Exploring the Security and Privacy Risks of Chatbots in Messaging Services

Jide Edu
King's College London
London, UK

Cliona Mulligan
King's College London
London, UK

Fabio Pierazzi
King's College London
London, UK

Jason Polakis
University of Illinois at Chicago
Chicago, US

Guillermo Suarez-Tangil
IMDEA Networks Institute
Madrid, Spain

Jose Such
King's College London
London, UK

ABSTRACT

The unprecedented adoption of messaging platforms for work and recreation has made it an attractive target for malicious actors. In this context, third-party apps (so-called chatbots) offer a variety of attractive functionalities that support the experience in large channels. Unfortunately, under the current permission and deployment models, chatbots in messaging systems could steal information from channels without the victim's awareness. In this paper, we propose a methodology that incorporates static and dynamic analysis for automatically assessing security and privacy issues in messaging platform chatbots. We also provide preliminary findings from the popular Discord platform that highlight the risks that chatbots pose to users. Unlike other popular platforms like Slack or MS Teams, Discord does not implement user-permission checks—a task entrusted to third-party developers. Among others, we find that 55% of chatbots from a leading Discord repository request the "administrator" permission, and only 4.35% of chatbots with permissions actually provide a privacy policy.

CCS CONCEPTS

• **Security and privacy** → **Social network security and privacy**; *Web application security*; **Usability in security and privacy**; **Privacy protections**.

KEYWORDS

Security and Privacy, Messaging platorms, Chatbots, Discord

ACM Reference Format:
Jide Edu, Cliona Mulligan, Fabio Pierazzi, Jason Polakis, Guillermo Suarez-Tangil, and Jose Such. 2022. Exploring the Security and Privacy Risks of Chatbots in Messaging Services. In *Proceedings of the 22nd ACM Internet Measurement Conference (IMC '22), October 25–27, 2022, Nice, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3517745.3561433

1 INTRODUCTION

Instant messaging platforms have become an essential collaborative tool, playing a pivotal role in redefining how people communicate and interact [36, 37]. By allowing real-time conversational communication in a virtual workplace, these platforms improve synergy and promote productivity. Beyond messaging, many of the platforms also support file sharing and videoconferencing. Services like Slack, Discord, MS Teams, and Telegram are so popular that their number of active users ranges from millions to billions [23, 50].

However, research has shown that messaging platforms also suffer from security and privacy issues [32, 39, 40, 54]. More recently, there have been reported security incidents in the media involving Discord being used to host, distribute and control malware [7, 9, 30], with >17,000 unique URLs in Discord's content delivery network pointing to malware [30]. Research also showed messaging platforms exposing users' data within its platform and connected accounts [33].

Furthermore, most messaging platforms, including those marketed for enterprise, support third-party chatbots (add-ons) [36] that add extra features to enrich their functionality. The third-party chatbots request permissions to enhance the users' conversations with content from external services, offer customizable features, or perform specific tasks on behalf of users. While the purpose of having permissions is to protect users, some permissions may be dangerous if granted to a malicious application. However, the permission model is often lax and/or poorly defined as there are no prevailing standards yet. This is analogous to the evolution observed in Android's permissions, which was necessary for addressing the multitude of flaws identified by research [19, 27, 34, 55].

As third-party chatbots in messaging platforms become increasingly popular for personal and business use, there is a need for assessing their security and privacy. In this paper, we propose a novel methodology for automatically assessing security and privacy issues of a chatbot service. Our analysis pipeline incorporates *static* (code and traceability analysis) and *dynamic* techniques (deploying deceptive honey-resources for detecting chatbots stealing users' information). We also apply the methodology to Discord, one of the most popular of such instant messaging platforms. As our findings in Discord reveal the inherent risks posed by chatbots and highlight the need for a more comprehensive analysis of this ecosystem, we hope that our work will motivate additional research in this space.

2 THREAT MODEL

Instant messaging platforms offer a fixed set of capabilities that users expand using *chatbots*, usually developed by third-parties, which leverage dedicated APIs provided by the platforms. Chatbots, unlike mobile apps (which may seem similar), do not run on the end user's device. Instead, chatbots are hosted directly by the developer either in a self-hosted server or in a cloud server (e.g., AWS, GCP, Azure). This creates a unique attack vector as developers can alter the chatbot code at any time after installation without the users being made aware of any change taking place (unless such changes entail asking for new permission) [13]. This creates a layer of obfuscation that prevents users from detecting what actions a chatbot takes outside of the explicit user-provided commands it receives. More importantly, malicious developers could take advantage of this to potentially sneak malicious code into their software via the application backend, as has been seen in similar domains [52].

Importantly, for users to take advantage of chatbots' capabilities, they need to permit access to their data. This includes data from the user's account, such as the user's email address, text or audio messages from the channels the chatbot is present in, or sensitive data inferred from user interactions with the chatbots. Considering that messaging platforms have become an integral workplace tool being used for extremely sensitive communication, the need for an in-depth analysis of the chatbot ecosystem becomes apparent. For instance, over-privileged chatbots that collect sensitive information or are endowed with excessive capabilities pose significant security and privacy risks [6]. As such, it is important to note that even though chatbots are subjected to a vetting process, existing research from other domains [14] showed how malicious developers may bypass this process. Messaging platforms that support third-party applications are known to have a very similar architecture (e.g., Discord, Slack, MS Teams, and Telegram). For instance, their third-party applications reside in the cloud, use a two-level access control system consisting of the OAuth protocol and a runtime policy enforcer [13], and their source codes are not publicly available for analysis, hence they are exposed to similar risks. In this paper, we aim to understand the risks posed by chatbots in messaging platforms and propose a method and pipeline for uncovering invasive or malicious behaviors.

3 METHODOLOGY

We design a methodology for automatically assessing security and privacy issues in messaging platform chatbots. Figure 1 shows an overview of our analysis pipeline. Recall that chatbots' software is not readily available in central repositories. Instead, services run on developer-controlled servers that get feeds from conversations once they are triggered in end devices. Users enable (install) the software through a repository listing the chatbot (typically maintained by the messaging service or its users). In some cases, the source-code of the chatbot is shared together with the description of the chatbot in its listing. To understand the chatbot ecosystem, we first build a system that crawls existing listings of chatbots. We then design a method for analyzing chatbots dynamically, and — when available — statically (traceability and code analysis). We demonstrate the applicability of our methods to Discord in Section 4.2.

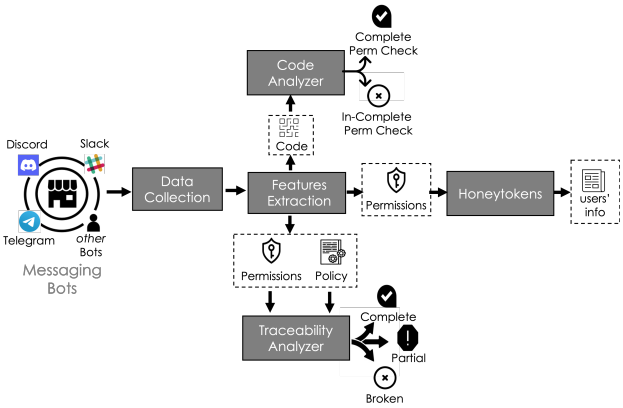

Figure 1: Overview of the Methodology.

Data Collection. Our data collection process traverses listings of chatbots and extracts attributes such as the permissions they request, sample commands, their privacy policy, and the link to their source code repository (if present). For this, we develop a Web scraper using the Selenium Python framework [48] that automatically crawls the messaging platforms' websites with third-party chatbots and extracts all their attributes. Platform providers implement different anti-scraping strategies, making it challenging to scrape the repositories. These include captchas and email verification. Also, some of the repositories have varying page structures. Thus: i) We limit the rate at which we generate our requests; ii) We use "2Captcha", a Captcha solving service, to overcome the captchas restriction; iii) We mimic human behavior; iv) We design our scraper to handle and react to exceptions such as "NoSuchElementException", when elements unexpectedly become unavailable, or "TimeoutException", when a command takes more than the wait time.

Traceability Analysis. We perform traceability analysis to understand how developers disclose and justify the data permissions they request. The Traceability Analyzer collects all data permissions found in the privacy policy and compares them with the original set of data permissions requested by the chatbot. This is done to determine if there is a reported reason for the data request. As done in other domains [2, 4, 5, 24, 25, 38, 47], we classify disclosure practices as *complete*, *partial*, or *broken*, depending on how well they are disclosed. Due to the sheer amount of chatbots, we automate the traceability analysis through a keyword-based approach [8]. First, we identify words that are often used in privacy policies to identify data practices in other domains: *Collect, Use, Retain, and Disclose. Collect* relates to set of keywords used to describe when a program collect, gather, or acquire data from the user; *Use* indicates a program uses or processes the data; *Retain* means storing or remembering user data, and *Disclose* means a that the program shares or transfers data to another party. We then identified the synonyms of these words and keywords akin to the chatbot ecosystem obtained from existing chatbot permissions and privacy policies. When a privacy policy explains how data is collected, used, retain and disclosed we say that the policy is *complete*. When any of the keyword-set is described, we say that the policy is *partial*, and *broken* when none.

Dynamic Analysis (Honeypot). In the absence of a direct access to the software of a chatbot, we develop a dynamic analysis approach to study remote programs in their environment. For this, we use a honeypot [51] instrumented with canary tokens (also known as honeytokens) [46]. Examples of canary tokens include files, URLs, and email addresses. Our method is based on the assumption that a chatbot should not be interacting with a token posted in a channel unless it is part of its functionality. We design a framework for controlled experiments in messaging services to test available chatbots. Our framework implants unique honeytokens only available to chatbots under study. Specifically, we test each chatbot in an independent and isolated messaging environment. This let us determine which chatbots gain unauthorized access to the honeytoken, i.e., when it opens a file, accesses a link, or uses email addresses shared in the chat.

For the honeypot environment to appear active and in use, we provide a feed of frequent exchange of messages from multiple (automated) users. To make this feed realistic, our system takes as input an exchange of real messages. While the Enron emails dataset is widely used in research for similar purposes [15], our implementation leverages publicly available messages from social networks (OSN) like Reddit. Our rationale is that the style of the communication used in an instant messaging environment is shorter and less formal than email, thus rendering OSN messages more suitable and contextually relevant to our study. Our implementation uses four canary tokens: email, URL, word, and PDF. Canary tokens consist of unique identifiers embedded in URLs or placed in a document meta-data. Requesting the URL or opening the document allows us to receive a signal tied to the token.

Code Analysis. To analyze the risks of third-party chatbots, we design a mechanism to understand how the platform's permission system is used by the chatbots, via program analysis of publicly-available chatbot code. This mechanism is comprised of three steps. First, we collect the source-code of the chatbots. For this, we parse the description in the app listing, retrieve pointers to open source repositories, and download their source-code. As mentioned in Section 2, note that not all chatbots make their code available. Second, we model permission checks through a manual review of a subset of case studies. For this, we identify the APIs that enable developers to verify if a user has the right set of permissions to access a resource. Third, we build an automated approach that looks for these APIs and determines if a bot conducts the expected permission checks declared in the documentation (manifest).

4 USE CASE: DISCORD

In this section, we apply the proposed methodology to explore the security and privacy risks in the Discord ecosystem.

4.1 Discord Ecosystem

Discord Guild: Users in Discord can create or be invited to join servers called guilds. These guilds are comprised of voice and text channels, where users can talk or exchange messages. Discord manages access within these guilds using a role-based system. All users within a guild have a primary role called "everyone", which provides some simple permissions to access channels and send messages. The guild owner and those with the "manage roles" permission [20] can then give users further roles with additional permissions. Thus, roles in Discord can be assigned on both a guild-based level and a channel-based level. In addition, Discord has both private and public guilds. Private guilds require an invitation to join, while public guilds are open for anyone to participate.

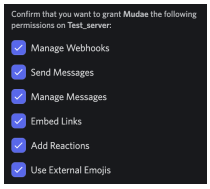

Figure 2: Example Discord chatbot installation page.

Discord Users: Users in Discord can spawn across the whole platform, in text and voice chat, be part of guilds, and participate in them. Users are classified as "bot" (chatbot) or "normal" users. While chatbots and normal users share some similarities, chatbots are automated users that are "owned" by another normal user. Unlike normal users that have restrictions on the number of guilds they can be part of, chatbots do not have any limits [21]. Chatbots can be installed through an OAuth link, and OAuth is used for access delegation and authorization flows for applications, phones, and smart devices [41]. Bots in Discord are used for various purposes, including moderation, role management, scheduling, and other extra features. Currently, there is no official marketplace for Discord chatbots, and they are primarily found at "www.top.gg".

Discord Permissions: Users must have the "manage guild" permission to install a chatbot into a guild. Upon installation the Discord chatbot can request permissions to access the messaging platform resources (see [20] for the complete permission list in Discord). When installing a chatbot, a screen is shown to the users that details what information and system resources the application intends to access (e.g., Figure 2). A user must explicitly grant this access to continue with the installation. Most chatbots need to access multiple permissions, and permissions can also be changed at any time after the chatbot is installed.

Discord implements a "permission hierarchy" system, and the rules are as follows: i) A chatbot can grant roles to other users of a lower position than its own highest role. ii) A chatbot can edit roles of a lower position than its highest role, but it can only grant permissions it has to those roles. iii) A chatbot can only sort roles lower than its highest role. iv) A chatbot can only kick, ban, and edit nicknames for users whose highest role is lower than the chatbot's highest role. v) Otherwise, permissions do not obey the role hierarchy. It should be noted that the "administrator" permission lets the user access all permissions for all channels in the guild.

Some Discord chatbots may also request additional scopes on top of the chatbot scope. This can give them extra user data as well as other privileges. It should be noted that some of these scopes can only be requested if whitelisted by Discord staff. Likewise, some of these scopes are only for testing purposes. The bot scope is by nature required for all chatbots. Once installed in a guild, users can usually interact with a chatbot by sending a message or command into a channel. These messages typically contain a prefix, e.g., "!info". As long as the user has the "send message" permission within the channel, they should be able to interact with the chatbot.

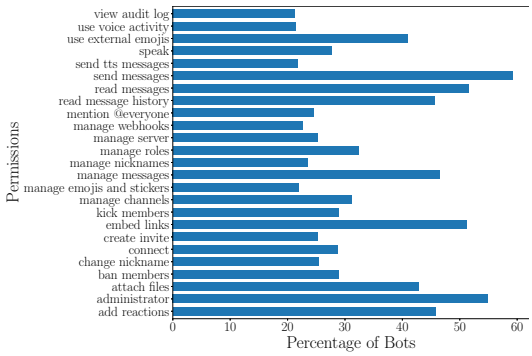

Figure 3: Percentage distribution of top 20 permissions requested by Discord chatbots.

Table 1: Bots distribution by number of developers.

No of Bots	Developers (No. & %)		No of Bots	Developers (No. & %)	
1	11,070	89.08%	6	6	0.05%
2	1,089	8.76%	7	4	0.03%
3	185	1.49%	8	2	0.02%
4	50	0.40%	11	1	0.01%
5	19	0.15%	12	1	0.01%

4.2 Discord Measurement

We scrape *top.gg*, the leading repository of Discord chatbots, navigate the "top chatbot" list, and extract metadata from the chatbots, which includes the chatbot's ID, name, URL, tags, permissions, guild count, description and GitHub link.

Permissions Measurement. We collected a total of 20,915 Discord chatbots. 74% (15,525) of the chatbots requested valid permissions on the installation page; the remaining 26% (5,390) have invalid permissions due to invalid invite links, have been removed, or timed out due to slow redirect links. Figure 3 shows the percentage distribution of the top 20 permissions requested by the chatbots. The most commonly requested permission is the "SEND MESSAGE" permission, which is requested by 9,188 (59.18%) chatbots. This is followed by the "administrator" permission requested by 8,521 (54.86%). Interestingly, the "administrator" permission allows all permissions, bypasses channel permission overwrites, and gives bots access to sensitive user data among others.

Discord Chatbots and Developers: Table 1 shows the chatbots distribution by the number of developers. From the 12,427 developers in our dataset, 89% (11,070) have published just one chatbot. The developer with the highest number of chatbots (namely, developer *editid#6714*) has 12 unique chatbots. We also see developers using third-party development platforms such as "botghost.com", "autocode.com", "discordbotstudio.org" which lower development barriers. These platforms offer free sample chatbots that developers can customize regardless of their technical background. This enables regular internet users to deploy bots without adequately understanding the ecosystem, resulting in security vulnerabilities and privacy violations [1].

Discord Chatbots Data Traceability. Discord's privacy policy states that chatbots have "access to their end users' information, including message content, message metadata, and voice metadata" and that they "must use such information only to provide the SDK/API functionality within their applications and/or services" [22]. Notwithstanding, it is unclear how the user data is being used and whether there is sufficient disclosure about it. There has been increasing ambiguity in the past about the access that third-party applications have to user's data, and the Cambridge Analytica scandal [16] is a prime example of the risks. We look at chatbot privacy policies to understand how developers disclose and justify the data permissions they request.

Table 2: Discord Traceability Results.

Features	Count	Percent
Unique active chatbots	15,525	100%
Website Link	5,786	37.27%
Privacy Policy Link	676	4.35%
Privacy Policy	673	4.33%

Discord chatbots tend to not have any visible privacy policies on "top.gg". This necessitates visiting the chatbot's website (if present) for finding its privacy policy document. We automate this process using the Selenium Python framework and leveraging element locators [49], which let us identify the HTML DOM element to act on. If the website link is not available and a privacy policy is not found, we assume broken traceability. This implies that Discord users do not know about developers' data practices and cannot identify when certain data practices may harm them. Table 2 shows that there are 5,786 (37.27%) chatbots with a website link, but only 676 (4.35%) of the them have a privacy policy. This indicates that the remaining 14,852 (95.67%) chatbots have broken traceability, as they do not have a privacy policy document to disclose how data is accessed, used, shared, or stored. Furthermore, out of the 676 privacy policy links, only 673 (4.33%) lead to a valid page. Upon further analysis of these policies, using keyword-based traceability, we do not find any chatbot with complete traceability. Instead we find partial traceability as policies do not completely disclose their data practices. Due to the limitations of the keyword-based traceability approach, as later discussed in Section 5, we perform a validation of the traceability results through a random selection of 100 privacy policies and a manual review process. The result shows that none of these privacy policies was misclassified. Furthermore, we observed that many of these policies are generic and they are not tailored to this ecosystem. These results match those observed in earlier studies [2, 3, 24] where developers were found to be reusing existing privacy policies verbatim across different domains, and permissions without modification.

Discord Chatbots Honeypots. We tested a diverse sample of most-voted chatbots from "www.top.gg" as these chatbots are more likely to be active and maintained. We considered doing a sample from the middle and least voted but they were mainly offline or not being used (i.e., in 0 guilds). The bots tested ranged in guild count (3M to 25), vote count (876K to 6) and chatbot purpose (such as gaming, fun, social, music, meme), thus reflecting the diversity of the general population of chatbots. Besides, many of these chatbots were present in over 250,000 guilds, and if they were malicious, they would put many users at risk. In our experiments we create new private guilds, add a chatbot to the guild using the chatbot invite link and post messages using automation. We name each guild after the corresponding chatbots for easy identification. To add a chatbot to the guild, we need to solve a Google reCAPTCHA. Due to its affordability and quick solving time, we used the captcha-solving service "2Captcha" to automate the process. Next, we create the

canary tokens and post them as messages to the guild. We use the guild name as our identifier to detect triggered tokens. We note that to post a seemingly real conversation we create fake personas by registering virtual users into Discord. In practice, we found that when a new account quickly joins many guilds, it is flagged by Discord, and mobile verification is required. As such, we completed this step manually. Subsequently, our system ensures that the virtual accounts post alternating messages so that interactions resemble legitimate conversations between actual users.

We created 5 virtual users each in 100 guilds and install a total of 500 chatbots on individual guilds. Each guild was populated with a canary URL, email address, pdf and word document tokens. In addition, we posted 25 conversational messages to let the guild appear active. At the time of writing, our system has detected one chatbot triggering the canary token. The word document and URL were accessed for a chatbot named "Melonian". Melonian does not offer a functionality that would require opening word documents or visiting URLs. After the triggers, a user posted a message as the guild's chatbot that reads "`[w]tf is this bro`", which is clearly not an automated message generated by a bot. It appears the chatbot owner/developer logged in as the chatbot, potentially through a third-party service, made a cursory inspection of the contents of the messages in the guild, and accessed files posted *without authorization*. This contradicts Discord's privacy policy, which states that developers "must use such information only to provide the SDK/API functionality within their applications and/or services" [22]. Furthermore, the owner/developer of the chatbot could have been infringing computer fraud, misuse or abuse acts that regulate the unauthorized access to computer material of information. They could have also violated data protection legislation like GDPR or CCPA should the word document contain personal or confidential information. When looking at what this finding means to other users of Discord, our result needs to be interpreted with caution as this particular chatbot is only present in a few guilds Nevertheless, this result suggests an inherent risk present in the Discord ecosystem. It also confirms that users would not be able to detect a breach of their privacy without the trigger of the canary tokens and subsequent messages.

Discord Chatbots Code Analysis. In Discord, permissions checks are not enforced by the platform. Instead, the developer of a chatbot is responsible for checking if the user invoking the chatbot has the permission to perform any of the actions supported by the chatbot. As chatbots may have more privileges than a user, failing to check this permission could lead to re-delegation attacks [18, 29]. We built a Web scraper that visits the GitHub links extracted from the top.gg website to check for the presence of the GitHub code section. If this is found, we then analyze the repository. The scraper will then check for languages used for the code and extracts the first (main) language provided for the repository. This will help to pinpoint what APIs to check afterwards. We traversed over 800 pages from the "top chatbot" list and recorded information of 15,525 chatbots. Out of these chatbots, 23.86% (3,705) had GitHub links on their description page. Furthermore, 60.46% (2,240) of these links lead us to valid repositories. The rest links take us to user profiles, a GitHub with no repositories, a GitHub with no public repositories, or an invalid link.

Table 3: Discord role checks in JavaScript & Python.

No.	Checks	No.	Checks
1	.hasPermission (	3	member.roles.cache
2	.has(	4	userPermissions

From our analysis we find that JavaScript (41%) is the most popular language, closely followed by Python (32%). This is unsurprising as both languages have well-documented Discord chatbot libraries, "discord.js" and "discord.py". However, there are also some repositories that we could not identify their language. Manual inspection of these repositories shows even though these are valid repositories, they do not contain any source code. Many only have READ.ME files with chatbot descriptions or commands, or just information on licensing and changelogs. Considering these chatbots, only 14.39% (2,234) of the 15,525 chatbots have publicly available source code. Since Javascript and Python are the most popular languages (73% together), we check which Discord chatbots developed with these languages. We identify four ways for permission or role checking within JavaScript and Python (see Table 3) using the method described in Section 3.

We check the selected source codes for the presence of these APIs to identify chatbots that are performing permission checking. In total, we analyzed 925 available JavaScript repositories and 718 Python repositories. Out of these repositories, only 675 (72.97%) of JavaScript repositories and 19 (2.65%) of Python repositories contained one of the APIs that are used to perform permission checking. The rest, 27.02% (250), and 97.35% (699) of JavaScript and Python repositories, respectively, do not perform any form of permission checking. This creates the possibility for permission abuse as users can take advantage of the privilege assigned to a chatbot for performing unauthorized actions.

5 DISCUSSION AND LIMITATIONS

Here we discuss the key findings and limitations of our study.

Improper Permission Checks. As aforementioned, a bot can not perform actions if it does not have the corresponding permission. However, there is the possibility of potential permission abuse in a situation where a user can take advantage of the privilege assigned to a bot to perform actions the user is not permitted to do. For instance, in Discord, the current permission framework allows the developer to implement and perform the necessary permission check. However, this results in it becoming the developer's responsibility to ensure that the bot retrieves the message's author and checks if the user has the required permission before acting for them. Thus, improper permission checks could leads to permission re-delegation attacks allowing users to bypass privileges.

Incomplete Traceability. By exploring how permissions are handled and requested, we find that 95.67% of Discord chatbots that request permissions lack a privacy policy. Importantly, even when privacy policies are present, they do not adequately disclose their data practices. This finding highlights the need for a more comprehensive analysis of this ecosystem and chatbots' data practices.

Misunderstanding the permission system. As shown in Figure 3, in addition to other permissions, the majority of bots request the admin permission, which encompasses all other permissions. However, asking for anything in addition to admin is redundant and may imply that the developer does not completely understand

the permission system. Hence, there is a need for developers to better understand the permission system so they can build secure, privacy-aware bots with the minimal required permissions.

Limitations. A number of important limitations need to be considered. First, there are thousands of chatbots. Performing dynamic analysis for all these bots will be challenging and time-consuming. Through the case of Melonian, we saw that malicious chatbots in Discord currently involve a manual assessment of the target. This is a feature we have seen other fraudulent activities do [10, 43] before attacks get commoditized [11, 42, 53]. While we selected a diverse sample of bots, we could have overlooked some potentially bots with issues. Thus, our findings suggest further research in the area is needed. This includes novel ways to address the challenge of assessing the compliance of software hosted in the cloud or for which there is no access to the software itself.

Second, our traceability analysis relies on keyword-based approaches as there is currently no annotated dataset that can be used to train a ML model for the different chatbot platforms. However, words often have multiple meanings and could also be written in various forms, which could affect the accuracy of the traceability result. Nevertheless, we note that this does not affect the cases with broken traceability results (due to the absence of a privacy policy all together), which represents the vast majority of cases. Exploring ML techniques for the analysis would be an interesting research direction, as it has been done for voice assistants [24, 25]. Also, we could not use any of the existing NLP-based tools [2, 3, 31], because their ontologies do not cover all the data types in this new ecosystem. Nonetheless, we expect that including privacy policies will become the norm in the future, as messaging platforms have a more active interest (or legal requirement) to secure their ecosystems, similar to what we have seen recently in voice assistants [25].

Third, our code analysis is limited only to a few bots as i) not all bots' source codes are available to the public for analysis. ii) we only considered the bots developed using the JavaScript and Python libraries. For example, while these sets of bots represent more than 70% of the bots in the top Discord repository, follow-up work on other languages is part of our future research. Besides, malicious bots are less likely to post source code voluntarily.

Fourth, adding a new account to many guilds for the dynamic analysis often requires mobile verification. We currently complete this step manually, which takes time. A possible area of future work would be to develop an automated way of creating virtual users eliminating the manual mobile verification step. Likewise, we used our proposed method to explore the risks in the Discord platform. Scaling our analysis and applying our methodology to other popular platforms like Slack, MS Teams, and Telegram is also part of our future work.

Although this is an exploratory work that aims to identify flaws, misconfigurations, and problematic practices, part of our future studies is to perform a large-scale measurement that quantifies the *prevalence* of such phenomena.

Ethical considerations. Our research fully abides by the ethical principles guidelines outlined in the Menlo and Belmont Report. In particular, our system does not intentionally interact with humans nor collects data containing personal identifiable information. Moreover, our data collection process of crawling websites and our system interacting with chatbots, was done at a rate that does not create any disruption to other service users.

6 RELATED WORK

Research on Instant Messaging Platform Chatbots. Authors in [12] studied how instant messaging chatbots extend the collaborative benefits of instant messaging into new areas, while acknowledging potential security and privacy risks. The research in [33] found that messaging platforms such as Discord, Telegram and WhatsApp expose users' sensitive data within the platforms and other connected third-party accounts such as Twitch, Spotify and Twitter. For example, Discord was found to expose at least one social media account for 30% of users. The work in [35] performed a statistical analysis on chatbot usage for moderation by randomly joining some Discord guilds. The authors found that larger communities use chatbots for moderation. Chatbots having access to large communities only furthers privacy concerns. Anyway, the study is limited by the small sample size used for the analysis and the manual methodology. In a parallel effort, authors in [13] show how malicious chatbots can eavesdrop on the user by reading their messages without permission; launch fake video calls; and automatically merge code into repositories without user approval. However, unlike our work that uses honeytoken to measure misuse in the wild, the authors exploited the messaging platform's access control model to identify potential malicious practices. In addition, [13] focuses its analysis on Slack and MS Teams, which have a runtime mechanism to enforce security policies. Our work shows that Discord does not implement a runtime enforcer delegating trust on third party developers, which widens the attack surface.

Detecting Data Misuse by third-parties. There is considerable research into security analysis techniques and privacy violation detection, one of which is deception technology. This method aims to create a fake entity, which can be a file, page or account details, in a system to entice an attacker [17, 26, 28, 44, 45]. This is commonly referred to as honeypot (or honeytoken) [51]. The use of honeytoken is described as a cost-effective, simple to deploy, and highly effective solution in detecting internal data leak threats [46]. More recently, a study [28] used honeypots to detect data misuse by third-party applications on social networks. The authors supplied third-party applications in Facebook with canary email addresses and analyzed the emails these accounts received. While our dynamic analysis experiment is conceptually similar, we focus on a different domain and in addition to canary emails, we also use URL, word, and PDF honeytokens within the Discord workspace.

7 CONCLUSIONS

Over-privileged chatbots that collect sensitive information or are endowed with excessive capabilities pose significant security and privacy risks. In this paper, we presented a methodology for automatically assessing the security and privacy issues of chatbots in instant messaging services. Our methodology is then used to explore the risks in the Discord service, and our findings reveal the inherent risks chatbots pose to users' security and privacy (55% of bots asking for administrator permissions, lack of traceability, improper use of those permissions) highlighting the need for a more thorough analysis of this ecosystem. As this technology continue to

grow, so will the number of developers and chatbots, which could usher in a new level of threats and threat actors. Adopting stricter scrutiny when developers collect data and a continuous rigorous vetting process by the platform's provider could help mitigate risks. We hope our work will motivate additional research in this space, and the methodology will serve as the basis for future work.

Acknowledgments

This research was funded by EPSRC under grant EP/T026723/1, the "Ramon y Cajal" Fellowship RYC-2020-029401-I, and the National Science Foundation (CNS-1934597, CNS-2211574, CNS-2143363). Any opinions, findings, conclusions, or recommendations expressed herein are those of the authors.

REFERENCES

[1] Yasemin Acar, Michael Backes, Sven Bugiel, Sascha Fahl, Patrick McDaniel, and Matthew Smith. 2016. Sok: Lessons learned from android security research for appified software platforms. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 433–451.

[2] Benjamin Andow, Samin Yaseer Mahmud, Wenyu Wang, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Tao Xie. 2019. Policylint: investigating internal privacy policy contradictions on Google play. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 585–602.

[3] Benjamin Andow, Samin Yaseer Mahmud, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Serge Egelman. 2020. Actions speak louder than words: Entity-sensitive privacy policy and data flow analysis with policheck. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*. 985–1002.

[4] Pauline Anthonysamy, Matthew Edwards, Chris Weichel, and Awais Rashid. 2016. Inferring semantic mapping between policies and code: the clue is in the language. In *International Symposium on Engineering Secure Software and Systems*. Springer, 233–250.

[5] Pauline Anthonysamy, Phil Greenwood, and Awais Rashid. 2013. Social networking privacy: Understanding the disconnect from policy to controls. *Computer* 46, 6 (2013), 60–67.

[6] Alexandre Bartel, Jacques Klein, Yves Le Traon, and Martin Monperrus. 2012. Automatically securing permission-based software by reducing the attack surface: an application to Android. In *2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. 274–277. https://doi.org/10.1145/2351676.2351722

[7] Becky Bracken. 2021. Attackers Blowing Up Discord, Slack with Malware. https://threatpost.com/attackers-discord-slack-malware/165295/. [Online; last accessed 18-August-2021].

[8] Travis D. Breaux, Hanan Hibshi, and Ashwini Rao. 2014. Eddy, a Formal Language for Specifying and Analyzing Data Flow Specifications for Conflicting Privacy Requirements. *Requir. Eng.* 19, 3 (Sept. 2014), 281–307. https://doi.org/10.1007/s00766-013-0190-7

[9] Edmund Brumaghin. 2021. Sowing Discord: Reaping the benefits of collaboration app abuse. https://blog.talosintelligence.com/2021/04/collab-app-abuse.html. [Online; last accessed 18-August-2021].

[10] Elie Bursztein, Borbala Benko, Daniel Margolis, Tadek Pietraszek, Andy Archer, Allan Aquino, Andreas Pitsillidis, and Stefan Savage. 2014. Handcrafted fraud and extortion: Manual account hijacking in the wild. In *Proceedings of the 2014 conference on internet measurement conference*. 347–358.

[11] Juan Caballero, Chris Grier, Christian Kreibich, and Vern Paxson. 2011. Measuring {Pay-per-Install}: The Commoditization of Malware Distribution. In *20th USENIX Security Symposium (USENIX Security 11)*.

[12] Stephen Chan, Benjamin Hill, and Sarita Yardi. 2005. Instant messaging bots. *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work - GROUP '05* (2005). https://doi.org/10.1145/1099203.1099221

[13] Yunang Chen, Yue Gao, Nick Ceccio, Rahul Chatterjee, Kassem Fawaz, and Earlence Fernandes. 2022. Experimental Security Analysis of the App Model in Business Collaboration Platforms. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 2011–2028. https://www.usenix.org/conference/usenixsecurity22/presentation/chen-yunang-experimental

[14] Long Cheng, Christin Wilson, Song Liao, Jeffrey Young, Daniel Dong, and Hongxin Hu. 2020. Dangerous Skills Got Certified: Measuring the Trustworthiness of Skill Certification in Voice Personal Assistant Platforms. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*. In press.

[15] William W. Cohen. 2022. Enron Email Dataset. https://www.cs.cmu.edu/~enron/. [Online; last accessed 15-May-2022].

[16] Nicholas Confessore. 2021. Cambridge Analytica and Facebook: The Scandal and the Fallout So Far. https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html. [Online; last accessed 18-August-2021].

[17] Emiliano De Cristofaro, Arik Friedman, Guillaume Jourjon, Mohamed Ali Kaafar, and M. Zubair Shafiq. 2014. Paying for Likes? Understanding Facebook Like Fraud Using Honeypots. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (Vancouver, BC, Canada) *(IMC '14)*. Association for Computing Machinery, New York, NY, USA, 129–136. https://doi.org/10.1145/2663716.2663729

[18] Biniam Fisseha Demissie, Mariano Ceccato, and Lwin Khin Shar. 2020. Security analysis of permission re-delegation vulnerabilities in Android apps. *Empirical Software Engineering* 25, 6 (2020), 5084–5136. https://doi.org/10.1007/s10664-020-09879-8

[19] Michalis Diamantaris, Elias P. Papadopoulos, Evangelos P. Markatos, Sotiris Ioannidis, and Jason Polakis. 2019. *REAPER: Real-Time App Analysis for Augmenting the Android Permission System*. Association for Computing Machinery, New York, NY, USA, 37–48. https://doi.org/10.1145/3292006.3300027

[20] Discord. 2020. Discord Developer Portal — API Docs For Bots And Developers. https://discord.com/developers/docs/topics/permissions.

[21] Discord. 2020. Users Resource. https://discord.com/developers/docs/resources/user [Online; last accessed 8-October-2021].

[22] Discord. 2021. Privacy policy. https://discord.com/privacy. [Online; last accessed 18-August-2021].

[23] Pavel Durov. 2020. 400 Million Users, 20,000 Stickers, Quizzes 2.0 and €400K for Creators of Educational Tests. https://telegram.org/blog/400-million. [Online; last accessed 04-January-2021].

[24] Jide Edu, Xavi Ferrer Aran, Jose Such, and Guillermo Suarez-Tangil. 2021. SkillVet: Automated Traceability Analysis of Amazon Alexa Skills. *IEEE Transactions on Dependable and Secure Computing* (2021), 14. https://doi.org/10.1109/TDSC.2021.3129116

[25] Jide Edu, Xavier Ferrer-Aran, Jose Such, and Guillermo Suarez-Tangil. 2022. Measuring Alexa Skill Privacy Practices across Three Years. In *Proceedings of the ACM Web Conference 2022 (WWW '22)*. Association for Computing Machinery, New York, NY, USA, 670–680.

[26] El Bouzekri El Idrissi Younes, El Mendili Fatna, and Maqrane Nisrine. 2016. A security approach for social networks based on honeypots. In *2016 4th IEEE International Colloquium on Information Science and Technology (CiSt)*. 638–643. https://doi.org/10.1109/CIST.2016.7804964

[27] Zheran Fang, Weili Han, and Yingjiu Li. 2014. Permission based Android security: Issues and countermeasures. *Computers & Security* 43 (2014), 205–218. https://doi.org/10.1016/j.cose.2014.02.007

[28] Shehroze Farooqi, Maaz Musa, Zubair Shafiq, and Fareed Zaffar. 2020. CanaryTrap: Detecting Data Misuse by Third-Party Apps on Online Social Networks. *Proceedings on Privacy Enhancing Technologies* 2020, 4 (2020), 336–354. https://doi.org/10.2478/popets-2020-0076

[29] Adrienne Porter Felt, Helen J. Wang, Alexander Moshchuk, Steve Hanna, and Erika Chin. 2011. Permission Re-Delegation: Attacks and Defenses. In *20th USENIX Security Symposium (USENIX Security 11)*. USENIX Association, San Francisco, CA. https://www.usenix.org/conference/usenixsecurity11/permission-re-delegation-attacks-and-defenses

[30] Sean Gallagher and Andrew Brandt. 2021. Malware increasingly targets Discord for abuse. https://news.sophos.com/en-us/2021/07/22/malware-increasingly-targets-discord-for-abuse/. [Online; last accessed 18-August-2021].

[31] Hamza Harkous, Kassem Fawaz, Rémi Lebret, Florian Schaub, Kang G. Shin, and Karl Aberer. 2018. Polisis: Automated Analysis and Presentation of Privacy Policies Using Deep Learning. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Baltimore, MD, 531–548. https://www.usenix.org/conference/usenixsecurity18/presentation/harkous

[32] Stephen Hilt. 2017. Chat App Discord Abused to Attack ROBLOX Players. https://www.trendmicro.com/en_us/research/17/h/chat-app-discord-abused-cybercriminals-attack-roblox-players.html. [Online; last accessed 18-January-2021].

[33] Mohamad Hoseini, Philipe Melo, Manoel Junior, Fabricio Benevenuto, Balakrishnan Chandrasekaran, Anja Feldmann, and Savvas Zannettou. 2020. Demystifying the Messaging Platforms' Ecosystem Through the Lens of Twitter. *Proceedings of the ACM Internet Measurement Conference* (2020). https://doi.org/10.1145/3419394.3423651

[34] Asma Khatoon and Peter Corcoran. 2017. Android permission system and user privacy — A review of concept and approaches. In *2017 IEEE 7th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*. 153–158. https://doi.org/10.1109/ICCE-Berlin.2017.8210616

[35] Charles Kiene and Benjamin Mako Hill. 2020. Who Uses Bots? A Statistical Analysis of Bot Usage in Moderation Teams. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems (CHI EA '20)*. Association for Computing Machinery, New York, NY, USA, 1–8.

[36] Ruth Lechler, Emanuel Stöckli, Roman Rietsche, and Falk Uebernickel. 2019. Looking Beneath the Tip of the Iceberg: The Two-Sided Nature of Chatbots and Their Roles for Digital Feedback Exchange. In *Proceedings of the 27th European Conference on Information Systems (ECIS)*. https://www.alexandria.unisg.ch/257166/

[37] Jennie Lin. 2020. Picking Up the Slack: Collaboration Tools to Build Community and Increase Productivity in Nephrology. *Seminars in Nephrology* 40, 3 (2020), 298 – 302. https://doi.org/10.1016/j.semnephrol.2020.04.009 Nephrology and Social Media.

[38] Gaurav Misra, Jose Such, and Lauren Gill. 2017. A Privacy Assessment of Social Media Aggregators. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. ACM, 561–568.

[39] Elizabeth Montalbano. 2020. TrickBot Attack Exploits COVID-19 Fears with DocuSign-Themed Ploy. https://threatpost.com/trickbot-attack-covid-19docusign-themed-malw/155391/. [Online; last accessed 18-January-2021].

[40] Ruchna Nigam and Kyle Wilhoit. 2017. TeleRAT: Another Android Trojan Leveraging Telegram's Bot API to Target Iranian Users. https://unit42.paloaltonetworks.com/unit42-telerat-another-android-trojan-leveraging-telegrams-bot-api-to-target-iranian-users/. [Online; last accessed 18-January-2021].

[41] Oauth.net. 2020. OAuth 2.0. https://oauth.net/2/.

[42] Jeremiah Onaolapo, Nektarios Leontiadis, Despoina Magka, and Gianluca Stringhini. 2021. {SocialHEISTing}: Understanding Stolen Facebook Accounts. In *30th USENIX Security Symposium (USENIX Security 21)*. 4115–4132.

[43] Jeremiah Onaolapo, Enrico Mariconti, and Gianluca Stringhini. 2016. What happens after you are pwnd: Understanding the use of leaked webmail credentials in the wild. In *Proceedings of the 2016 Internet Measurement Conference*. 65–79.

[44] Abigail Paradise, Asaf Shabtai, Rami Puzis, Aviad Elyashar, Yuval Elovici, Mehran Roshandel, and Christoph Peylo. 2017. Creation and Management of Social Network Honeypots for Detecting Targeted Cyber Attacks. *IEEE Transactions on Computational Social Systems* 4, 3 (2017), 65–79. https://doi.org/10.1109/TCSS.2017.2719705

[45] Sampsa Rauti. 2020. Towards Cyber Attribution by Deception. *Hybrid Intelligent Systems* (2020), 419–428. https://doi.org/10.1007/978-3-030-49336-3_41

[46] Penny Ross. 2013. *The use of honey tokens in database security*. Number COM2013-0428 in ATINER conference paper series. Atiner.

[47] Zimmeck Sebastian, Story Peter, Smullen Daniel, Wang Abhilasha, Ravichanderand Ziqi, Reidenberg Joel, Russell N. Cameron, and Sadeh Norman. 2019. MAPS: Scaling Privacy Compliance Analysis to a Million Apps.

[48] Selenium. 2020. Selenium WebDriver. https://www.selenium.dev/. [Online; last accessed 15-October-2020].

[49] Selenium. 2022. Locator strategies. https://www.selenium.dev/documentation/webdriver/elements/locators/. [Online; last accessed 06-September-2022].

[50] Craig Smith. 2020. 55 Slack Statistics and Facts (2020) | By the Numbers. https://expandedramblings.com/index.php/slack-statistics/. [Online; last accessed 04-January-2021].

[51] L. Spitzner. 2002. *Honeypots: Tracking Hackers*. Addison-Wesley Longman Publishing Co., Inc., USA.

[52] SRLabs. 2019. smart spies: alexa and google home expose users to vishing and eavesdropping. https://srlabs.de/bites/smart-spies/

[53] Rolf Van Wegberg, Samaneh Tajalizadehkhoob, Kyle Soska, Ugur Akyazi, Carlos Hernandez Ganan, Bram Klievink, Nicolas Christin, and Michel Van Eeten. 2018. Plug and prey? measuring the commoditization of cybercrime via online anonymous markets. In *27th USENIX security symposium (USENIX security 18)*. 1009–1026.

[54] Paul Wagenseil. 2019. Discord 'Spidey Bot' Malware Is Stealing Usernames, Passwords. https://www.tomsguide.com/news/discord-spidey-bot-malware-is-stealing-usernames-passwords. [Online; last accessed 18-January-2021].

[55] Yang Wang, Jun Zheng, Chen Sun, and Srinivas Mukkamala. 2013. Quantitative Security Risk Assessment of Android Permissions and Applications. In *Data and Applications Security and Privacy XXVII*, Lingyu Wang and Basit Shafiq (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 226–241.