTS Enforcement

We use mitmproxy [14] to launch an SSL-stripping attack against a user attempting to access EMA webservers. If possible, the connection will degrade to HTTP instead of the usual HTTPS. The most secure practice is to use HSTS enforcement, which ensures users can only access a website through HTTPS. We find that only CerebralMega has HSTS enforcement enabled on their webserver. In all other apps, we were able to successfully use an SSL-stripping attack on a client visiting EMA web servers.

5.7.2 TLS Downgrade Vulnerability

We use ssllabs.com [50] to assess what versions of TLS are enabled on each EMA web-server. The most secure and up-to-date protocols are TLS 1.2 and TLS 1.3. Older versions of TLS enabled could be a vulnerability, as these protocol versions are susceptible to known attacks (BEAST and CRIME). We find that 8/10 apps have TLS 1.0 enabled on their web-server, shown in Table 5.6. 2/10 apps (AliveRecord and CoreCrew) have only TLS 1.2 and higher enabled. We consider AliveRecord and CoreCrew to be the most secure in regards to this particular threat.

Hostname	Vulnerable?
https://2.timedoctor.com	yes,TLS 1.0 enabled
https://app.activtrak.com	no
https://app.hubstaff.com	no
https://www.teramind.co	yes, TLS 1.0 enabled
https://desktime.com/	yes, TLS 1.0 enabled
https://www.worktime.com	yes, TLS 1.1 enabled
https://app.controlio.net	yes, TLS 1.0 enabled
https://app.insightful.io (WorkPuls)	yes, TLS 1.0 enabled
https://dashboard.clevercontrol.com/	yes, TLS 1.0 enabled

Table 5.6: TLS versions for each web-server

Product	Open Cloud Storage?
EmployHour	no
AliveRecord	no
DateExpert	yes, but not private data
CerebralMega	yes, but not private data
CoreCrew	no
CounterCube	no
Oversightio	no
IndustryVibe	no
SmartAuthority	no
JoltDrone	no

Table 5.7: Publicly Accessible Cloud Storage

5.7.3 Open Cloud Buckets

We use cloud_enum [43] to enumerate potential cloud buckets for each EMA. We find that 2/10 apps (DateExpert and CerebralMega) both have open cloud storage buckets, shown in Table 5.7. We manually check the buckets to see if private data is stored in them. In both cases, we do not find private user data. In CerebralMega's bucket, we find a marketing material such as a landing page. In DateExpert's bucket, we find development code, which could indicate how the web app operates and may provide insight to potential attackers. The code we found was related to image editing. The threat level of this development code is likely low, but remains a security oversight.

Chapter 6

Discussion and Conclusion

6.1 Contributions

We presented our analysis framework and assessed 10 EMAs for Windows. Our analysis revealed a number of bugs and security vulnerabilities. Our research question was *how well do EMA vendors protect the data they collect about employees?* We identify vulnerabilities in each of the apps we assess. Our findings demonstrate that EMA vendors need to provide better app-level and back-end security to adequately protect employee's private personal data. EMAs use highly privileged features yet claim to protect employee privacy and security. An overview of our findings is tabulated in Table 6.1. Our key results are as follows: We find that EmployHour has no network encryption, so any eavesdroppers can read private data pertaining to an employee's behaviour on the computer. We observe Oversightio collecting and transmitting window titles and web URLs while GDPR mode is on. This observation contradicts Oversightio's claims that window titles and web URLs are not collected at the client-side while GDPR mode is on. CerebralMega, EmployHour, and SmartAuthority have weak password requirements. EmployHour and DateExpert do not have rate limiting for password brute force attacks. EmployHour is especially vulnerable to account hijacking because of the combination of weak password requirements and a lack of rate-limiting protection against a brute-force password attack. All apps except CerebralMega lack HSTS enforcement and are vulnerable to SSL stripping attacks. CerebralMega, EmployHour, DateExpert, CoreCrew, CounterCube, Oversightio and JoltDrone leave the account owner uninformed of either logins from new devices or password changes. These kinds of warnings can help indicate that someone is trying to hijack the account. CerebralMega uses a proxy for network traffic interception and collection. This

Security Issue	CerebralMega	EmployHour	AliveRecord	DateExpert	SmartAuthority	CoreCrew	CounterCube	Oversightio	IndustryVibe	JoltDrone
Insecure PII Transmission		♡								
Weak password requirements	♠	♠			♠					
SSL stripping		♡	♡	♡	♡	♡	♡	♡	♡	♡
Password brute force attack		♠		♠						
Uninformed suspicious activities	♠	♠		♠		♠	♠	♠		♠
Proxy Vulnerabilities	♡	-	-	-	-	-	-	-	-	-
Certificate Pinning		-	♡	♡	♡	♡	♡	♡	♡	
Access control of private media		-	♠	♠	♠			♠	♠	
Open cloud storage	♠	♠		♠		♠				

Table 6.1: Summary of security vulnerabilities in employee monitoring applications. ♡: on-path network attacker, ♠: remote attacker, -: not applicable, blank: no threat

proxy introduces known vulnerabilities, such as trusting unknown certificate issuers, which make the user more vulnerable to interception attacks. AliveRecord, DateExpert, SmartAuthority, Oversightio and IndustryVibe have publicly accessible private media storage. The URIs can be accessed without credentials; however, the URIs are difficult to guess. CerebralMega, EmployHour, DateExpert and CoreCrew have open cloud storage. These cloud buckets do not contain private user data but instead marketing and development material.

With the security and privacy risks we found in our search, we suggest that EMAs be designed and developed with privacy and security as a priority. EMAs collect sensitive data and must comply with their employers' workplace surveillance practices. Because of the vulnerable position employees are in, it is of the utmost importance that EMAs provide adequate and functional security and privacy features.

In this thesis, we demonstrate that EMAs need to improve in the domains of security and privacy. We find that ten out of ten apps are vulnerable to at least one issue we assess in our framework. The most concerning issues we found are that EmployHour sends data without network encryption, CerebralMega's web interception feature introduces proxy vulnerabilities, Oversightio's GDPR mode still sends data that they claim not to collect, Oversightio collects and sends a single video frame sent while the employee has disabled

monitoring, and SmartAuthority's restart button triggers new communication briefly when the employee has monitoring turned off. Surveillance can increase distrust between employees and employers. EMAs increase the risk of security and privacy and thus stretch the relationship between employee and employer further. So, EMA vendors should at least provide adequate security and uphold the privacy they claim to protect. Given the trend towards increasing prevalence of EMAs, developers must take accountability for the protection of employees' security and privacy.

6.2 Recommendations for Developers

Given the highly sensitive nature of data collected by EMAs and the security and privacy issues we found, we suggest developers follow our recommendations:

1. Any traffic sent over the public Internet should be encrypted. The Internet is inherently public, and it should not be assumed that data sent through it is safe from modification and eavesdropping. Using up-to-date secure protocols for data transmission will ensure that the data collected about employees will arrive at the server untampered and unread.

2. Developers should apply our evaluation protocol (Section 3.2.6) to their privacy features to ensure they work as expected. EMAs offer a number of privacy features, so employees have some control over what and when data is collected. However, we have found a number of bugs in these features. The bugs mean that the employee may assume they are not being monitored, but in fact they are. This violates the trust an employee has in the surveillance practice.

3. Apply state-of-the-art back-end security configuration strategies. We found a number of open cloud storage URIs, web pages lacking HSTS enforcement, weak password requirements, a lack of password attack rate-limiting, and uninformed suspicious activities. Addressing those issues will improve the security of the system and thus better protect employee privacy.

4. Check for known proxy vulnerabilities with tools like badssl [34] and ensure the proxy only trusts desired certificates. Otherwise, the developers can use simpler strategies to monitor employee internet usage, such as gathering browsing history from local folders. Fixing the vulnerabilities we identified will improve communication security against traffic interception attacks.

6.3 Limitations and Future Work

A key limitation in our work is we were unable to bypass certificate pinning. In related work we find that this is done commonly when analyzing Android apps with tools like SSLUnpinning [62]. Two out of ten apps in our assessment use certificate pinning, so we were still able to intercept traffic from the majority of apps. However, more developers may use certificate pinning and make analysis of their apps more difficult. Certificate unpinning on Windows would likely require advanced reverse engineering to intercept SSL functions in application code and replace or remove the certificate. A system that could automate this task would be very useful to security and privacy researchers analyzing Windows apps. Such a system would likely need to intercept system calls the app uses to validate certificates. Intercepting said function and always returning true would allow us to bypass certificate pinning. This would require in-depth reverse engineering and would likely differ for each app, however there could be similarities between the apps and a general tool could be developed.

As this industry grows, so will the number of apps. To create a more complete view of the landscape, we suggest that the analysis of EMAs continues to include more apps. We only looked at Windows apps because of the prevalence in the workplace. A number of companies we looked at also offer mobile apps for monitoring employees phones. Analyzing mobile EMAs could apply our ideas to Android and iOS platforms but would likely require different techniques and tools.

Much of the dynamic analysis performed in this work was performed manually in a sandbox environment. An automated system that executes our evaluation protocol would help in the analysis of more apps. This system would require interacting with the UI, to turn features on and off throughout analysis.

The apps we assess may check whether they are in a VM or sandbox and alter their behaviour. We are able to observe all the apps features, and saw no explicit record of whether the apps checks whether it is in a VM. We assume the app was behaving normally in a sandbox. This assumption could be tested by evaluating the apps as we did but on a host OS instead of a sandbox environment.

We looked at some of the back-end security configurations; however there is still more that could be assessed. The API security could be analyzed for vulnerabilities such as authentication strength.

Finally, we suggest that lawmakers develop regulation regarding employee privacy in the workplace. So that employees have control over what is tracked, and that their data is protected by a standard.

6.4 Concluding Remarks

The EMA industry is growing due to ease of deployment, and the increasing trend towards remote work. This industry has not been analyzed for security and privacy issues because of its recent emergence. We have assessed 10 Windows EMAs with our analysis framework. Our framework uses dynamic analysis techniques such as network interception to read encrypted EMA traffic, anti-keylogging analysis, investigating changes to filesystem and registry, and assessing proxy vulnerabilities. We also apply static analysis techniques like malware analysis and assessing several backend vulnerabilities.

EMAs vendors claim that their product will protect security and privacy of the company and employees. Our analysis reveals that each app in our analysis is vulnerable to at least one of the issues we assessed. We find several bugs, which are particularly concerning when they relate to privacy features of the EMA. The apps claim to support employee privacy. However, the feature does not work as expected and the employees privacy can be violated. This is worse than employees assuming they have no privacy at all. Rather we have a few cases where the employee will assume their activity is private, but instead they are still monitored because of a bug in the system.

Our analysis lays a general landscape of the security and privacy issues of EMAs. We suggest that analysis of EMAs continues, to create a more complete picture of the security and privacy vulnerabilities of EMAs. This will hopefully urge EMA developers to fix these bugs and issues to better protect employee privacy.

References

[1] The human factor in it security: How employees are making businesses vulnerable from within. https://www.kaspersky.com/blog/the-human-factor-in-it-security/.

[2] Postman. https://www.postman.com/.

[3] scyllahide: Advanced usermode anti-anti-debugger. https://github.com/x64dbg/ScyllaHide.

[4] howsmyssl.com. https://www.howsmyssl.com/, 2022.

[5] Written policy on electronic monitoring of employees. https://www.ontario.ca/document/your-guide-employment-standards-act-0/written-policy-electronic-monitoring-employees, Jul 2022.

[6] Suzan Ali, Mounir Elgharabawy, Quentin Duchaussoy, Mohammad Mannan, and Amr Youssef. Betrayed by the guardian: Security and privacy risks of parental control solutions. In *Annual Computer Security Applications Conference*, pages 69–83, 2020.

[7] Gianni Amato. peframe. https://github.com/guelfoweb/peframe.

[8] Meisam Navaki Arefi, Geoffrey Alexander, and Jedidiah R Crandall. Piitracker: automatic tracking of personally identifiable information in windows. In *Proceedings of the 11th European Workshop on Systems Security*, pages 1–6, 2018.

[9] Yohanes Aristianto. KL-detector: detect keylogger on your computer. http://dewasoft.com/privacy/kldetector.htm, 2010. [Version 1.3].

[10] Marcus Botacin, Hojjat Aghakhani, Stefano Ortolani, Christopher Kruegel, Giovanni Vigna, Daniela Oliveira, Paulo Lício De Geus, and André Grégio. One size does not fit all: A longitudinal analysis of brazilian financial malware. *ACM Transactions on Privacy and Security (TOPS)*, 24(2):1–31, 2021.

[11] Diane Buckner. No slacking allowed: Companies keep careful eye on work-from-home productivity during covid-19 — cbc news. https://www.cbc.ca/news/business/working-from-home-employer-monitoring-1.5561969, May 2020.

[12] Ben Burgess, Avi Ginsberg, Edward W Felten, and Shaanan Cohney. Proctoring suite analysis tool. https://github.com/WWP22/ProctoringSuiteAnalysisTool.

[13] Ben Burgess, Avi Ginsberg, Edward W Felten, and Shaanan Cohney. Watching the watchers: bias and vulnerability in remote proctoring software. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 571–588, 2022.

[14] Aldo Cortesi, Maximilian Hils, Thomas Kriechbaumer, and contributors. mitmproxy: A free and open source interactive HTTPS proxy. https://mitmproxy.org/, 2010–. [Version 9.0].

[15] Luke Craig, Andrew Fasano, Tiemoko Ballo, Tim Leek, Brendan Dolan-Gavitt, and William Robertson. Pypanda: taming the pandamonium of whole system dynamic analysis. In *NDSS Binary Analysis Research Workshop*, 2021.

[16] Xavier de Carné de Carnavalet and Mohammad Mannan. Killed by proxy: Analyzing client-end tls interception software. In *Network and Distributed System Security Symposium*, 2016.

[17] Xavier de Carné de Carnavalet and Mohammad Mannan. Privacy and security risks of" not-a-virus" bundled adware: The wajam case. *arXiv preprint arXiv:1905.05224*, 2019.

[18] Xavier de Carné de Carnavalet. Private communication, 2023.

[19] Erik Derr. Libscout: Third-party library detector for java/android apps. https://github.com/reddr/LibScout.

[20] Brendan Dolan-Gavitt, Josh Hodosh, Patrick Hulin, Tim Leek, and Ryan Whelan. Repeatable reverse engineering with panda. In *Proceedings of the 5th Program Protection and Reverse Engineering Workshop*, PPREW-5, New York, NY, USA, 2015. Association for Computing Machinery.

[21] Manuel Egele, Theodoor Scholte, Engin Kirda, and Christopher Kruegel. A survey on automated dynamic malware-analysis techniques and tools. *ACM computing surveys (CSUR)*, 44(2):1–42, 2008.

[22] G2.com. Best employee monitoring software. https://www.g2.com/categories/employee-monitoring.

[23] Nicola Galloro, Mario Polino, Michele Carminati, Andrea Continella, and Stefano Zanero. A systematical and longitudinal study of evasive behaviors in windows malware. *Computers & Security*, 113:102550, 2022.

[24] Greenbone. Greenbone/openvas-scanner: This repository contains the scanner component for greenbone community edition. https://github.com/greenbone/openvas-scanner.

[25] Moritz Gruber, Christian Höfig, Maximilian Golla, Tobias Urban, and Matteo Große-Kampmann. "we may share the number of diaper changes": A privacy and security analysis of mobile child care applications. *Proceedings on Privacy Enhancing Technologies*, 3:394–414, 2022.

[26] Ilfak Guilfanov and Hex-Rays. Interactive disassembler. https://hex-rays.com/ida-pro/, 2023.

[27] Hamza Harkous, Kassem Fawaz, Rémi Lebret, Florian Schaub, Kang G. Shin, and Karl Aberer. Polisis: Automated analysis and presentation of privacy policies using deep learning. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 531–548, Baltimore, MD, August 2018. USENIX Association.

[28] Adam Hickman and Lydia Saad. Reviewing remote work in the u.s. under covid-19. https://news.gallup.com/poll/311375/reviewing-remote-work-covid.aspx, Nov 2022.

[29] Galen Hunt and Doug Brubacher. Detours: Binary interception of win32 functions. In *Third USENIX Windows NT Symposium*, page 8. USENIX, July 1999.

[30] Tatum Hunter. Here are all the ways your boss can legally monitor you. https://www.washingtonpost.com/technology/2021/08/20/work-from-home-computer-monitoring/, Oct 2021.

[31] Initex. Proxifier the most advanced proxy client. https://www.proxifier.com/, 2022.

[32] Tsukasa OI Jesse Kornblum, Helmut Grohne. ssdeep - fuzzy hashing program. https://ssdeep-project.github.io/ssdeep/.

[33] Pranay Kapoor, Rohan Pagey, Mohammad Mannan, and Amr Youssef. Silver surfers on the tech wave: Privacy analysis of android apps for the elderly. In *Security and Privacy in Communication Networks: 18th EAI International Conference, SecureComm 2022, Virtual Event, October 2022, Proceedings*, pages 673–691. Springer, 2023.

[34] April King, Lucas Garron, and Chris Thompson. badssl.com. https://badssl.com/.

[35] Davis E. King. Davisking/dlib: A toolkit for making real world machine learning and data analysis applications in c++. https://github.com/davisking/dlib.

[36] Jeffrey Knockel, Thomas Ristenpart, and Jedidiah Crandall. When textbook rsa is used to protect the privacy of hundreds of millions of users. *arXiv preprint arXiv:1802.03367*, 2018.

[37] Joxean Koret. pyew. https://github.com/joxeankoret/pyew.

[38] Enze Liu, Sumanth Rao, Sam Havron, Grant Ho, Stefan Savage, Geoffrey M Voelker, and Damon McCoy. No privacy among spies: Assessing the functionality and insecurity of consumer android spyware apps. *Proceedings on Privacy Enhancing Technologies*, 1:1–18, 2023.

[39] Spherical Insights LLP. Global employee monitoring software market size to reach usd 2.10 billion by 2030: Cagr of 7.2%. https://www.globenewswire.com/en/news-release/2022/11/10/2553615/0/en/Global-Employee-Monitoring-Software-Market-Size-To-Reach-USD-2-10-Billion-By-2030-html, Nov 2022.

[40] Gordon Fyodor Lyon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure, Sunnyvale, CA, USA, 2009.

[41] MaxMind. MaxMind GeoIP2. https://www.maxmind.com/en/home, 2022.

[42] Neil McAllister. The best employee monitoring software for 2023. https://www.pcmag.com/picks/the-best-employee-monitoring-software, Jan 2022.

[43] Chris Moberly. cloud_enum. https://github.com/initstring/cloud_enum, 2022.

[44] Netresec. Polarproxy tls proxy. https://www.netresec.com/?page=PolarProxy, 2022.

[45] Air Force Office of Special Investigations, The Center for Information Systems Security Studies, and Research. Foremost. https://foremost.sourceforge.net/.

[46] Vinay Pamnani. Windows sandbox - windows security. https://learn.microsoft.com/en-us/windows/security/application-security/application-isolation/windows-sandbox/windows-sandbox-overview.

[47] Nisha Patel. Written policy on electronic monitoring of employees: Your guide to the employment standards act. https://www.ontario.ca/document/your-guide-employment-standards-act-0/written-policy-electronic-monitoring-employees, Oct 2022.

[48] Mark C. Perna. Why 78% of employers are sacrificing employee trust by spying on them. https://www.forbes.com/sites/markcperna/2022/03/15/why-78-of-employers-are-sacrificing-employee-trust-by-spying-on-them/?sh=666efd251659, Mar 2022.

[49] PortSwigger. Burp suite community edition. https://portswigger.net/burp/communitydownload.

[50] Qualys. ssllabs.com. https://www.ssllabs.com/ssltest/, 2022.

[51] Mark Russinovich. Process Monitor. https://learn.microsoft.com/en-us/sysinternals/downloads/procmon, 2023. [Version 3.93].

[52] Skye Schooley. Uses of employee monitoring software. https://www.business.com/articles/employee-monitoring-software-in-office/, journal=business.com, Apr 2023.

[53] Hispasec Sistemas. Virustotal.com. https://www.virustotal.com/.

[54] Nir Sofer. FolderChangesView: Monitor folder/drive/file changes on windiows. https://www.nirsoft.net/utils/folder_changes_view.html, 2012. [Version 2.35].

[55] The Week Staff. The rise of workplace spying. https://theweek.com/articles/564263/rise-workplace-spying, Jul 2015.

[56] Chris Sullo and David Lodge. Nikto2. https://cirt.net/Nikto2.

[57] The Tcpdump team. tcpdump. https://github.com/the-tcpdump-group/tcpdump.

[58] The Wireshark team. Wireshark. https://www.wireshark.org/.

[59] The OpenSSL Project. OpenSSL: The open source toolkit for SSL/TLS. https://www.openssl.org/, April 2003. www.openssl.org.

[60] threat9. Threat9/routersploit: Exploitation framework for embedded devices. https://github.com/threat9/routersploit.

[61] Xabier Ugarte-Pedrero, Davide Balzarotti, Igor Santos, and Pablo G Bringas. Sok: Deep packer inspection: A longitudinal study of the complexity of run-time packers. In *2015 IEEE Symposium on Security and Privacy*, pages 659–673. IEEE, 2015.

[62] GitHub users https://github.com/ac–pm and https://github.com/V–E– O. Sslunpinning - xposed module. https://github.com/ac-pm/SSLUnpinning_Xposed.

[63] Roberto M Vergaray and Julian Rrushi. On sustaining prolonged interaction with attackers. In *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 478–485. IEEE, 2017.

[64] Nisha Vinayaga-Sureshkanth, Raveen Wijewickrama, Anindya Maiti, and Murtuza Jadliwala. An investigative study on the privacy implications of mobile e-scooter rental apps. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 125–139, 2022.

[65] Daniel Votipka, Seth M Rabin, Kristopher Micinski, Jeffrey S Foster, and Michelle M Mazurek. An observational investigation of reverse engineers' processes. In *Proceedings of the 29th USENIX Conference on Security Symposium*, pages 1875–1892, 2020.

[66] x64dbg. X64dbg/x64dbg: An open-source user mode debugger for windows. optimized for reverse engineering and malware analysis. https://github.com/x64dbg/x64dbg.

[67] Yucheng Yang, Jack West, George K Thiruvathukal, Neil Klingensmith, and Kassem Fawaz. Are you really muted?: A privacy analysis of mute buttons in video conferencing apps. *arXiv preprint arXiv:2204.06128*, 2022.

[68] Miuyin Yong Wong, Matthew Landen, Manos Antonakakis, Douglas M Blough, Elissa M Redmiles, and Mustaque Ahamad. An inside look into the practice of malware analysis. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3053–3069, 2021.