Abusability of Automation Apps in Intimate Partner Violence

Shirley Zhang[†], Paul Chung[‡], Jacob Vervelde[†], Nishant Korapati[†], Rahul Chatterjee[†], Kassem Fawaz[†]

[†] *University of Wisconsin–Madison* [‡] *University of California–San Diego*

Abstract

Automation apps such as iOS Shortcuts and Android Tasker enable users to "program" new functionalities, also called *recipes*, on their smartphones. For example, users can create recipes to set the phone to silent mode once they arrive at their office or save a note when an email is received from a particular sender. These automation apps provide convenience and can help improve productivity. However, these automation apps can also provide new avenues for abuse, particularly in the context of intimate partner violence (IPV). This paper systematically explores the potential of automation apps to be used for surveillance and harassment in IPV scenarios. We analyze four popular automation apps — iOS Shortcuts, Samsung Modes & Routines, Tasker, and IFTTT — evaluating their capabilities to facilitate surveillance and harassment. Our study reveals that these tools can be exploited by abusers today to monitor, impersonate, overload, and control their victims. The current notification and logging mechanisms implemented in these automation apps are insufficient to warn the victim about the abuse or to help them identify the root cause and stop it. We therefore built a detection mechanism to identify potentially malicious Shortcuts recipes and tested it on 12,962 publicly available Shortcuts recipes. We found 1,014 recipes that can be used to surveil and harass others. We then discuss how users and platforms mitigate such abuse potential of automation apps.

1 Introduction

Automation applications enable users to automate certain tasks on their phones through user-friendly trigger-action interfaces. Users can create rules or "recipes" and chain them together to create new functionalities. For example, one can create a recipe to automatically open the Map application when they get into their car or put their phone on silent mode when they arrive at the office. Several automation tools are now available for various mobile platforms.[1] Shortcuts app [12] is preinstalled in Apple devices, and Modes & Routines[2] app [67] is preinstalled in Samsung phones and tablets. Other popular third-party automation tools like IFTTT [46] and Tasker [47] are available for Android and iOS users. Users can create complex automation recipes and share them using links or by uploading them on recipe libraries.

Figure 1: (**Left**) A user's post on Reddit about a self-made spying recipe using iOS Shortcuts. (**Right**) A post on Weibo — the most popular social media site in China — shows how to spy on an intimate partner's location using Shortcuts.

Although automation applications are designed for legitimate productivity tasks, they can be repurposed to create malicious recipes for surveillance and harassment. We found multiple social media posts that describe how to build such recipes. As shown in Fig. 1, one Reddit user shares a Shortcuts recipe capable of recording video, taking photos, and tracking location. A Weibo post, viewed over 32.5 million times, demonstrates how to use a shared note to secretly access a partner's geolocation via iOS Shortcuts.

Once installed on a victim's device, these recipes can function like spyware while being difficult to detect. Existing mobile anti-spyware tools scan for malicious apps, not user-created automation recipes. Because platforms like iOS Shortcuts and Samsung Modes & Routines are preinstalled and considered legitimate, malicious automations often go unno-

[1] Although some automation tools also work on laptops, we focus on mobile devices (Android and iOS) for this paper as they are more widely used today than other devices.

[2] formerly Bixby Routines, changed in 2023.

ticed. This poses significant risks in the context of intimate partner violence (IPV), where abusers often have physical and authenticated access to a victim's device [28, 38, 42, 56, 72]. Two authors, who provide direct support to IPV survivors, have observed the use of automation recipes for surveillance and harassment.

While prior research has examined the role of mobile apps in tech-enabled abuse [9, 28, 38, 64], automation platforms remain underexplored. These recipes introduce unique risks: they are difficult to detect, easy to distribute, and not subject to the same restrictions as conventional malware. Distribution of malicious automation recipes is largely unregulated and hard to trace. Users can share and download these recipes through unregulated online websites as well as personal messages.

Motivated by the real-world threats and risks posed by automation applications, this work systematically explores the capabilities of popular mobile automation platforms and their potential for misuse in IPV scenarios. We investigate the extent to which these tools can be repurposed for surveillance and harassment by addressing these research questions:

RQ1: *What are the capabilities of popular automation applications?*

RQ2: *What specific harms might these capabilities pose to IPV victims?*

RQ3: *How can abusive automation recipes be detected to support affected users?*

To answer these questions, we study four popular automation applications: Shortcuts for iOS, Tasker for Android, Bixby Routines for Samsung devices, and IFTTT for iOS and Android. Using a UI-Stepthrough analysis [21], we identify the capabilities of these applications for device controlling, communicating, and data managing (Section 3). We then demonstrate that combining the capabilities can yield surveillance and harassment attacks in the IPV context (Section 4).

Reflecting on the attacks, we show that the existing abuse prevention and detection strategies in mobile platform are not effective against malicious recipes (Section 4.2). We therefore develop an LLM-assisted system for analyzing Shortcuts recipes to detect recipes that pose the potential for abuse. We then run our detection system on 12,962 publicly available iOS Shortcuts recipes.

We identify 1,014 recipes that have the capabilities to be used for surveillance (*e.g.*, exfiltrate location data) or harassment (*e.g.*, sending too many notifications or toggle flashlights repeatedly). We also present an analysis of the detector on recipes that were explicitly marketed for IPV attacks (Section 5.4). Based on our findings, we discuss possible mitigation strategies from the perspective of both users and platforms in Section 6.

In summary, the key contributions of this paper are:

- We are the first to investigate the potential risk of automation applications for IPV. We analyze four automation applications across iOS and Android, and demonstrate how automation applications can be used to create recipes for surveilling and harassing victims.
- We build a novel LLM-based detector that leverages step-by-step prompting to minimize LLM hallucinations, enabling detailed assessments of a Shortcuts recipe's potential to constitute IPV attacks. Our approach is adaptable not only in analyzing automation recipes but also in other domains requiring context-sensitive evaluations.
- We surveyed 12,962 Shortcut recipes on four third-party libraries and found 1,014 recipes that are either committed to be malicious or can be repurposed for IPV.

To encourage more research on the security implications of automation tools, we will release the LLM detector code and prompts. For safety purposes, the annotated dataset will only be shared with verified researchers by request.

2 Background & Related work

Automation applications have gained popularity in recent years. Smartphone and smart home users utilize these applications to create new functionalities that meet their needs.

2.1 Automation Applications

Automation applications ("automation apps", in short) allow users to automate various tasks and processes on their mobile devices, saving time and effort. These apps create workflows, routines, or *"recipes"* that are sequences of operations. Recipes can be executed manually or by setting *triggers* such as time, location, and device state changes.

Automation apps offer a powerful and user-friendly interface that enables users to create complex automations without requiring a technical background. Many automation apps are available on official app markets and sometimes preinstalled on mobile devices. For example, Apple's iOS 12 introduced *Shortcuts* for task automation, and Samsung introduced *Bixby Routines* in 2019. Among third-party apps, *Tasker* (only available for Android) can automate tasks on mobile phones, *IFTTT* (available on both iOS and Android) can automate tasks that also involve IoT devices, and *Hootsuite* [57] mainly automate social media management.

End-user Programming. End-user programming (EUP) allows people without software development experience to program computers. Barricelli et al. [18] present a systematic EUP study highlighting a growing integration of EUP into different fields, including IoT and smart home automation. Automation apps represent one type of EUP: they simplify how users control their devices, allowing users to connect various services, devices, and actions seamlessly. We show in this paper that an abuser, even without coding skills, can

leverage automation apps to conduct powerful attacks in the IPV setting.

Privacy and Security Risks with Automation Apps. Prior reports and news have raised several safety concerns about automation apps. In 2019, researchers discovered that the Siri shortcut can be used to trick users into a *scareware* attack, which reads information and demands a fee from the user through native Siri voice [48]. Online forums contain discussions about the potential of leaking sensitive information through iOS Shortcuts [41]. Moreover, a malicious shortcut could be spread to everyone in the victim's contact list [50]. To address these security issues and concerns, Apple implemented safety measures to protect users. For instance, a shortcut must ask the user for permission to start a new recording. Although the usability of Shortcuts has been studied in different areas [14, 78, 81], no prior work has systematically studied the security and privacy issues of automation tools in an IPV context. Our work aims to fill this gap by analyzing and measuring their potential threats.

2.2 Technology-Facilitated Abuse

Technology is facilitating forms of abuse that were once limited to the physical world, such as stalking and harassing individuals [24, 32–34, 60, 77, 83]. According to Thomas et al. [77], over 48% of internet users report experiencing online abuse, and the threat has been increasing over the years. A global survey by the World Wide Web Foundation and the World Association of Girl Guides and Girls Scouts found that 52% of young women and girls reported experiencing online abuse [88]. Although researchers are trying to identify solutions that prevent abuse [32, 68, 85], the implementation remains cursory. Combating such abuse requires complex socio-technical solutions and a comprehensive shift in the way we design and use technology [16, 34, 87].

Technology-facilitated abuse (TFA) can be harmful for teenagers [86], women [59, 66, 82], and LGBTQIA+ population [52]. TFA has become a new tool for surveilling and harassing victims of intimate partner abuse.

TFA in Intimate Partner Violence. Intimate partner violence (IPV) encompasses physical, sexual, and psychological aggression by current or former partners [25]. Abusers often exploit their deep knowledge of victims, including personal details like birthdate or address, and they may coerce victims into revealing credentials [38]. Prior research [17, 38, 51, 73] highlights various abuse forms, such as financial abuse [20, 49, 62], non-consensual intimate imagery [60], stalking [69, 70, 73], and harassment via smart home devices [10, 27, 54, 72, 74–76]. Researchers address IPV through analyzing abuse factors [22, 23, 26, 39, 56], developing detection tools [64, 65], and supporting victims with social resources [37, 71, 76].

Freed et al. and Chatterjee et al. show abusers misuse spyware or *dual-use* apps, originally designed for benign purposes like locating lost phones, to spy on and harass victims [28, 38]. These tools enable remote access to sensitive information without physical proximity [9, 28, 74]. While prior work focuses on third-party mobile apps, we demonstrate that abusers can exploit automation tools, which leverage high privileges or built-in OS features, to facilitate IPV.

Real-world TFA Threats from Automation Apps. Two authors volunteer at a local organization that directly provides support to IPV survivors who are experiencing tech-facilitated abuse. The organization has worked with over a hundred clients in the last three years. They have seen clients in the past where the Shortcuts app was used to harass victims. On one occasion, Shortcuts was used to repeatedly close certain applications on launch, such as Google Maps. This was probably a parental control recipe but was repurposed to prevent the survivor from using several apps. Following up on this, we searched online with queries such as "spy using iOS Shortcut." We found several posts on TikTok, Reddit, and Weibo discussing how to use automation applications for malicious purposes like surveillance and harassment, with thousands of comments and millions of views. We also found posts describing users' concerns about a shortcut to be malicious. On a Reddit post [44], the maintainers of RoutineHub — an unofficial Shortcuts recipe repository — request users to report malicious shortcuts, underscoring the need to understand the abuse potential of automation apps. Although the current prevalence of automation apps in IPV contexts remains unclear, both our firsthand experience and emerging online discussions indicate their potential to become a significant new vector for abuse.

2.3 TFA Threat Model

We consider the threat from an abusive intimate partner who is trying to use technology to harm their victim. Unlike other security works that focus on purely machine-side vulnerabilities, we take a human-centered threat modeling perspective [80], focusing on constructing models grounded in the lived experience of real users, particularly those from vulnerable populations (*e.g.*, IPV survivors).

We specifically consider a *UI-bound intimate partner attacker*, which was first introduced by Freed et al. [38] and referenced in later research to analyze the privacy and security risks in IPV settings [21, 28, 64, 75, 79]. This adversary is limited to interacting with the device through its standard user interfaces. However, they often already know the passwords or PINS to the victim's mobile device, or can coerce the victim to unlock their devices [20, 28, 38]. With such authenticated physical access to the victim's mobile phone, the attacker can configure automation apps to surveil and harass the victim. Attackers are also able to communicate with the victim through phone calls, text messages, WhatsApp, and other platforms.

We argue that these attacks are accessible even to non-technical users, as malicious automation recipes can be easily

installed with brief physical access to the victim's device. Attackers need not develop the recipes themselves — they can reuse those shared online, often accompanied by step-by-step tutorials from Google searches (*e.g.*, "spy using Shortcuts on iPhone"), social media posts, and even official app web pages. For example, Tasker's official website includes guidance on hiding or disabling app notifications. Because automation apps are designed for benign utility, these recipes appear legitimate and approachable.

TFA attacks. IPV adversaries frequently use technology to abuse their victims, control them, and perpetuate power over them. We consider four types of attacks from the taxonomy by Thomas et al. [77] that can be achieved through compromising the victim's devices: Surveillance, Impersonation, Overloading, and Lockout & Control. We do not consider the other three types of attacks: sending Toxic Content to the victim, Content Leakage — sharing victim's information without consent — and False Reporting the victim to authorities, as they are typically executed without compromising the victim's account or devices.

Surveillance. Surveillance and stalking refer to the act of secretly monitoring or collecting information about a victim's activities without their consent. For instance, an attacker might configure a recipe to get a copy of the victim's incoming messages without being noticed.

Impersonation. An attacker can impersonate the victim and misrepresent them to others. For instance, an attacker can set a recipe with the action to send an email from the victim's phone with the content and recipient address received from the attacker via a text message.

Overloading. An overloading attack refers to either repeatedly forwarding information through automation apps to the victim or manipulating specific functionalities on the victim's device at a high frequency, potentially resulting in denial of service. For instance, the attacker can make the victim's device output 1,000 notifications at once.

Lockout/Control. A lockout/control attack allows attackers to manipulate the victim's device's physical features or software controls to gaslight and confuse the target, causing psychological distress and making the target feel unsettled. One example attack will be when the attacker changes the setting or deletes the information without the victim's notice and causes harm.

3 Analyzing Capabilities of Automation Apps

First, we answer RQ1 *(What are the capabilities of popular automation applications?)* by studying popular automation apps on iOS and Android, and extracting their capabilities.

App name	Platform	Rating	Pricing
Shortcuts	iOS	4.4 (17K)	Free
Tasker	Android	4.5 (54K)	$3.49
IFTTT	Android, iOS	4.6 (57K)	Free/$3.99
Bixby Routines	Android (Samsung Only)	1.5 (N/A)	Free

Figure 2: Four popular automation apps analyzed in this paper.

3.1 Automation App Selection

We analyzed the functionality of popular automation apps on Android and iOS to identify the basic automation operations they offer. As we demonstrate later, composing these operations can enable a range of attacks. Our method is as follows.

We searched for the keyword automation in the app stores and chose the applications with the highest number of reviews. We then downloaded the top 10 apps for each platform and verified whether they provided standalone automation functionalities—that is, automation features that do not simply redirect to another automation app.

From this selection, we chose the highest-rated, independent automation apps for each mobile operating system: Shortcuts on iOS; Tasker and Bixby Routines on Android; and IFTTT, which is available on both platforms. Fig. 2 presents the metadata for these four apps.

- **Shortcuts.** The Shortcuts App is installed by default on iOS 13.0 or later. We analyze and test this app on two physical iPhones: iPhone 11 with iOS version 17.4.1 and iPhone 15 with iOS version 18.0.1.
- **Tasker.** Tasker could be downloaded from the Google Play Store. We analyze Tasker (v6.2.22) on a Samsung Galaxy S10 running Android 12 and a OnePlus Nord N10 running Android 11.
- **IFTTT.** For IFTTT (v4.83), we evaluate the Pro version, which has access to the IFTTT voice mailbox. The more expensive Pro+ version provides access to scripting functionality.
- **Bixby Routines.** We evaluate Bixby Routines on a Samsung Galaxy S10 running Android 12. It is available for and preinstalled in selected Samsung mobile devices released after 2019, such as S10, S20, and Note10.

3.2 Capability Analysis

Automation apps offer their functionality in the **"Trigger→Action"** pattern. We define "Capability" of an automation app as the set of triggers and action(s) functionalities it supports. We conducted a comprehensive UI stepthrough analysis for each automation application to assess its capabilities. UI stepthrough, adapted from cognitive walkthrough methods [53], has been used in prior

Trigger	Sh	Ta	IF-i	IF-a	Bx
Internal events					
Geofence	Y	Y	Y	Y	Y
Battery Level	Y	Y	N	Y	Y
External events					
Receiving Communication[1]	Y	Y	N	Y	Y
Preset time					
Time in a Day	Y	Y	Y	Y	Y
User actions					
App Usage	Y	Y	N	N	Y
New Image/Recording	Y	Y	Y	Y	N
Notes Created	Y	Y	N	N	N
Calendar/Reminder Created	Y	Y	Y	Y	N
Device Usage	Y	Y	N	N	Y

[1] Trigger fires when the device receives an incoming message, call, or similar communication.

Figure 3: Different *triggers* supported (Y) or not supported (N) by automation apps grouped by their types.

Resource	Sh	Ta	IF-i	IF-a	Bx
Vibrate	Y	Y[a]	N	Y	Y
Flashlight	Y	Y[a]	N	N	Y
Screen Lock	Y[b]	Y	N	N	Y
Screen Content	Y[b]	Y	N	N	N
Play Sound	Y	Y	N	N	Y
Airplane Mode	Y	Y	N	N	N
Bluetooth	Y	Y	N	N	N
WiFi	Y	Y	N	N	Y
Notification	Y	Y	Y	Y	Y
Screen Recording	Y	Y	N	N	N
Clock	Y	N	N	N	Y

[a] Requires third-party plug-in and/or rooting.
[b] Phone must be unlocked.

Figure 4: Different *device & system control* actions supported (Y) or not supported (N) by different automation apps.

works [21, 38] to identify the system's features from a user perspective and to build a comprehensive inventory.

UI-Stepthrough. Our UI-Stepthrough analysis begins with installing the app (if it is not preinstalled) and familiarizing ourselves with the process of building a recipe on the app. This process often involves selecting a set of triggers and actions. Thus, we note down all the triggers offered by the platform (*e.g.*, "On Email Receive", "On New Notes created") along with the options supported by the app for each trigger (*e.g.*, Email sender, Title contains, etc.).

We then look into the set of supported actions. Recipes are sequential combinations of actions that can be executed on certain triggers. Actions can be from the system (*e.g.*, get device location), third-party apps (*e.g.*, add to Google Drive), or logic operators (*e.g.*, if…else, repeat), or they can even be previously created (or downloaded) recipes. For each action, there may be configurations (*e.g.*, the specific value to set the volume). In iOS Shortcuts, a sequence of actions is called a "shortcut," while a shortcut with triggers is called an "automation." Both Shortcuts and Tasker natively support Turing-complete logic, whereas IFTTT achieves Turing completeness via JavaScript (available to Pro+ subscribers). Bixby Routines, by contrast, are limited to simple one-shot if-this-then-that logic.

Feature	Sh	Ta	IF-i	IF-a	Bx
Capture (live)					
Record Photo	D	D	–	–	D
Record Video	U	D	–	–	U
Record Audio	U	A	–	–	–
Location	A	A	–	–	–
Retrieve (local)					
Photo Album	A	A	–	–	–
iCloud	A	n/a	–	n/a	n/a
Cloud Storage	A	A	A	A	–
Communication Log	–	A	–	–	–
Notes	U	–	–	–	–
Calendar/Reminder	A	A	–	–	–
Device Info	D	A	–	–	A
Files	U	A	–	–	–

Figure 5: Different *content & data management* actions supported by automation apps. Some actions always work (A), while others depend on device-state (D), such as device unlocked or app opened, or require explicit user interaction (U), and some are not supported (–) by some apps.

Method / Entry Point	Sh	Ta	IF-i	IF-a	Bx
Insertion					
App Input	Y	Y	N	N	N
Web Input	Y	Y	Y	Y	N
Message (SMS)	Y	Y	N	Y	N
Email	Y	Y	N	N	N
Exfiltration					
SSH	Y	Y	N	N	N
JavaScript	Y	Y	Y	Y	N
Nearby Share*	Y	Y	N	N	N
Message (SMS)	Y	Y	N	Y	N
Email	Y	Y	Y	Y	N
Social Media	Y	Y†	Y	Y	N

* "Airdrop" on iOS; "Quick Share"/Bluetooth on Android.
† Tasker requires the device to be unlocked and auto-taps UI buttons.

Figure 6: *Communication* methods supported (Y) or not (N) by different automation apps.

At this point, we have a list of triggers and actions for each automation app. We then group them into categories based on their functionality. We identified four types of triggers: (1) **Internal events** monitored by the device (*e.g.*, location change), (2) **External events** triggered outside the device (*e.g.*, receiving a message from other people), (3) **Preset time** (*e.g.*, afternoon at 1:00 PM), and (4) **User actions** (*e.g.*, user opens an application). The set of triggers is presented in Fig. 3.

We also identified four types of actions: (1) **Device and System Controls**: Actions that interact with or modify OS settings and on-device resources (*e.g.*, Set WiFi, Get Battery Level; see Fig. 4). (2) **Data Collection & Management**: Actions that capture, access, or modify local data such as photos,

media, files, or location (*e.g.*, Take Photo, Delete Last Photo, Get Current Location, see Fig. 5). (3) **Communication**: Actions that transmit information externally via SMS, email, HTTP, FTP, or SSH, or internally to other apps on the device, see Fig. 6. (4) **Automation Logic and Flow Control**: Actions that support control structures and variables (*e.g.*, If, Repeat, Wait). These are typically combined with other actions, so we did not evaluate them independently.

iOS Shortcut. The Shortcuts app has a comprehensive set of triggers and actions, and these pairings are composable. Given that it is Turing-complete, users can achieve any program that is feasible on one device with the permissions enabled (the user will be prompted for their first-time access) and with minimal additional user interaction (if the action itself does not need user interaction, such as "Choose from List").

Android Tasker. Tasker provides powerful capabilities similar to iOS Shortcuts but with even greater system access. It offers a Turing-complete environment that allows users to create complex automation with nested loops, conditional logic, and recursive task calling. Tasker can also package those routines as standalone APKs with custom names and icons for easy sharing. For a better user experience without additional friction, Tasker can utilize Android Debug Bridge (ADB) commands to bypass system notifications. Overall, Tasker excels at keeping track of device states, communications, and any sensor data, including location.

iOS IFTTT. IFTTT always begins with one single "If this" trigger and ends with one or more "Then that" actions. Unlike the preinstalled Shortcuts, most triggers and actions are external events (*e.g.*, when receiving a new email in Gmail). Only a few system/local events exist, including location, photo access, notifications, etc. The free tier runs one-shot if-this-then-that, and upgrading to Pro (which we mainly discuss in this work) unlocks a Webhook block that allows receiving of web requests (*e.g.*, JSON). The highest-tier Pro+ unlocks Queries for additional trigger parameters and a Filter Code block that is capable of running JavaScript. The automation on IFTTT depends on its running time (*e.g.*, users sometimes need to reopen the App to refresh the connection). Overall, iOS IFTTT focuses mainly on cloud services and largely depends on the APIs offered by third-party apps.

Android IFTTT. The Android IFTTT shares nearly the same but slightly higher capability as the iOS version while adding system-specific actions, such as triggers for WiFi and Bluetooth state changes, SMS receiving, phone call status, and battery level. The Android version also provides extra actions, such as setting the ringtone, toggling WiFi, or sending SMS messages.

Android Bixby Routines. Bixby Routines strictly follows one-shot if-this-then-that, and the automation will always end without looping. There are also limited actions and triggers compared to the other apps discussed above. The available actions cluster around system toggles (*e.g.*, sound mode), notifications, and limited content management (*e.g.*, launch apps).

4 TFA Attacks Using Automation Apps

We address RQ2 in this section: *What harms can the capabilities we discovered in Section 3.2 pose to IPV victims?* First, we present the feasibility of using automation recipes to conduct each of the four TFA attacks (Section 2.3) by combining the capabilities from Section 3. Second, we demonstrate that it is possible to emulate the functionalities of powerful mobile spyware using these automation apps on iOS and Android. We then show how existing abuse prevention and detection strategies are insufficient to protect IPV victims.

4.1 Attack Feasibility

We show that with one-time access to a victim's device, attackers can construct sophisticated attacks using only built-in triggers and actions. First, we present four proof-of-concept attack scenarios based on our TFA taxonomy to show how surveillance, impersonation, overload, and lockout attacks can be orchestrated with minimal user visibility. Then, we extend this proof-of-concept by emulating full-featured spyware (Flexispy) behavior through automation recipes to show that automation platforms can enable covert surveillance.

4.1.1 Proof-of-Concept TFA Attacks

We implement TFA attack scenarios that leverage the capabilities in automation apps. As we show in Section 5, automation recipes with similar functionalities already exist in the wild.

Surveillance. To surveil the victim, the attacker needs to collect information from the victim's device and exfiltrate the information out of the device; the process needs to be done in a stealthy manner without user consent. There are triggers and actions in automation apps to access local data (*e.g.*, actions in Data Collection & Management), and actions in Communication allow exfiltration of the data, some without any traces (*e.g.*, send via HTTP post request). We can also set values for triggers and actions to prevent notification to the victim (*e.g.*, Run Immediately, turn 'Preview' to false). For example, we present the following iOS Shortcut attack to spy on the victim's photo: The trigger is set to be a fixed time; run the following actions when the trigger fires: get the last five photos, make a zip file over it, attach the archived file to the preset URL, and use the "Get Contents of" the URL action to make an HTTP POST call to an attacker-controlled URL. No explicit logs or notifications are shown during this attack. The web traces and image access could be discovered in iOS's App Privacy Report if the victim has enabled the

feature. Note, this information is tagged to the Shortcuts app, and not to any specific recipe.

Impersonation. For an impersonation attack, the attacker needs to send information from the victim's device to the third party without the victim's consent. The actions in the Communication group can be utilized in this attack, while the triggers can be arbitrary. The attacker needs to set values for triggers and actions to proceed without user confirmation. We present the following example to send a malicious message to the victim's contacts: In IFTTT (either iOS or Android version), select the trigger to be the specific time, select the action to send email, connect the victim's account through the installed Gmail application, preset the contacts to be victim's contacts, and input the malicious message. When the trigger is fired, the message will be sent automatically. The only trace is that IFTTT will record this automation run.

Overloading. Overloading attacks repeatedly trigger settings or actions on the victim's device, causing a denial of service. In this case, only automation apps that are Turing-complete are capable of running loops. Specifically, actions from Device & System Controls or Communication, which explicitly affect devices and cannot be interrupted, are capable of conducting this attack. We present an example from Android Tasker: pick arbitrary triggers (*e.g.*, when the display is on), add a while loop with the desired condition, and execute actions that cannot be interrupted inside the loop (*e.g.*, toggle Wi-Fi). The action causes a problem when the victim attempts to use the Wi-Fi service for a continuous time period. Tasker allows users to disable run logging. Therefore, the attack leaves no obvious traces.

Lockout/Control. For lockout/control attacks, the goal is to cause confusion or anxiety. The attacker can manipulate the actions in Device & System Controls to silently modify system settings. We show an example in the Bixby Routine to perform a series of actions to scare the victim: Pick a trigger, such as when Wi-Fi is disconnected, run actions to lock the phone, turn on airplane mode, flip the visual themes, and set the media to 0% volume. The full action chain executes without additional confirmation. While Samsung logs routine executions in a history panel, it is hard to trace if the victim does not have prior knowledge about automation apps.

4.1.2 Emulating Spyware through Automation Recipes

We further present the abusability of automation apps by emulating spyware functionalities. Using Flexispy [36] — a powerful spyware app linked to IPV cases [28] — as a baseline, we replicate key features such as location tracking, media capture, app monitoring, and app blocking. Flexispy is not available in official stores and is incompatible with iOS 17+. We show that recipes can emulate Flexispy while leaving little to no trace, enabling covert surveillance and harassment. A detailed comparison with Flexispy is in Appendix A.

Category	Sub-type	Description	Sh	Ta	IF-i	IF-a	Bx
Notification	Accessing Data	Appear when accessing information	N	1st Acc.	N	N	n/a
	Extracting Data	Permission asked to start action	P	N	N	N	N
		Progress bar shown	P	N	N	**Y**	N
		User interaction needed	P	n/a	N	P **	Y
	Execution	Notification during execution	**P**	N	**Y**	**Y**	N
		Summary of app activity*	**Y**	n/a	n/a	n/a	N
		Summary of how data used	N	Loc. Use	Y	**Y**	N
Logs	System	Log of app activity in settings	Y	Y	Y	Y	N
	In-App	Log of automation run in app	N	Y	Y	**Y**	Y
	Communication	Message that sent to attacker	**Y**	**Y**	**Y**	**Y**	N

* Only available on iOS and Samsung devices. ** Denotes action required by OS (auto-clicking). **Bolded Text** denotes that the trace can be bypassed.

Figure 7: *Notifications and Logs*. Y = app leaves notifications or logs, P = only for some actions there will be notification, and N is otherwise. 1st Acc specifies notifications triggered only during the first data access, and Loc. Use applies only when location data is used. n/a = the notification or log type is not applicable for the app.

iOS Shortcuts. We replicated 31 (22 fully, 9 partially) features out of 45 features in Flexispy that do not require third-party apps. Our demo Shortcut recipe can send spoofed SMS/Email, track location, extract contacts, take photos/videos, and exfiltrate files – all running in the background with minimal trace after basic configuration. Advanced features like keylogging and SMS deletion were not possible with iOS Shortcuts.

Installing such recipes is straightforward: an attacker only needs to share an iCloud URL. Configuration involves setting a command trigger (*e.g.*, SMS, Email, or scheduled) and an exfiltration URL. We used public hosts for anonymous data collection. When using SMS or email, attackers must anonymize their address to avoid detection.

Android Tasker. Similar capabilities can be replicated on Android using Tasker. Unlike iOS Shortcuts, Tasker uses separate task profiles for each function. On non-rooted devices, we replicated 8 out of 13 Flexispy features, including call/SMS monitoring, camera control, contact/media access, app activity tracking, and custom alerts. Tasker also enables exporting tasks as standalone APKs with custom names and icons, obscuring their true function. These APKs can be sideloaded like any other app and run silently in the background without running Tasker itself, making detection more challenging. For example, a location tracker could be disguised as a weather app with minimal permissions.

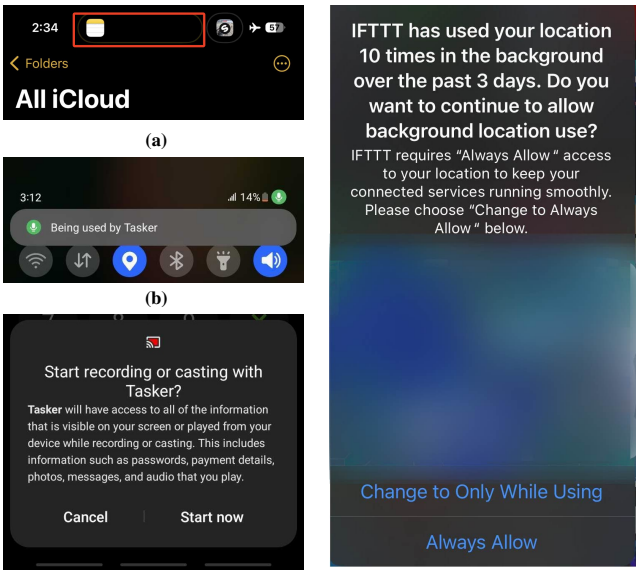

Figure 8: **(a)** Progress bar for a shortcut on iPhone 15 in the dynamic island. **(b)** System-enforced notification on Android 12. **(c)** Required user permission to start screen recording by a Tasker recipe on Android 12. **(d)** Platform notification by iOS for IFTTT's background location usage.

4.2 Undermining Abuse Prevention and Detection Mechanisms of Automation Apps

We analyze how the current abuse prevention and detection strategies are not effective against automation-based attacks even when the victim's device is not rooted or "jailbroken," We summarize the different notification and logging mechanisms that could potentially detect malicious recipes for automation apps in Fig. 7 and highlight the ones that can be easily bypassed.

Permissions. Mobile OSes rely on trusted app sources and runtime permissions to prevent malicious apps. Since automation apps are installed from official stores or preinstalled in the system, and permissions are granted at the app level — not per automation recipe— the system assumes trust once access is approved. For example, a benign recipe may gain permission that is later exploited by a malicious one.

Tasker further circumvents protections by using advanced permissions typically unavailable to regular apps. It leverages ADB commands to extend its privileges, such as modifying system settings. For instance, the command `adb shell pm grant net.dinglisch.android.taskerm android.permission.WRITE_SECURE_SETTINGS` allows Tasker to change system settings silently, avoiding persistent notification prompts.

User Consent. Another layer of defense is the runtime consent mechanism implemented by automation apps like iOS Shortcuts. These mechanisms are designed to prompt the user for approval before executing potentially sensitive actions. For example, if a Shortcuts recipe attempts to initiate a video recording, the system will display a permission prompt that the user must manually confirm to start or stop. Again, this protection is limited in scope, as consent is only required for a narrow set of actions deemed sensitive by the platform. Many other actions, such as accessing contacts, sending network requests, and modifying device settings, can often proceed silently after a one-time consent.

App-Based Notification. App notifications by the automation app (and the third-party apps it uses) are a way to inform the user about the execution of certain automation recipes. However, for iOS, the third-party app (*e.g.*, IFTTT) notifications are limited to banners, which can be easily disabled via OS settings. Shortcuts also have multiple notification formats: the newer iPhone models (iPhone 14 Pro and later) show a minimized progress bar in the Dynamic Island (an interactive interface element that integrates notifications and alerts into the pill-shaped cutout at the top of the screen), while the older models show a normal banner. While by default, iOS does not allow switching off Shortcuts notifications, we find a method online to disable Shortcuts notifications completely through the Screen Time settings [19]. Regardless of the notification settings, a banner/progress bar appears on the screen when the user manually runs a shortcut.

We argue that both designs fail to adequately alert users about a shortcut running in the background. For reference, Fig. 8 (a) shows a progress bar for a shortcut that "opens the Notes app, waits 5 seconds and takes a screenshot." On older iPhones, the larger banner includes only a button to halt the shortcut, offering no context. On newer devices, the progress bar appears only for a subset of actions.

For the Android system, once the user initially accepts required permissions and has disabled notifications for an automation app, most operations proceed without alerts. Exceptions include privacy-critical actions, such as accessing the camera, microphone, or screen, as shown in Fig. 8 (b), while some of these can be bypassed.

System-Level Alerts. iOS system provides pop-up notifications, showing app activity (for Shortcuts) and data usage (for IFTTT). These pop-ups highlight how often apps have run or accessed critical information (*e.g.*, location) over recent days, helping flag potential abuse. Fig. 8 (d) shows an example of such a pop-up. Similarly, Android provides periodic system reminders to notify users about sensitive permissions (*e.g.*, screen control). Apps running long-duration background tasks require persistent, non-dismissible notifications unless the service stops (though they can be hidden). In addition, Android also has system-enforced notifications that are mandated for certain types of app behavior, such as location access (Fig. 8 (c)).

Third-party Malware Detector. We tested three popular

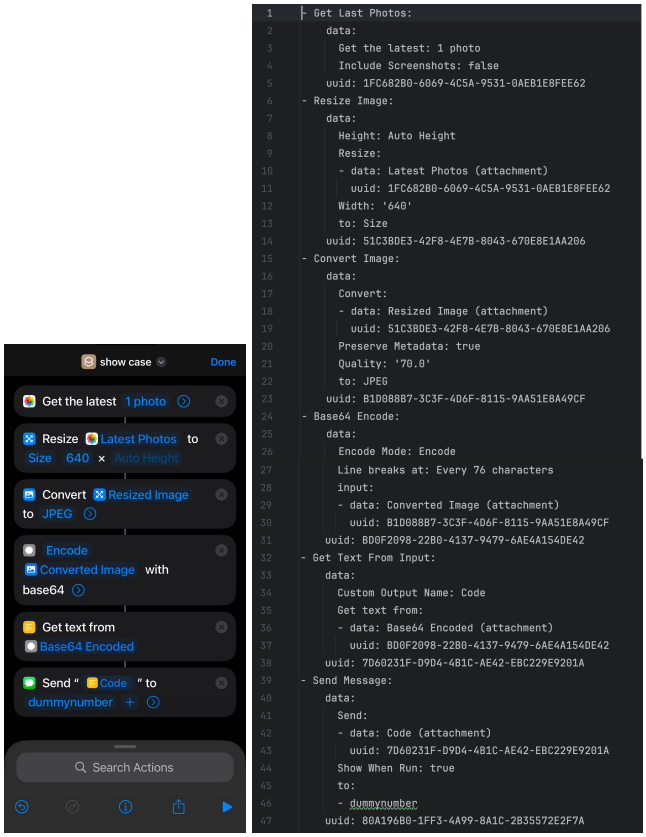

Figure 9: A Shortcut recipe (**left**) and it's YAML/Text representation (**right**).

malware/spyware detectors for both systems (Norton, iVerify, MalwareBytes for iOS; F-Secure, Avast, AVG for Android) to see if they are capable of detecting malicious recipes. We run a malicious recipe (uploading a photo to a website without user notice) on the test device, then run the analysis feature on the detector apps. We find that while none of the tested detection apps flag Shortcuts or their corresponding recipe.

System-level Reporting. Both iOS and Android provide system-level reports. iOS provides a notification activity summary per app and privacy reports (not turned on by default) in system settings, including data/sensor access logs for different apps. Android provides system logs and usage indicators to track sensitive actions, like accessibility services or critical permissions. On the application side, most automation apps (except iOS Shortcuts) provide in-app logs. All logs discussed above are non-bypassable. However, they are hard for victims to discover and analyze. From the victim's perspective, if they do not have prior knowledge that the attacks are from automation apps, it is hard to trace back non-real-time events in system logs: the sensors record every access of the application every minute, increasing the difficulty for analysis. Also, sensor logs are only kept for the past seven days on both OSes.

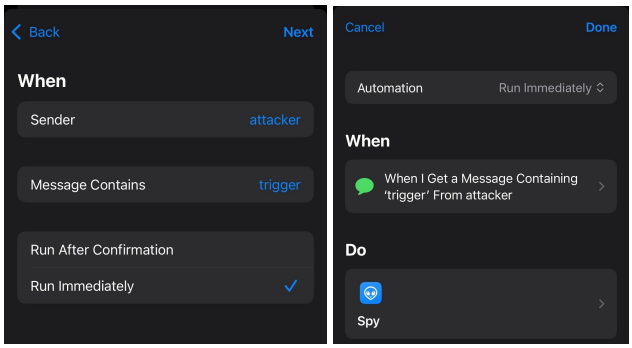

Figure 10: Trigger (**left**), Automation (**right**) in iOS Shortcuts.

5 Detect Abusive Automation Recipes

As we show in the previous section, automation apps can be used to design recipes with powerful surveillance and harassment capabilities, and the current detection is not sufficient to capture them. In this section, we answer RQ3 (*How to detect abusive automation recipes to defend affected victims?*) by designing a Large Language Model (LLM)-powered detector to measure the prevalence of recipes with such capabilities. We focus on Shortcuts, as it allows convenient sharing of recipes using iCloud links, and we found evidence of Shortcuts recipes with surveillance and harassment capabilities (See Fig. 1). We also found repositories of publicly available Shortcuts recipes that users can easily download and install.

The iOS Shortcuts app provides two types of functionality: a recipe and an "Automation", which automatically runs these recipes with preset triggers. A Shortcuts recipe is a combination of different Shortcuts **Actions** and can be exported and shared via iCloud links. Distributors can include import questions that prompt users for information; their responses are then automatically filled in corresponding fields within the Shortcuts. A recipe can also be exported as app icons with custom images and configured to run while the device is locked. We include an example of a recipe and its text representation in Fig. 9. Actions and their categories are detailed in Section 5.3.

An **Automation** consists of a trigger and a sequence of Shortcuts actions or a Shortcuts recipe. When the trigger activates, the actions execute sequentially. Triggers can be set as 'Run After Confirmation' or 'Run Immediately.' An example of a trigger setting and automation is shown in Fig. 10.

5.1 LLM-Powered Detector

We design an LLM-based detector to help determine if a recipe poses TFA threats from the taxonomy in Section 2.3.

Prompting an LLM to directly map a recipe to an attack is prone to errors and hallucinations, especially for smaller and open source models [45]. We address this problem in two steps. First, we abstract each attack from Section 2.3 into a

sequence of attack operations, providing a more precise definition of the attack. Then, we prompt the LLM to progressively identify attack operations and map sequences of operations to the attack, thereby reducing the hallucinations.

5.1.1 Attack Specification

Attack Operations. We analyze the comprehensive set of triggers and actions that automation apps offer through the lens of MITRE's Mobile Tactic [15], identifying abstract operations that could enable the attacks from Section 2.3. We arrive at a set of five operations needed to build an automation-based attack: *data collection* – TA0035 (Collection), *data exfiltration* – TA0036 (Exfiltration), *data insertion* – TA0028 (Persistence), *resource control* – TA0037 (Command and Control), and *trace hiding* – TA0030 (Defense Evasion).

- *Data Collection:* refers to the attacker's ability to retrieve information, using content and data management actions, from the victim's device. This operation can be preceded by any trigger.
- *Data Insertion:* refers to the attacker injecting internal or external information into the victim's device. Attackers can exploit entry points in automation apps (*e.g.*, Email/Message Content) to transmit data.
- *Data Exfiltration:* refers to transmitting information generated during data collection or insertion to attackers, victims, or other parties. This transmission utilizes actions such as messaging, emailing, social media posts, or even SSH.
- *Resource Control:* refers to controlling software or hardware resources on the device. For example, an attacker can control an output interface to overload the user.
- *Trace Hiding:* refers to the attacker's ability to hide its identity or operation from the victim.

Attack Instantiations. Each operation has different instantiations depending on the context, data, or attacker intentions. We then represent these automation-based attacks as ordered sets of instantiated operations.

- *Surveillance :* To perform a surveillance attack, the attacker first collects information they want from the victim's device through *data collection*. Then, the attacker exports the information through *data exfiltration* to a data sink under their control. The attacker uses the *hiding traces* operation to prevent the victim from being aware of the attack.
- *Impersonation.* To perform an impersonation attack, the attacker optionally performs *data insertion* to inject the content or recipient information, then does *data exfiltration* from the victim's device. The attacker transmits data from the victim's device to third-party recipients, who may be from the victim's contact list or specified by the attacker. This attack often involves sending messages from the victim's number to embarrass or harm the victim.

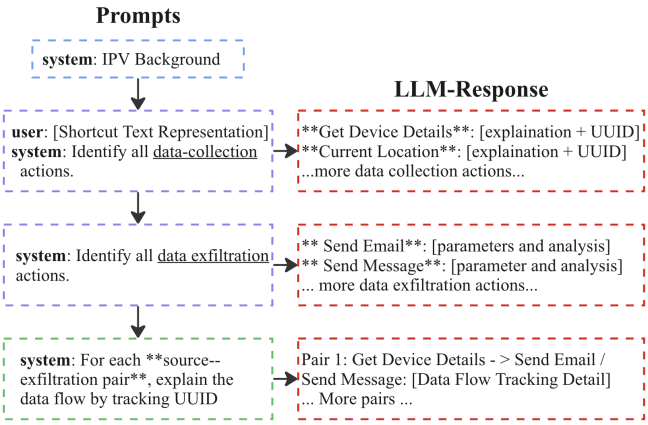

Figure 11: Example surveillance attack prompt design & LLM response. We design similar prompts for all TFA attacks.

- *Overloading.* To perform an overloading attack, the attacker needs to *repeatedly* perform *resource control* to manipulate functionalities or *data exfiltration*. The attacker can perform all functions in Fig. 4 or Fig. 6.
- *Lockout/Control.* To perform a lockout/control attack, the attacker performs the *resource control* operation without a clear explanation to the victim. The attackers can utilize software features like toggling the mobile hotspot, WiFi, and sending notifications, or engage physical features like phone vibration, lock screen, and switching flashlights.

5.1.2 Prompt Design

We design a *per-attack prompt* that leverages these attack specifications. For each attack, we start the prompt with (1) a general command that includes a summary of the IPV background, specific attack definitions, and corresponding operators. We then proceed to (2) capability extraction: the model will be asked to identify actions related to the attack operator(s); all other actions must be ignored in this stage. Using the actions fetched from (2), we ask the model to evaluate whether the recipe actually meets the attack definition. For example, as shown in Fig. 11, instead of directly asking if a shortcut constitutes a "surveillance" attack, we ask the following: identify all `data-collection` actions in the Shortcuts, identify all `data-exfiltration` actions, and then determine if there is an execution path between any collection-exfiltration pairs that exfiltrate collected data to a third party.

5.1.3 Prompting Strategy

We prompt LLMs, particularly Qwen2.5-Coder-32B-Instruct [45] and gpt-4o [61], with the text representation of the recipes, a per-attack prompt (from Section 5.1.2), and a system prompt. Akin to chain-of-thought prompting [31, 84], the system prompt instructs the LLM to first generate the rationale of step-by-step analysis on recipes, then analyze the

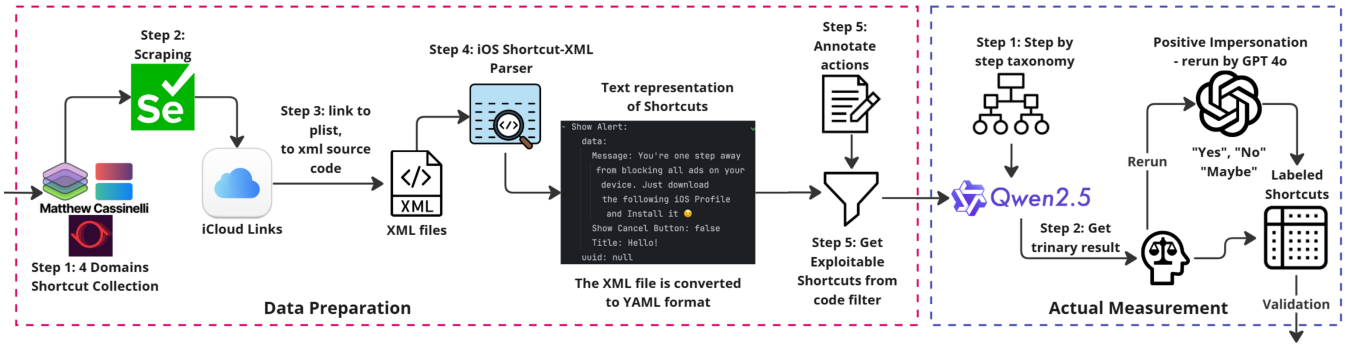

Figure 12: Our measurement pipeline to analyze publicly available Shortcuts recipes.

rationale to provide a trinary result: "Yes", "No", "Maybe" on whether a shortcut can perform a specific attack.

This final prompt proved robust in identifying surveillance, overloading, and lockout, while impersonation had a higher false positive rate. To refine our detector, we used `GPT-4o` to re-evaluate shortcuts flagged as positive for impersonation. Results show `GPT-4o` outperformed the previous model under the same prompt. In a randomly selected subset of 190 shortcuts, `Qwen2.5-Coder-32B-Instruct` identified 31 as impersonation, with manual validation revealing 12 false positives. `GPT-4o` correctly identified all 12 as non-impersonation (true negatives) while accurately classifying the remaining 19 as true positives.

5.2 Evaluate our Detector on Publicly Available Shortcuts Recipes

We collected a large number of iOS Shortcuts from publicly available unofficial Shortcuts repositories, which we analyzed using our measurement pipeline (Fig. 12).

Data Preparation First, we used Google searches with queries like "iOS Shortcuts" and "iOS Shortcuts Gallery" and recommendations from online forums [6] to identify four large galleries for Shortcuts recipes: Shortcutsgallery.com [8], MatthewCassinelli.com/shortcuts [3], RoutineHub.co [5], and ShareShortcuts.com [7]. These galleries provide (iCloud) URLs to Shortcuts recipes. We collected all URLs from these websites, downloaded the recipe files (in `plist` format) using a publicly available GitHub script [90], and converted them into XML files for easier analysis.

We removed duplicate iCloud URLs and the broken URLs that could not be converted to a valid XML. This step yielded a total of 13,745 valid XML files: 2,018 from Shortcutsgallery.com, 437 from MatthewCassinelli.com/shortcuts, 9,118 from RoutineHub.co, and 2,172 from ShareShortcuts.com. We attempted to find duplicate recipes across the domains by comparing names and functionality. We found $< 2\%$ shortcuts in our dataset that might be duplicates by comparing files with the same names. Shortcuts recipes with (near) identical functionalities can have different

Category	Flagged by Code Filter	Flagged by LLM		
		Yes	No	Maybe
Surveillance	371	21 (0.2%)	244	106
Impersonation	5,541	340 (2.6%)	4,499	702
Lockout/Control	3,067	656 (5.1%)	2,227	184
Overloading	479	65 (0.5%)	101	313
Total	7,616	1,014 (7.9%)	/	/

Figure 13: Classification of 7,616 shortcuts (out of 12,962) that are flagged as potentially exploitable by our initial filtering using the LLM-powered detection pipeline. Note some recipes are classified into multiple attack categories.

URLs; therefore, we also run a similarity test based on the actual actions in evaluated recipes. We used text embedding (using `sentence-transformers/all-MiniLM-L6-v2`) to compute pairwise cosine similarities of the Shortcuts recipe XML files, and our result finds less than 1% duplicates (similarity above > 0.95).

We then implemented a parser to convert Shortcuts recipe XMLs to the YAML format [89]. This format keeps the essential information and metadata (*e.g.*, uuid, Show When Run, etc.) for each action while reducing the repetitive XML format tags; an example is shown in Fig. 9.

Filtering potentially malicious shortcuts. To reduce analysis cost, we filter out the recipes that do not include all the attack operations required to instantiate an attack (see Section 5.1.1). For example, to conduct a surveillance attack, an recipe must include at least one `data-collection` operation and at least one `data-exfiltration` action. Starting from the set 13,745 YAML files, we identified 5,346 recipes that are not exploitable to perform any attack from our initial filtering. Of the remaining 8,399 recipes, 783 were longer than the `Qwen2.5-Coder-32B-Instruct`'s context window (128k with YARN configuration); therefore, we manually annotated those. There are no surveillance recipes identified, but we found some recipes capable of doing lockouts (70), overloading (10), and impersonation (12) attacks. The high number of lockouts is due to longer recipes, often including base64-encoded audio and actions to decode and play them, typically used to startle or scare victims. We consider

the remaining 7,616 recipes as potentially exploitable into at least one of the four attack categories and analyze them further using our LLM-based detector.

If a shortcut was flagged by the code filter for surveillance, we only tested it with the corresponding prompts; if it was flagged for two attack categories, we tested for both. Consequently, each recipe had a different number of LLM-generated answers. We find that our LLM-based detector flagged 1,014 recipes (13.4% of 7,616 recipes it has analyzed) as capable of conducting at least one of the attacks. We present the breakups and the analysis results in Fig. 13. We also included a breakdown of each domain in the Appendix B.

Manual Validation. We manually validate the LLM outputs by randomly sampling 500 recipes labeled as either "Yes" or "No" by the LLMs. This sampling yielded 160 recipes with Yes/No labels for Impersonation, 140 for Lockout, 100 for Overload, and 100 for Surveillance.

Two authors collaboratively annotated the IPV attacks in the selected validation set without seeing LLM responses, including a binary answer of 'Yes'/'No' for the given TFA attack and corresponding rationale. Two authors independently verified the same 100 of these shortcuts, achieving near-perfect agreement ($\kappa = 0.96$) [58]. We then compare the binary LLM response with our annotation.

We show the results of the manual validation in Fig. 14. The false negative rates for the impersonation, surveillance, and overload attacks are less than 5%. This result demonstrates that the LLM can effectively filter out non-attack recipes. The only exception is in the lockout category, where the LLM missed subtle setting cases where the recipe does not change the device settings (*e.g.*, airplane mode, phone volume, etc.) directly, but rather runs other actions such as playing music, or opening apps. The false positive rate is higher, which is due to our conservative definition (and subsequently labeling) of what constitutes an attack. If a recipe requires user interaction or shows a notification when run, we consider it as *not* an attack; however, LLM flags it as potentially capable of an attack. For lockout and control attacks, false positives occurred when LLM failed to account for the 'Show When Run' setting or the 'Speak Text' actions, which would notify users about exfiltrating actions. For overloading attacks, the false positives were instances where repetitive actions within loops were stoppable or did not impact normal usage. Note that modifying the per-attack prompt to account for such cases removes the false negatives and false positives.

We randomly selected 200 recipes labeled as 'Maybe' by the LLM and manually analyzed them. For all attack operations combined, approximately 12% were found to be malicious (*i.e.*, would be labeled as 'Yes'). The majority were labeled as 'No.' Notably, the *lockout/control* category had a higher proportion of malicious cases compared to others: *impersonation* (4/46), *surveillance* (2/48), *overloading* (4/46), and *lockout/control* (14/36).

Category	**No**	**Yes**	**False Positives**	**False Negatives**
Surveillance	86	14	9	0
Impersonation	141	19	7	0
Overload	59	41	7	3
Lockout	103	37	6	22
Total	389	111	29	25

Figure 14: Manual verification of 500 shortcuts flagged by LLM.

Category	Num. of Actions	Num. of Shortcuts Containing Action
Data-collection	71	9,849
Data-insertion	36	11,814
Data-exfiltration	33	11,936
Resource-control	138	11,314
Trace-hiding	6	2,697

Figure 15: Action Annotation and Shortcuts Coverage.

5.3 Findings

We use the measurement pipeline to analyze shortcuts across the four domains and examine the action distribution within the dataset. Our analysis yields the following findings.

Action Distribution. Our XML parser identified 982 actions, excluding those deprecated by system updates. To contextualize the data, we refer to our threat model, which focuses on analyzing default Shortcuts actions without requiring third-party applications. For example, the "Take Photo" action, along with the pre-installed "Camera" app, is available on all iPhone devices, whereas the "Open in Dropbox" action requires users to manually download the "Dropbox" app from the iOS App Store. In this section, we classify the former as *internal actions* and the latter as *external actions*.

There are 422 internal actions and 560 external actions. We analyzed the distribution for each action appearing across the four datasets and found that "If" (211,569 times), "Nothing" (127,666 times), and "Choose from Menu" (119,763 times) are the three most frequently appearing actions.

We annotated 422 internal actions based on the attack spec in Section 5.1.1, and the number of recipes containing those actions in all 13,745 shortcut representation files. The detailed action distributions are included in Fig. 15. Note that the number for `trace-hiding` might not reflect the full potential of an attacker trying to hide their trace, as there are settings to hide the action process from the user for many actions, such as "Show When Run", is not recorded in counting since they are included in action's metadata.

Surveillance Analysis. We identified 21 surveillance attacks using the LLM detector. For a recipe to be classified as positive, it must include a data collection action, and the collected data must be traceable to a data exfiltration action via their UUID. The data exfiltration is carried out in a stealthy manner (*e.g.*, the "Show When Run" option is turned off, or the message is base64-encoded). Negative samples are typically

filtered out because they contain actions that alert the victim, such as "Choose From List."

Most of the positive cases are publicly labeled under the *Utilities* or *Photo and Videos* categories. Interestingly, we observed that this category contains the fewest positive samples, which may be attributed to regulatory constraints [44].

Impersonation Analysis. We identified 340 impersonation attacks. The positive recipes usually have actions to exfiltrate the data from the device without consent from or a clear explanation to the victim (*e.g.*, Send Message, Send Email, or open a URL with data embedded into a link). There are many utility recipes with mini-menus (*e.g.*, Choose From Menu about next operation) identified as negative. While these recipes send data out of the device, they present text indicators about their actions and need user interaction to pass the data out. The top three categories for positive recipes are: *Utilities*, *Productivity*, and *API*.

Lockout/Control Analysis. We identified 656 recipes capable of executing lockout and control attacks, which can cause psychological harm to victims. These recipes perform various actions: (1) some decode and play a long base64-encoded audio file, (2) modify system settings that visibly affect the victim's device, and (3) display incomprehensible and/or incessant notifications. A common characteristic among these patterns is the presence of unexpected actions that directly impact the victim, although some could still be classified as utility recipes.

Negative samples typically include text-based hints or request user permissions via pop-up windows, which reduces their potential for harm. The top three app categories are consistent with those found in impersonation attacks; however, we observed a growing number of recipes labeled under *Fun* and *Entertainment*.

Overloading Analysis. We identified 65 recipes capable of performing overloading attacks. A recipe is classified as such if it satisfies the following conditions: (1) the presence of a loop, (2) at least one action within the loop directly affects the victim's device, (3) the loop causes a repeated change of state on the device, and (4) there are no actions requiring user interaction that would interrupt the loop.

Some recipes include loops but are excluded from this category because they either prompt the victim to make a choice from a list or contain looped actions that do not visibly affect the device. The most prevalent categories for overloading attacks are *Utilities*, *Fun*, and *Game*.

5.4 Recipes Explicitly Intended for Abuse

We analyze the recipes that are explicitly designed for abusive purposes. We use search terms based on prior work and blog posts [9, 28, 30]. The search terms are a combination of 12 verbs (`track`, `spy`, `monitor`, `locate`, `follow`, `hack`, `read`, `record`, `forward`, `bomb`, `spam`, `crash`) and 10 target nouns (`wife`, `husband`, `girlfriend`, `boyfriend`, `partner`, `spouse`, `fiancé`, `fiancé`, `ex`, `cheater`). We query these $12 \times 10 = 120$ search terms on Google and on public Shortcuts repositories like `routinehub.com`, and record the responses that contain any of the search terms in their title, tag, or description, and ones that are potentially designed for malicious intent (*e.g.*, 'Crash Device,' 'Want to be a spy? Well now you can…'). Although we did not find any recipe explicitly marketed for IPV, we found 91 recipes associated with these abusive keywords. We removed the recipes with broken links and then removed the duplicate ones using our deduplication process (explained in Section 5.2). Finally, we have 66 unique recipes showing abusive intent.

Manual Analysis. We then manually labeled the 66 recipes based on their recipe procedure, following the process described in Section 5.2. Our manual analysis of the 66 recipes shows that only a few recipes are capable of producing IPV-related attacks. Most of the recipes we found clustered around keywords "spam", "bomb", and are intended to send hundreds of messages from *attack* devices to a specific contact. While they could be harmful, these recipes are out of the scope of our work, as we study recipes installed on victim devices. For the ones left for attackers to "send to their enemy" [43], although a lot of them have the capability of causing harm, they usually have a "caveat" (*e.g.*, Show notifications of 'Are you really sure? This could cause actual harm.') before proceeding with abusive actions. We denote those recipes as "Yes if without caveat" in labeling, by including those, the final abusive recipes are: 11 lockout/control, 5 overloading, 3 impersonation, and 0 surveillance. The negative samples are mostly because they require user interaction, such as "Choose from list", which clearly notifies the victim about automation running. The rest of the 52 recipes had no attack capabilities.

Running LLM Detector on the Recipes. We pass the same 66 recipes through our pipeline, following the five data preparation and the two measurement steps from Fig. 12. Comparing the LLM-powered detector with the human labeling yielded two false positives and two false negatives. The false positives — one lockout and the other is impersonation — contain a notification or an alert that explains the purpose of the recipe. However, LLM considers them to be insufficient for the users. In the false negative cases (also one lockout and one impersonation), although the LLM recognized potential harm in its rationale, it labeled them as 'No' rather than 'Maybe'. There are 11 'Maybe' from LLM, while 3 of them are labeled as 'Yes' by the human annotator.

5.5 Reporting Safety Concerns

After mapping capabilities and analyzing shortcuts in public domains, we reported safety concerns about automation applications in an IPV setting to the application carrier and public domain manager in September 2024. While the domain manager did not respond, Tasker and Apple (iOS Shortcuts)

provided feedback. Tasker acknowledged the concern but emphasized that their functionality prioritizes utility and user experience, noting that notifications may not directly alert victims if attackers intentionally misuse the app. Apple's product security team stated that the described attacks require physical device access and recommended using strong passcodes to prevent unauthorized access. However, such protections are ineffective under the IPV threat model, as the attacker can have authenticated physical access to the victim's device.

6 Discussion

Our research demonstrates that automation apps can be exploited for surveillance and harassment, particularly in IPV scenarios. However, platform developers remain unaware of this threat and the harm their tools may cause IPV survivors.

Automation apps represent a new class of "dual-use" tools [28] — not inherently malicious, but capable of enabling abuse. They are designed to empower users by creating complex workflows and automating tasks. To enhance usability, these apps often minimize runtime permission prompts and provide features like "Run Immediately" in Shortcuts and background profiles in Tasker [63]. However, as we demonstrate, these features can be weaponized in IPV contexts.

This presents a critical design dilemma: *how can we safeguard victims from abuse without undermining the legitimate utility of automation apps?*

Our findings suggest that how platforms abstract recipes contribute to their potential misuse. Platforms deploy notifications, permissions, and checks at the app level, treating recipes as merely "in-app configurations." For example, the user grants permissions to the Shortcut app, not the individual recipes. Also, Android's privacy dashboard and iOS App Privacy Reports list permissions and recent sensitive data accesses for each app, and it does not make any distinction for recipes. As a result, users have limited visibility into the execution of automation apps. Our main insight is that platforms, and subsequently users, have to treat recipes similarly to apps in terms of threat identification, protection, detection, and response [29].

Identification: Identify malicious recipes. Automation recipes repositories should regularly check if (potentially) malicious recipes are being distributed through their platform. We provide a recipe detection pipeline in Section 5. This can be directly used for analyzing public recipes. Similarly, mobile platforms should also have an in-OS monitoring system for installed recipes, which flags potentially malicious execution patterns and warns users of such recipes on their device. Thus, it is crucial to ensure this detection pipeline is robust to adversarial manipulation [55]. Inspecting unpublished recipes–those created locally or shared privately–should be performed in a privacy-preserving manner, as these recipes may contain sensitive information, such as locations or API secrets. Our current detection pipeline requires the recipes to be sent to third-party LLMs, with limited control over how the data is used. Future research should look into how to conduct privacy-preserving malicious recipe checking on devices. Moreover, we need more research to find a user-friendly way to communicate the capabilities (and the risks) of dual-use recipes to the users (similar to privacy labels [11, 13, 40]).

Protection: Per-recipe permissions and checks. We suggest platforms should treat automation recipes as regular apps and apply the existing safeguards applied to regular mobile apps. This means that recipes should require separate permission from the user for their execution and are subject to regular background permission use checks — permissions are revoked if the user has not explicitly executed the recipe. Providing permissions at the recipe level might overburden users with runtime permissions, which leads to habituation [35]. There is a need to design these permission systems to balance between usability and safety. For example, runtime permission can be displayed only if the LLM-detector flags a recipe for potential misuse.

Detection: Detect if malicious recipes are installed. Based on our detection pipeline (Section 5), we have created a Docker image for a webservice[3] for checking if a shortcut recipe can be used for one of the four attacks we analyzed in this paper. We have also created a Shortcut recipe to export all recipes installed on an iPhone. All of these will be publicly available.[4] This web service can be used by survivor services such as Tech Clinics [1, 2, 4] for investigating tech-facilitated abuse. This webservice does not record any information; however, we use third-party LLMs; thus, while using this service, users should only check the recipes that they feel comfortable submitting for evaluation, avoiding the risk of exposing private information.

Response: Persistent recipes execution notification and logging. Our detection tool can be helpful for users who are already suspicious of TFA and proactively want to check their automation apps. However, for unsuspecting users, platforms need to carefully design notifications and consent mechanisms for automation apps to alert users of any potentially malicious recipes installed on their devices. Most automation apps have notification options to inform users that a recipe was executed; however, these notifications are relatively small and temporary, only appearing during execution. Moreover, all automation apps allow ways to turn off such notifications. Instead, we suggest using persistent banner notifications on locked and unlocked screens of the user's phone that link to the executed recipe and disappear only after user interaction. Notifications should be designed to be informative while avoiding notification fatigue.

[3] https://hub.docker.com/r/anninobear/shortcut_server

[4] https://www.icloud.com/shortcuts/34df135ad7be43799e8e8d10a6732fe8

In addition to notifications, apps should also preserve read-only, machine-readable logs of execution history for at least four weeks. Moreover, OS should also support evidence export, which includes signed logs that survivors could share with advocates, tech clinics, or legal services.

7 Conclusion

Automation tools are becoming increasingly valuable tools for simplifying tasks and enhancing efficiency while also giving attackers chances to perform IPV abuses. We deeply analyze four popular automation tools on both iOS and Android by presenting their capabilities and how the combinations of these capabilities are weaponized for spying and harassment. We present an LLM-powered detector and perform a measurement on the public Shortcuts collections, addressing the overlooked threat from these automation tools. We hope our research helps raise awareness of this growing issue of malicious automation.

Acknowledgment

We thank the reviewers and shepherd for their constructive comments on our work. This work is supported by the National Science Foundation under grant 2247381 and 2339679.

Ethics Considerations

This research involves two primary activities: (1) evaluating publicly available mobile automation applications, and (2) collecting and analyzing publicly shared automation "recipes" or scripts. Our goal is to understand the abuse potential of these tools in the specific and increasingly relevant threat model of interpersonal abuse, and to inform proactive strategies for harm reduction.

Research Artifacts. During our research, we identified and developed several automation recipes that could potentially be used for surveillance, harassment, or other abusive purposes. While these artifacts are valuable for understanding risk and for building future detection systems, they also pose potential for misuse. Our aim is to assess and raise awareness about such risks—not to enable abuse. To minimize harm, we do not share links to harmful recipes or include detailed methods for bypassing app restrictions in the paper or accompanying materials. We follow principles similar to those used in security research on malware or phishing, where risky artifacts are shared responsibly for scientific benefit but not publicly released. We make our dataset of potentially harmful recipes available only to verified academic researchers, upon explicit written request. Access to these materials is granted under the condition that they are not redistributed or used outside of a research context. Our institution maintains a formal data-sharing process to authenticate and vet requesting researchers. All shared datasets are accompanied by clear warnings about their potential for misuse and the expectation of ethical use only within academic and research boundaries.

Research Environment. Studying tools and scenarios related to interpersonal abuse can take a psychological toll. To support researcher well-being, the research team holds regular check-ins to discuss progress and emotional impacts. Two team members have formal training and professional experience working with survivors of intimate partner violence (IPV). All testing of potentially harmful recipes was conducted on secured, dedicated research devices. No personal devices or accounts were used, and no personal data was involved at any stage of the analysis.

Responsible Disclosure. We followed responsible disclosure practices by notifying relevant developers and platform operators — including Apple (Shortcuts), Tasker, and ShareShortcuts — about the abuse potential of recipes on their platforms. Our goal was to assist these stakeholders in recognizing risks and improving safeguards in their tools.

This research addresses an underexplored but pressing threat to vulnerable individuals through automation applications. We approach this work with careful attention to ethics: avoiding the dissemination of harmful technical details, restricting access to potentially abusive content, and prioritizing both researcher safety and survivor protection. We hope this work helps the broader community better recognize and mitigate the risks of automation abuse.

Open Science

We commit to openly sharing our research artifacts. In particular, we created a dataset that includes the XML-to-YAML parser, the recipes we have labeled, the model outputs on the evaluation datasets, and the scripts to replicate our results. We make these artifacts available at https://doi.org/10.5061/dryad.b2rbnzssm. We included pseudonyms instead of the real names of recipes in our public dataset for safety purposes. For all data access that includes mapping pseudonyms to the real names of recipes and the raw XML files of recipes, we will share a "restricted access" dataset with researchers who have been verified by our university's research ethics committee upon request.

References

[1] CETA: Clinic to End Tech Abuse. https://www.ceta.tech.cornell.edu/. Accessed: 2023-07-22.

[2] Madison Tech Clinic. https://techclinic.cs.wisc.edu/. Accessed: 2024-04-22.

[3] Matthew cassinelli - shortcuts. https://matthewcassinelli.com/shortcuts/. Accessed: 2024-11-21.

[4] NewBegin: Technology Enabled Coercive Control. https://newbegin.org/find-help/staying-safe/technology-safety/. Accessed: 2023-07-22.

[5] Routinehub - shortcuts directory. https://routinehub.co/. Accessed: 2024-11-21.

[6] r/shortcuts wiki. https://www.reddit.com/r/shortcuts/wiki/repositories/. Accessed: 2024-12-10.

[7] Shareshortcuts. https://shareshortcuts.com/index.php. Accessed: 2025-01-13.

[8] Shortcutsgallery.com - shortcut galleries. https://shortcutsgallery.com/. Accessed: 2024-11-21.

[9] Majed Almansoori, Andrea Gallardo, Julio Poveda, Adil Ahmed, and Rahul Chatterjee. A global survey of android dual-use applications used in intimate partner surveillance. *Proceedings on Privacy Enhancing Technologies*, 2022.

[10] Ahmed Alshehri, Malek Ben Salem, and Lei Ding. Are smart home devices abandoning ipv victims? In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1368–1375. IEEE, 2020.

[11] Amazon Developer Documentation Team. Amazon appstore privacy labels. https://developer.amazon.com/docs/app-submission/appstore-privacy-labels.html, 2025. Accessed: 2025-05-25.

[12] Apple. Shortcuts. https://apps.apple.com/us/app/shortcuts/id915249334.

[13] Apple Inc. Privacy labels. https://www.apple.com/privacy/labels/, 2025. Accessed: 2025-05-25.

[14] Deniz Arsan, Ali Zaidi, Aravind Sagar, and Ranjitha Kumar. App-based task shortcuts for virtual assistants. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 1089–1099, 2021.

[15] MITRE ATT&CK®. Mobile tactics. https://attack.mitre.org/tactics/mobile/.

[16] Jane Bailey and Jacquie Burkell. Tech-facilitated violence: thinking structurally and intersectionally. *Journal of Gender-Based Violence*, 5(3):531–542, 2021.

[17] Louis Bailey, Joanne Hulley, Tim Gomersall, Gill Kirkman, Graham Gibbs, and Adele D Jones. The networking of abuse: Intimate partner violence and the use of social technologies. *Criminal Justice and Behavior*, 51(2):266–285, 2024.

[18] Barbara Rita Barricelli, Fabio Cassano, Daniela Fogli, and Antonio Piccinno. End-user development, end-user programming and end-user software engineering: A systematic mapping study. *Journal of Systems and Software*, 149:101–137, 2019.

[19] Sam Beckman. Remove shortcut banners and hide the dock on ios 16! https://www.youtube.com/watch?v=eLQEnVC54DY, 2023.

[20] Rosanna Bellini. Paying the price: When intimate partners use technology for financial harm. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–17, 2023.

[21] Rosanna Bellini, Kevin Lee, Megan A Brown, Jeremy Shaffer, Rasika Bhalerao, and Thomas Ristenpart. The Digital-Safety risks of financial technologies for survivors of intimate partner violence. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 87–104, 2023.

[22] Rosanna Bellini, Emily Tseng, Nora McDonald, Rachel Greenstadt, Damon McCoy, Thomas Ristenpart, and Nicola Dell. " so-called privacy breeds evil" narrative justifications for intimate partner surveillance in online forums. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW3):1–27, 2021.

[23] Rosanna Frances Bellini. Abusive partner perspectives on technology abuse: Implications for community-based violence prevention. *Proceedings of the ACM on Human-Computer Interaction*, 8(CSCW1):1–25, 2024.

[24] Danah Boyd. Truth, lies, and 'doxxing': The real moral of the gawker/reddit story. https://www.wired.com/2012/10/truth-lies-doxxing-internet-vigilanteism/, 2012.

[25] Matthew Breiding, Kathleen C Basile, Sharon G Smith, Michele C Black, and Reshma R Mahendra. Intimate partner violence surveillance: Uniform definitions and recommended data elements. version 2.0. *Division of Violence Prevention, National Center for Injury Prevention and Control*, 2015.

[26] Nick Ceccio, Naman Gupta, Majed Almansoori, and Rahul Chatterjee. Analyzing the patterns and behavior of users when detecting and preventing tech-enabled stalking. 2023.

[27] Rose Ceccio, Sophie Stephenson, Varun Chadha, Danny Yuxing Huang, and Rahul Chatterjee. Sneaky spy devices and defective detectors: the ecosystem of intimate partner surveillance with covert devices. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 123–140, 2023.

[28] Rahul Chatterjee, Periwinkle Doerfler, Hadas Orgad, Sam Havron, Jackeline Palmer, Diana Freed, Karen Levy, Nicola Dell, Damon McCoy, and Thomas Ristenpart. The spyware used in intimate partner violence. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 441–458. IEEE, 2018.

[29] James Costanzo. What is the NIST CyberSecurity Framework?, September 2022. Accessed: 2025-05-27.

[30] Nicki Dell, Karen Levy, Damon McCoy, and Thomas Ristenpart. How domestic abusers use smartphones to spy on their partners. *Vox*, May 2018. Accessed: 2025-05-17.

[31] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3563–3578, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[32] Karthik Dinakar, Birago Jones, Catherine Havasi, Henry Lieberman, and Rosalind Picard. Common sense reasoning for detection, prevention, and mitigation of cyberbullying. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2(3):1–30, 2012.

[33] David M Douglas. Doxing: a conceptual analysis. *Ethics and information technology*, 18(3):199–210, 2016.

[34] Louise Ellison and Yaman Akdeniz. Cyberstalking: The regulation of harassment on the internet. In *Cyberspace Crime*, pages 275–293. Routledge, 2017.

[35] Adrienne Porter Felt, Serge Egelman, Matthew Finifter, Devdatta Akhawe, and David Wagner. How to ask for permission. *HotSec*, page 0, 2012.

[36] Ltd Flexispy. Flexispy. https://www.flexispy.com/.

[37] Diana Freed, Sam Havron, Emily Tseng, Andrea Gallardo, Rahul Chatterjee, Thomas Ristenpart, and Nicola Dell. " is my phone hacked?" analyzing clinical computer security interventions with survivors of intimate partner violence. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–24, 2019.

[38] Diana Freed, Jackeline Palmer, Diana Minchala, Karen Levy, Thomas Ristenpart, and Nicola Dell. "a stalker's paradise" how intimate partner abusers exploit technology. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–13, 2018.

[39] Diana Freed, Jackeline Palmer, Diana Elizabeth Minchala, Karen Levy, Thomas Ristenpart, and Nicola Dell. Digital technologies and intimate partner violence: A qualitative analysis with multiple stakeholders. *Proceedings of the ACM on human-computer interaction*, 1(CSCW):1–22, 2017.

[40] Suzanne Frey. Get more information about your apps in google play. https://blog.google/products/google-play/data-safety/, April 2022. Accessed: 2025-05-25.

[41] Funny-Discussions. What kind of data can be stolen using shortcut. https://www.reddit.com/r/shortcuts/comments/kk9ups/what_kind_of_data_can_be_stolen_using_shortcut/.

[42] Mélissa Godin. How domestic abusers have exploited technology during the pandemic. *TIME*, December 2020.

[43] hazzer74. Phone crasher (send to your enemies), 2020. Accessed: 2025-05-18.

[44] hmhrex. Routinehub - the state of malicious shortcuts. https://www.reddit.com/r/shortcuts/comments/as2cau/routinehub_the_state_of_malicious_shortcuts/, February 2019. Reddit post in r/shortcuts.

[45] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

[46] IFTTT. Ifttt - automate business & home. https://ifttt.com/explore.

[47] joaomgcd. Tasker for android. https://tasker.joaoapps.com/.

[48] Sean Michael Kerner. Ibm warns of apple siri shortcut scareware risk. https://www.eweek.com/security/ibm-warns-of-apple-siri-shortcut-scareware-risk/, 2019.

[49] Kelly King, Christine E Murray, Allison Crowe, Gwen Hunnicutt, Kristine Lundgren, and Loreen Olson. The costs of recovery: Intimate partner violence survivors' experiences of financial recovery from abuse. *The Family Journal*, 25(3):230–238, 2017.

[50] John Kuhn. Hey siri, get my coffee, hold the malware. https://securityintelligence.com/hey-siri-get-my-coffee-hold-the-malware/, 2019.

[51] Roxanne Leitão. Technology-facilitated intimate partner abuse: a qualitative analysis of data from online domestic abuse forums. *Human–Computer Interaction*, 36(3):203–242, 2021.

[52] Amanda Lenhart, Michele Ybarra, Kathryn Zickuhr, and Myeshia Price-Feeney. Online Harassment, Digital Abuse, and Cyberstalking in America. 2016.

[53] Clayton Lewis and Cathleen Wharton. Cognitive walkthroughs. In *Handbook of human-computer interaction*, pages 717–732. Elsevier, 1997.

[54] Isabel Lopez-Neira, Trupti Patel, Simon Parkin, George Danezis, and Leonie Tanczer. 'internet of things': How abuse is getting smarter. 2019.

[55] Neal Mangaokar, Ashish Hooda, Jihye Choi, Shreyas Chandrashekaran, Kassem Fawaz, Somesh Jha, and Atul Prakash. PRP: Propagating universal perturbations to attack large language model guard-rails. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10960–10976, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[56] Tara Matthews, Kathleen O'Leary, Anna Turner, Manya Sleeper, Jill Palzkill Woelfer, Martin Shelton, Cori Manthorne, Elizabeth F Churchill, and Sunny Consolvo. Stories from survivors: Privacy & security practices when coping with intimate partner abuse. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pages 2189–2201, 2017.

[57] Maverick. [2023] 10 powerful android automation apps (free & paid). https://www.airdroid.com/mdm/android-automation-app/, 2023.

[58] Mary L McHugh. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276–282, 2012.

[59] Marjan Nadim and Audun Fladmoe. Silencing women? gender and online harassment. *Social Science Computer Review*, 39(2):245–258, 2021.

[60] Borke Obada-Obieh, Yue Huang, Lucrezia Spagnolo, and Konstantin Beznosov. Sok: the dual nature of technology in sexual abuse. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2320–2343. IEEE, 2022.

[61] OpenAI. Gpt-4o. https://platform.openai.com/docs/models#gpt-4o.

[62] Eva PenzeyMoog and Danielle C Slakoff. As technology evolves, so does domestic violence: Modern-day tech abuse and possible solutions. In *The Emerald international handbook of technology-facilitated violence and abuse*, pages 643–662. Emerald Publishing Limited, 2021.

[63] Oriana Riva, Chuan Qin, Karin Strauss, and Dimitrios Lymberopoulos. Progressive authentication: deciding when to authenticate on mobile phones. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 301–316, 2012.

[64] Kevin A Roundy, Paula Barmaimon Mendelberg, Nicola Dell, Damon McCoy, Daniel Nissani, Thomas Ristenpart, and Acar Tamersoy. The many kinds of creepware used for interpersonal attacks. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 626–643. IEEE, 2020.

[65] Tania Roy, Eleanor Young, and Larry F Hodges. A second look at SecondLook: Design iterations and usability of digital dating abuse detection and awareness app. In *2020 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 1–11. IEEE, 2020.

[66] Nithya Sambasivan, Amna Batool, Nova Ahmed, Tara Matthews, Kurt Thomas, Laura Sanely Gaytán-Lugo, David Nemer, Elie Bursztein, Elizabeth Churchill, and Sunny Consolvo. " they don't leave us alone anywhere we go" gender and digital abuse in south asia. In *proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2019.

[67] Ltd. Samsung Electronics Co. [q os]bixby routine. https://galaxystore.samsung.com/prepost/000004770242.

[68] Sarita Schoenebeck, Cliff Lampe, and Penny Trieu. Online harassment: Assessing harms and remedies. *Social Media+ Society*, 9(1):20563051231157297, 2023.

[69] Aily Shimizu. Domestic violence in the digital age: Towards the creation of a comprehensive cyberstalking statute. *Berkeley J. Gender L. & Just.*, 28:116, 2013.

[70] Toby Shulruff. Gender-based violence enabled by location technologies. *International journal of geographical information science*, 36(12):2345–2351, 2022.

[71] Julia Slupska and Angelika Strohmayer. Networks of care: Tech abuse advocates' digital security practices. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 341–358, 2022.

[72] Julia Slupska and Leonie Maria Tanczer. Threat modeling intimate partner violence: Tech abuse as a cybersecurity challenge in the internet of things. In *The Emerald international handbook of technology-facilitated violence and abuse*, pages 663–688. Emerald Publishing Limited, 2021.

[73] Cynthia Southworth, Jerry Finn, Shawndell Dawson, Cynthia Fraser, and Sarah Tucker. Intimate partner violence, technology, and stalking. *Violence against women*, 13(8):842–856, 2007.

[74] Sophie Stephenson, Majed Almansoori, Pardis Emami-Naeini, and Rahul Chatterjee. " it's the equivalent of feeling like you're in Jail": Lessons from firsthand and secondhand accounts of Iot-Enabled intimate partner abuse. In *32nd Usenix Security Symposium (USENIX Security 23)*, pages 105–122, 2023.

[75] Sophie Stephenson, Majed Almansoori, Pardis Emami-Naeini, Danny Yuxing Huang, and Rahul Chatterjee. Abuse Vectors: A framework for conceptualizing IoT-enabled interpersonal abuse. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 69–86, 2023.

[76] Leonie Maria Tanczer, Isabel López-Neira, and Simon Parkin. 'i feel like we're really behind the game': perspectives of the united kingdom's intimate partner violence support sector on the rise of technology-facilitated abuse. *Journal of gender-based violence*, 5(3):431–450, 2021.

[77] Kurt Thomas, Devdatta Akhawe, Michael Bailey, Dan Boneh, Elie Bursztein, Sunny Consolvo, Nicola Dell, Zakir Durumeric, Patrick Gage Kelley, Deepak Kumar, et al. Sok: Hate, harassment, and the changing landscape of online abuse. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 247–267. IEEE, 2021.

[78] Daniel Thompson. Utilization of the ios shortcuts app to generate a surgical logbook tool: feasibility study. *JMIR Perioperative Medicine*, 4(1):e24644, 2021.

[79] Emily Tseng, Rosanna Bellini, Nora McDonald, Matan Danos, Rachel Greenstadt, Damon McCoy, Nicola Dell, and Thomas Ristenpart. The tools and tactics used in intimate partner surveillance: An analysis of online infidelity forums. In *29th USENIX security symposium (USENIX Security 20)*, pages 1893–1909, 2020.

[80] Warda Usman and Daniel Zappala. Sok: A framework and guide for human-centered threat modeling in security and privacy research. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 33–33. IEEE Computer Society, 2024.

[81] C. van Riper et al. Remote ecological monitoring with smartphones and connected devices. *Journal of Fish and Wildlife Management*, 12(1):163–170, Jan 2021.

[82] Jessica Vitak, Kalyani Chadha, Linda Steiner, and Zahra Ashktorab. Identifying women's experiences with and strategies for mitigating negative effects of online harassment. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, pages 1231–1245, 2017.

[83] Emily A Vogels. *The state of online harassment*, volume 13. Pew Research Center Washington, DC, 2021.

[84] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[85] Miranda Wei, Sunny Consolvo, Patrick Gage Kelley, Tadayoshi Kohno, Franziska Roesner, and Kurt Thomas. "There's so much responsibility on users right now:" Expert Advice for Staying Safer From Hate and Harassment. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–17, 2023.

[86] Pamela Wisniewski, Heng Xu, Mary Beth Rosson, Daniel F Perkins, and John M Carroll. Dear diary: Teens reflect on their weekly online risk experiences. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 3919–3930, 2016.

[87] Delanie Woodlock, Michael Salter, Molly Dragiewicz, and Bridget Harris. "living in the darkness": Technology-facilitated coercive control, disenfranchised grief, and institutional betrayal. *Violence against women*, 29(5):987–1004, 2023.

[88] World Wide Web Foundation and World Association of Girl Guides and Girl Scouts. The online experiences of girls and young women: A survey of 1,000 girls and young women across four countries, March 2020. Accessed: 2024-05-16.

[89] YAML. YAML: YAML Ain't Markup Language, 2025. Accessed: 2025-01-22.

[90] Glenn 'devalias' Grant (0xdevalias). Exporting shortcuts for sharing. https://gist.github.com/0xdevalias/27d9aea9529be7b6ce59055332a94477. Accessed: 2024-11-21.

A Simulating Spyware on Automation Apps

Shortcuts. We implement 31 out of 45 features of Flexispy Extreme through iOS Shortcuts, as explained in Sec. 4.1.2. The details of the replicability of a feature is specified in Figure 16.

Feature	Support	Feature	Support
Environment recording	yes	Browser bookmarks	yes
RemCam	yes	Network connections	yes
Spoof SMS	yes	Installed applications	yes
Email	yes	Photos	yes
Application screenshots	yes	Videos	yes
Notes	yes	Wallpaper images	yes
Address book	yes	Calendar	yes
SMS messages	yes	Send Remote Cmds (Web)	yes
MMS	yes	SMS Remote Commands	yes
Location tracking	yes	Check Battery Status	yes
Geo-Fencing	yes	Remotely restart device	yes
iMessage	part.	Application activity	part.
Audio files	part.	Visibility Option	part.
Remotely uninstall s/w	part.	Remotely deactivate s/w	part.
Remotely change features	part.	Remote Upgrade	part.
Run in Hidden Mode	part.	Phone Call recording	no
Phone Call interception	no	FaceTime Call Recording	no
Spycall	no	RemVid	no
Call Notification Alert	no	SMS Keyword deletion	no
FaceTime Call logs	no	SIM Changed Notification	no
Call logs	no	Browsing activity	no
Keylogger	no	Dashboard Alerts	no
Prevent uninstall of s/w	no		

Figure 16: Feature parity between FlexiSPY Extreme and what can be replicated with iOS Shortcuts. Third-party-app functions are omitted.

Feature	Support	Feature	Support
Call monitoring & recording	part.	Ambient listening	part.
Location tracking	yes	SMS monitoring	part.
Social-media/messaging logs	no	Keylogging	no
Remote camera access	part.	Media-file access	yes
Browser-history monitoring	no	App-activity monitoring	yes
Alerts & notifications	yes	Accessing contacts	yes
Stealth/detection	part.		

Figure 17: Tasker capabilities on Android. “partial” = root or major workaround required; “yes” = fully doable without root; “no” = not feasible even with root.

B LLM-Based Attack Analysis Breakdown by Domain

Domain	Category	Code Filter	LLM Labeled Yes
SG	Lockout	505	86
	Surveillance	54	2
	Impersonation	614	46
	Overloading	61	22
SS	Lockout	507	88
	Surveillance	49	3
	Impersonation	933	55
	Overloading	71	8
MC	Lockout	181	3
	Surveillance	10	0
	Impersonation	149	2
	Overloading	1	0
RH	Lockout	1,874	479
	Surveillance	258	16
	Impersonation	3,845	224
	Overloading	346	35

Figure 18: Potentially malicious recipes found in four Shortcuts recipe repositories: ShortcutGallery.com (**SG**); ShareShortcuts.com (**SS**), MatthewCassinelli.com/shortcuts (**MC**), and RoutineHub.co (**RH**)