A Study on Impact of User-generated Content on Cybersecurity
ユーザ生成コンテンツがサイバーセキュリティに及ぼす影響に関する研究

Hiroki NAKANO
中野 弘樹

Doctoral Dissertation
博士論文

Graduate School of Environment and Information Sciences,
Yokohama National University
on March, 2024

Supervisor: Professor Tsutomu MATSUMOTO

ABSTRACT

With the rapid development of the Internet, user-generated content (UGC) has become an integral part of modern society. In particular, the number of users of social networking services is rapidly increasing, reaching 4.9 billion users by 2023. The main characteristic of UGC is that individuals and communities can freely disseminate their opinions and knowledge, facilitating the diversification of information compared to the past when the media was the center of information sharing. However, UGC has been pointed out to have reliability and quality issues, and appropriate measures are required from the perspective of cyber security. On the other hand, there are many successful cases where new services and communities have been created through the successful use of UGC by companies and general users. Appropriate collection and analysis of these UGC information may contribute to the development of traditional countermeasure technologies related to cyber security.

This paper investigates the impact of UGC on cybersecurity from both positive and negative perspectives. First, based on the characteristics of UGC services, we organized the types of platforms and classified them into five types: Social Networking Service, Video Sharing Service, Online Forum Service, Blogging Service, and User Review Service. Then, we discussed the actual security threats that could occur for each type and the related research and actual conditions that have been conducted so far.

Next, we investigate the potential negative impact on cybersecurity. Specifically, we investigate whether vulnerable source code on online forums affects application vulnerabilities. The investigation results show that 15.8% of applications with SSL implementation vulnerabilities (improper hostname validation), 31.7% of applications with SSL certificate validation vulnerabilities, and 3.8% of applications with WEBVIEW remote code execution vulnerabilities contained vulnerable code snippets on Stack Overflow. In other words, the research revealed that inappropriate UGC information on online forums affected Android app security issues.

We then investigate attackers who operate across multiple UGC platforms. The investigation revealed that attackers exploit contextual events to grab users' attention and direct them to typical malicious websites (e.g., information theft, survey scams, installation of suspicious browser plug-ins, etc.) across platforms. We also confirmed that the current measures taken by UGC platform operators are inadequate, as 87.8% of those malicious sites have not been removed after more than 100 days, and 69.0% of malicious UGC remains and is accessible to them. These results reveal examples of attackers deploying cyber attacks on multiple UGC platforms, causing harm to a large number of users.

In addition, we investigate the possibility of a positive impact on cybersecurity. Using Twitter as a new observation point, we collect shared cybersecurity threat information and compare and evaluate it with existing countermeasure technologies. The results of a three-month research experiment confirmed that 31,960 URLs shared on Twitter were later detected by anti-virus engines. They were information that would not have been covered by existing technologies, and the UGC platform proved capable of collecting a large amount of useful phishing attack information. Successful use of this information will lead to the development of countermeasure technologies for phishing attacks.

Finally, based on the research and analysis conducted, we discussed the potential for future countermeasures and utilization of UGC platforms in cybersecurity. The development of UGC platforms has expanded opportunities for information sharing, but it also poses new security threats such as the spread of misinformation and privacy violations. To address these threats, three approaches are important: technical measures, legal regulations, and user education. Maintaining the proper balance between the free distribution of information and the protection of individual privacy is essential for the healthy evolution of UGC platforms. In the future, we would like to return our research results to society by cooperating with UGC platform operators and security vendors to develop cybersecurity countermeasure technologies.

論文要旨

インターネットの急速な発展に伴い，ユーザ生成コンテンツ（UGC）は現代社会において密接に関わるものとなっている．特に，ソーシャルネットワーキングサービスでは，利用者が急増しており，2023 年には 49 億人の利用者数となる．UGC の主な特徴は，個人やコミュニティが自由に意見や知識を発信できる点にあり，従来メディアが情報共有の中心であった時代と比較して，情報の多様化が促進されている．しかしながら，UGC には信頼性や品質の問題が指摘されており，サイバーセキュリティの観点からも，適切な対策が求められる．一方で，UGC を企業や一般ユーザが上手く活用することで，新たなサービスやコミュニティが生まれ，成功している事例も多数存在する．これらの UGC の情報を適切に収集・分析することで，サイバーセキュリティに関連する従来対策技術の発展に貢献できる可能性がある．

本論文では，UGC がサイバーセキュリティに及ぼす影響を良い面と悪い面の両方の観点から調査を行う．まず，UGC のサービスの特性からプラットフォームの種別について整理を行い，Social Networking Service，Video Sharing Service，Online Forum Service，Blogging Service，User Review Service の 5 種類に分類した．そして，各種別で実際に起こり得るセキュリティ脅威とこれまでに取り組まれた関連研究，実態について論じた．

次に，サイバーセキュリティに悪い影響を及ぼす可能性について調査を行う．具体的には，オンラインフォーラム上の脆弱なソースコードがアプリケーションの脆弱性に影響を与えているかどうかを調査する．調査結果では，SSL 実装の脆弱性（不適切なホスト名検証）を持つアプリの 15.8%、SSL 証明書検証の脆弱性を持つアプリの 31.7%，WEBVIEW のリモートコード実行の脆弱性を持つアプリの 3.8%が，Stack Overflow 上の脆弱性があるコードスニペットを含んでいたことを明らかにした．つまり，オンラインフォーラム上の UGC の適切ではない情報が Android アプリのセキュリティ問題に影響を与えていることを明らかにした．

そして，複数の UGC プラットフォームを横断して活動する攻撃者について調査を行う．本調査により，攻撃者はユーザの注意をひくようなイベントを文脈に悪用して，プラットフォームを横断して典型的な悪性 Web サイト（情報窃取、アンケート詐欺，不審なブラウザプラグインのインストールなど）へ誘導していることが判明した．また，それらの悪性サイトは 100 日以上経過しても 87.8%が削除されず，悪性な UGC は 69.0%が残ったままであり，それらにアクセス可能であることから，現状の UGC プラットフォーム事業者の対策が不十分であることを確認した．これらの結果により，攻撃者が複数の UGC プラットフォームにサイバー攻撃を展開しており，多数のユーザへ被害を与えている実例を明らかにした．

さらに，サイバーセキュリティに良い影響を及ぼす可能性について調査を行う．Twitter を新たな観測地点として利用し，共有されているサイバーセキュリティの脅威情報を収集し，既存対策技術と比較評価を行う．三か月の調査実験の結果，Twitter 上で共有されていた 31,960 件の URL が後にアンチウイルスエンジンによって検知されたことが確認された．それらの情報は，既存技術ではカバーできていないような情報であり，UGC プラットフォームでは，有用なフィッシング攻撃の情報を多数収集可能であることが判明した．これらの情報を上手く活用することで，フィッシング攻撃の対策技術の発展につながる．

最後に，今回の調査と分析をもとに，サイバーセキュリティにおける今後の UGC プラッ

トフォームの対策と活用方法の可能性について論じた．UGC プラットフォームの発展は情報共有の機会を拡大しているが，誤情報の拡散やプライバシー侵害などの新たなセキュリティ脅威をもたらしている．これらの脅威に対処するためには，技術的対策，法的規制，ユーザー教育の三つのアプローチが重要であり，情報の自由な流通と個人のプライバシー保護のバランスを適切に保つことが，UGC プラットフォームの健全な進化には不可欠であると言える．今後は，UGC プラットフォーム事業者やセキュリティベンダと協力してサイバーセキュリティ対策技術の発展に努めることで，研究成果を社会に還元したい．

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

With the rapid development of the Internet, User-generated Content (UGC) has become an integral part of modern society. UGC is a generic term for content created and posted independently by users on various platforms and services on the Internet. With the explosive growth of the Internet, UGC plays an important role in information sharing and communication, and is active in a wide range of fields, including websites, blogs, forums, social media, video sharing sites, and review sites. In particular, social networking services show no signs of slowing down, from 2.7 billion users in 2017 to 4.9 billion users in 2023, a 1.5-fold [1], with an expected 6 billion users in 2027.

The main characteristic of UGC is that individuals and communities can freely disseminate their opinions and knowledge. In conventional media, information is selected and organized by professional editors such as broadcasters and newspapers, published as articles, and people refer to this information [2]. The UGC allows non-professional users to proactively disseminate information and share diverse perspectives and opinions. As a result, people are expected to create new values and cultures, as well as promote diversification of information.

However, reliability and quality issues have been pointed out for UGC. This is because the expertise and credibility of information providers are often uncertain, and there is a risk of spreading misinformation and prejudice. For example, misinformation about news and social conditions spread from uncertain sources can cause people to behave incorrectly. Recently, cases of misinformation about COVID-19 being spread and correct information being perceived as misinformation by the public have been reported and have become a problem [3, 4] Some UGC also violates laws and ethics, such as copyright infringement, invasion of privacy, hate speech, and slander. This illegal and inappropriate content may violate users' rights and cause social and legal troubles Actually, issues such as the "legality" of the act of content downloading, the "protection" of content creators, and the "damage" allegedly brought to the lives of creators by piracy have been discussed [5]. Addressing these issues is an important challenge for the sustainable development of UGC. Therefore, platform operators are required to implement appropriate monitoring and screening systems, remove illegal and inappropriate content, and provide warnings and guidance to users. Some platforms have also developed a system to evaluate the quality and authenticity of UGC by utilizing AI technology and machine learning, but even now, however, this is not a sufficient countermeasure, as reports of malicious postings continue to emerge.

On the other hand, there are many examples of UGC being successfully utilized by companies and the general public [6, 7]. Companies can leverage UGC content to gain a credible means of promotion through word-of-mouth and recommendations; UGC-based campaigns can encourage consumer engagement and increase brand trust and awareness. For example, Starbucks held a "White Cup Content" for new designs, inviting people to decorate and draw whatever they wanted on white cups [8]. The winning design would be used as a limited edition cup in Starbucks stores, but to do so, the participants had to purchase a drink in a white cup and post it on the social networking service. The UGC-based campaign led to sales promotion, new design development, and increased brand awareness. In addition, through UGC, it is possible to collect user feedback and requests, which can be used to develop and improve products and services. Thus, depending on how they are used, companies can offer products that meet the needs of consumers. Furthermore, UGC platform users facilitate the formation of online communities where people with the same interests and concerns gather and share information and opinions. For example, users can find new ideas and solutions by sharing their knowledge and experiences with each other. Actually, people with the same interests are connecting on UGC platforms to launch new services and establish new communities. In general, UGC plays an important role in the Internet society, and its advantages and risks coexist. In the future, efforts aimed at improving reliability and quality will become even more important, and appropriate measures must be taken from the perspective of cybersecurity.

1.2 Motivation

This thesis investigates the impact of UGC on cybersecurity from both positive and negative aspects. First, in terms of the negative impact of UGC on cybersecurity, the analysis focus on Stack Overflow, an online forum on the web that is used to exchange information among developers. Android application developers are known to copy code snippets posted on Q&A sites and use them directly in their applications. However, if the code snippet is vulnerable, the Android app containing the vulnerable code snippet may have the same vulnerability. Nevertheless, the impact of such vulnerable snippets on Android apps has not been investigated in depth. We determine whether there are instances where UGC on online forums can cause actual application vulnerabilities to occur. Next, we analyze the characteristics of falese information spread by attackers, especially on social networking services. With the growth of UGC platforms, people are increasingly relying on UGC rather than search engines to find and access information on the web. In addition, attackers can exploit malicious content on highly populated UGC platforms to spread web-based social engineering (SE) attacks widely. Therefore, we investigate the actual situation across multiple UGC platforms to identify the false information that attackers are spreading. On the other hand, in terms of the positive impact of UGC on cybersecurity, we evaluate whether UGC shared on social networking services is useful in cybersecurity measures. The recent increase in phishing attacks via email and short message service (SMS) shows no sign of abating. Therefore, the first step in combating the ever-increasing number of phishing attacks is to collect more phishing cases reaching end users and understand their characteristics. To understand these characteristics, we collect reports of phishing attacks shared on Twitter and evaluate them against existing countermeasure technologies.

1.3 Contributions

Towards Finding Code Snippets on a Question and Answer Website Causing Mobile App Vulnerabilities (Chapter 3). In Chapter 3, we investigate the correspondence between vulnerable code snippets and vulnerable apps. We collect code snippets from the Q&A website, extract snippets of possible vulnerabilities, and calculate the similarity between those snippets and the bytecodes of vulnerable apps. Based on our experimental results, 15.8% of the apps with SSL implementation vulnerabilities (improper hostname validation), 31.7% of the apps with SSL certificate validation vulnerabilities, and 3.8% of the apps with WEBVIEW remote code execution vulnerabilities, which are Stack Overflow-derived vulnerabilities code snippets included in them. In the worst case, 4,844 apps contained vulnerabilities due to one type of problematic code snippet, accounting for 31.2% of all apps with that vulnerability that we collected. These findings reveal that inappropriate UGC information on online forums is affecting Android app security issues.

Exploring Event-synced Navigation Attacks across User-generated Content Platforms in the Wild (Chapter 4). In Chapter 4, we define and focus on event-synchronized navigation attacks, a type of web-derived SE attack that generates UGC containing links to malicious websites and delivers them synchronized with real-world events at specific times. We propose three steps to detect event-synchronized navigation attacks in real time by capturing traces of the attacks, which are inevitable for an attacker to lead a large number of people. We evaluate each step of the proposed system and finally confirm that the proposed system can classify malicious and non-malicious UGC with 97% accuracy. Furthermore, we conducted a comprehensive fact-finding survey on event-synchronized navigation attacks spread from UGC platforms with a large number of users. As a result, we found that 34.1% of malicious website FQDNs associated with event-synchronized navigation attacks were spread from two or more UGC platforms. We also found that 87.8% of FQDNs associated with typical malicious websites (e.g., information theft, survey fraud, installation of suspicious browser plug-ins, etc.) survived for more than 100 days, and despite the fact that malicious websites are frequently accessed by users, the UGC platform's countermeasures was also found to cover only 31.0% of the malicious UGC detected in this study. This research experiment confirmed that attackers spread numerous attacks starting from UGC on services with a large number of users, and that the current UGC platforms are insufficient to counter these attacks.

Understanding Phishing Reports from Experts and Non-experts on Twitter (Chapter 5). In Chapter 5, we propose an approach that uses Twitter as a new observation point to immediately collect and characterize phishing cases via email and SMS that bypass current anti-phishing measures and reach end users. Specifically, we propose CrowdCanary, a system that can extract phishing information (e.g., URLs and domains) structurally and accurately from phishing-related tweets from users who have actually discovered or encountered phishing. We operated CrowdCanary for three months and identified 35,432 phishing URLs from 38,935 phishing reports. Of these phishing URLs, 31,960 (90.2%) were later detected by anti-virus engines, demonstrating that CrowdCanary outperforms existing systems in both accuracy and quantity of threat extraction. We also analyzed users who shared phishing threats using the extracted phishing URLs and classified them into two groups: experts and non-experts. As a result, CrowdCanary was able to collect information specifically included in the reports of non-experts, such as information shared only by company brand names in tweets, information about phishing attacks contained only in images in tweets, and information about landing pages before redirects. Furthermore, a detailed analysis of the collected information on phishing sites revealed certain biases in the domain names and hosting servers of phishing sites, revealing new features that are useful for detecting unknown phishing sites. Our study and experiments demonstrate that UGC on social platforms can contribute to the development of security countermeasure technologies.

1.4 Outline

This thesis consists of the following parts. In Chapter 2, we organize what types of web platforms exist for UGC, introduce their impact on cybersecurity on a practical example basis, and discuss the issues addressed in this thesis. In Chapter 3, we collect source code fragments (code snippets) posted on Q&A sites and analyze their relevance to vulnerabilities in Android applications. In Chapter 4, we propose a system to detect malicious activities of attackers in the context of events that people pay attention to on multiple social platforms, and investigate the actual situation. In Chapter 5, we propose a system to extract reports of phishing attacks shared on Twitter from experts as well as non-experts, and evaluate it in comparison with existing techniques. Finally, in Chapter 6, we provide conclusions and future perspectives on this thesis as a whole.

Chapter 2

User-generated Content

2.1 Type of Platform

With the explosion of the Internet, UGC plays an important role in information sharing and communication, and is active in a wide variety of fields, including websites, blogs, forums, social media, video sharing sites, and review sites. In this section, we organize each characteristic of UGC along with the actual platforms. We have categorized them into five broad types: social networking services, blogging services, video sharing services, user review services, and online forum services, as shown in Table 2.1.

Social networking services (SNS) are platforms for people to communicate and share information through the Internet. SNSs allow users to connect with people such as friends, family, colleagues, and acquaintances, send and receive messages, share photos and videos, and exchange opinions and impressions. Typical platforms include Facebook, where users communicate mainly using their real names; Twitter (currently X), where users can post anonymously and freely; Instagram, where information is shared mainly through photos; and LinkedIn, whose main purpose is business use. SNSs have the largest number of users among UGC platforms, with 4.5 billion users worldwide in 2022 [1]. These services are used not only by individuals, but also by companies and organizations for marketing and public relations purposes. SNSs are characterized by their real-time and diffusion characteristics. Real-time is useful for news and events where breaking news is important because information can be transmitted instantly. Diffusion is the potential for information to spread to a large number of people through easy sharing. On the other hand, SNSs also present problems such as privacy concerns, the spread of fake news, and online slander.

Video sharing services are online platforms that allow users to upload, share, and watch video content over the Internet. Typical platforms include YouTube and DailyMotion, where a wide range of content is shared, including music videos, news, and sports, as well as Twitch, which specializes in gaming and e-sports. These platforms are used for a variety of purposes, including education, entertainment, business, and communication. Not only can users search and watch videos based on their interests and preferences, but users can also create their own channels and upload videos to share with other users. Video sharing services allow some content creators to generate revenue through advertising revenue and sponsorships, which is an incentive for users to share videos. On the other hand, copyright infringement by illegal video uploading and the spread of false information and hoaxes have had a significant impact on users and have become a social problem in recent years.

Online forum services are discussion and communication platforms available

Table 2.1: Platform List

Type	Platform Name
Social Networking Service	Twitter, Facebook, Instagram, LinkedIn
Video Sharing Service	YouTube, Twitch, DailyMotion
Online Forum Service	Reddit, Stack Overflow, Quora
Blogging Service	Blogger, WordPress, Medium
User Review Service	Amazon Review, TripAdvisor, GoogleMaps, Yelp

on the Internet. Typical platforms include Reddit, where communication takes place by creating groups called threads for each topic, Quora and Stack Overflow, where users can post and search answers to questions in specific fields, and others. Users can exchange information, post opinions and questions, and provide answers and comments on a variety of topics. Online forums range from those that focus on specific topics, hobbies, specialties, or communities to those that cover a wide range of topics. The characteristics of the main topics discussed on each platform differ, and the platform used is selected according to the user's own needs. Some people exchange their opinions on common topics, while others use the platform to allow users with no expertise to ask questions of those who do have expertise, and receive answers to their questions. On the other hand, users should be aware of the known risks of posting illegal content, directing users to malicious sites, and spreading false information.

Blogging services are platforms for individuals and organizations to disseminate information on the Internet and post their opinions, thoughts, daily life, hobbies, etc. Typical platforms include Blogger, Wordpress, Medium, etc., and they range from simple to highly functional services. Blogs are basically characterized by the fact that the administrator can freely update posts and third parties can comment on those posts. Blogs are also good for SEO (search engine optimization), and can attract a lot of traffic from search engines if appropriate keywords are used. Not only individuals, but also companies and organizations are using blogs to introduce their products and services, disseminate information, and communicate with their customers. Personal blogs are also used as a place to share hobbies and expertise, interacting with readers and forming communities. As with SNS, privacy, inappropriate comments, and slander are issues.

User review services are online platforms for sharing opinions and ratings about products and services. Typical platforms include Amazon reviews, where ratings about products are posted; TripAdvisor, where reviews are popular mainly for hotels and restaurants; Google Maps, where many ratings are posted for stores, tourist attractions, etc.; and Yelp, where reviews are posted mainly for facilities such as hospitals, beauty salons. These platforms are used by consumers to obtain information about products and services they are considering purchasing or using. Most user review services have their own rating systems. This function allows users to assign star ratings or rating points to products and services. These ratings are displayed as an average rating or ranking, allowing consumers to easily determine the quality and popularity of a product or service. On the other hand, user review services sometimes post inaccurate information or false reviews. This includes malicious reviews by competitors and reviews posted by companies to highly rate their own products and services.

2.2 Related Work on User-generated Content with a Focus on Cybersecurity

This section introduces the cyber security threats that lurk in the UGC platforms classified by type in the previous section, along with previous studies and reports.

2.2.1 Social Networking Service

The main cybersecurity threat related to social networking services is the proliferation of malicious posts by accounts created by attackers. Kurt et al. investigated Twitter accounts traded on the underground market and reported their findings [9]. A large number of accounts were used for phishing, fraud, and other activities, and they worked with Twitter Inc. to successfully suspend 95% of the account trading businesses investigated. Md. Sazzadur et al. also developed a novel application to protect Facebook users and reported on the results of its operation [10]. The application they developed detects malicious UGC with high accuracy that could not be detected by conventional countermeasure techniques such as blocklists, and they also identified a number of new attack campaigns that exploit the characteristics of Social Networking Service, and found that the rise of UGC platforms has led to a diversification of attacks. Since Social Networking Service allows users to create accounts freely, it is easy for attackers to create fake accounts, attract users' attention, and lead many users to malicious sites.

2.2.2 Video Sharing Service

Video sharing service, there are attackers who direct users to malicious sites in video comment sections and engage in fraudulent activities in increasing video engagement. Tulio et al. propose a method to identify with high accuracy accounts that post in YouTube comment sections that direct users to malicious sites [11]. Popular UGC on video sharing services are often exploited by attackers in this way because of the large number of accesses they receive from users. Dhruv et al. also reported that there is a group of attackers who are exploiting the system of YouTube to receive incentives based on the number of times a video is viewed [12]. The attackers are working to illegally earn engagement by having YouTube videos play as advertisements when watching videos on a different Video Sharing Service called 123Movies. In this way, attackers exploit the system's mechanisms to gain money or use it as a venue for influx of users to malicious sites, and operators of platforms with a large number of users need to take especially firm countermeasures.

2.2.3 Online Forum Service

Online forum services allow users to post UGC on free topics, but there are also numerous spam posts, and service providers are forced to take measures to prevent them. Felix et al. report the results of an investigation into the presence of vulnerable source code in posts on Stack Overflow, one of the online forum services for developers [13]. Many source codes were found to have security issues, revealing that while developers can easily search for source code on online forums, many security risks exist. Eshwar et al. report on the results of a large-scale analysis of posts deleted by moderators on Reddit [14]. Macro code violations included personal attacks, misogyny, racism and homophobia, while meso code violations included directing users to malicious sites, confirming the efforts of operators to keep the platform healthy from a variety of perspectives. In online forum services, users should similarly verify the authenticity of information and discard only correct information.

2.2.4 Blogging Service

Blogging services allow users to freely post blog posts, which can be used by attackers to evade detection of their malicious activities. Malwarebytes Labs has reported on an actual case where a user was redirected from content on a free blog service to the Explot Kit through the intermediary of a fake advertiser [15]. Blog visitors are automatically redirected to malicious sites prepared by attackers when they access UGC on blog services such as those mentioned above. If a vulnerability in the blog visitor's browser is exploited, it could lead to damage such as virus infection. Palo Alto Networks researchers have also reported that the malware's encrypted C2 information is posted in UGC on blogs [16]. The ability to reference information on blog services and communicate with C2 servers is present in some of the malware's processes, and blog services have been exploited as part of the attackers' detection evasion logic. Since these attackers' malicious activities are performed on legitimate blog service URLs, anti-virus may fail to detect them as malicious behavior.

2.2.5 User Review Service

User review services have been problematic in manipulating consumers through fake reviews by non-existent user accounts. Arjun et al. report an evaluation of Yelp's fake review detection algorithm Yelp [17]. In fact, the user review service has its own algorithm for countermeasures against fake reviews, and the aforementioned report found that Yelp's algorithm is reasonable and works well. Himangshu et al. also evaluate methods proposed to date to combat fake reviews [18]. The report found that both basic research approaches and approaches and regulations used in real business have failed to provide a limited level of protection in preventing the damage caused by fake reviews. Thus, it can be seen that although many businesses are attempting to take countermeasures against UGC in the user review service, they are currently inadequate.

2.3 Challenges in Analyzing User-generated Content

In this section, we discuss our roadmap and the challenges in analyzing UGC. Figure 2.1 summarizes the types of UGC and the position of each study in this thesis. Since the characteristics of UGC vary by platform, it is not easy to analyze UGC in relation to each other and to other events. Therefore, we need to collect the information necessary for the facts we wish to clarify in an analyzable format. In each chapter, we describe the characteristics of the UGC to be analyzed and the challenges that need to be solved in order to do so. We address the challenges and investigate the realities so that we can develop future cybersecurity measures.

First, in Chapter 3, we investigate the source code posted by any user on Stack Overflow, a Q&A site for developers, which is one of the online forums. Due to the nature of the online forum, as an open forum, anyone can post source code. On the other hand, if a vulnerability exists in the posted source code, and the developer uses that code to implement an application, the application will also be vulnerable without the developer's knowledge. However, the code implemented in the application is written in the application as compiled bytecode, not source

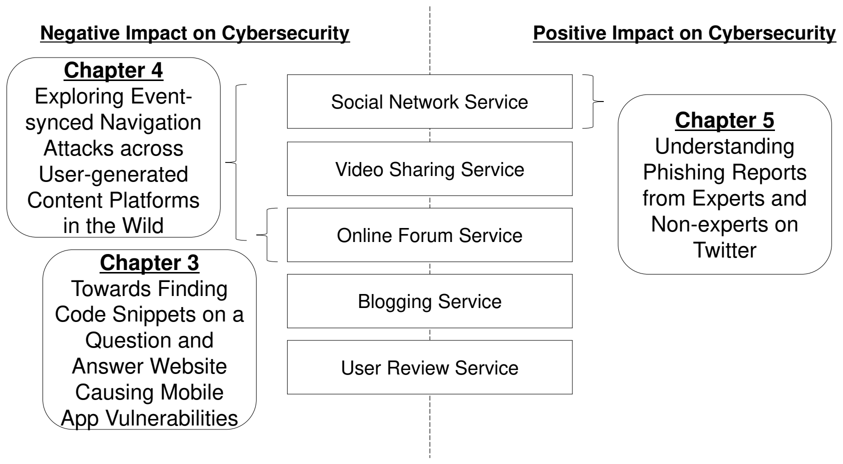

Negative Impact on Cybersecurity
Positive Impact on Cybersecurity
Chapter 4
Exploring Event-synced Navigation Attacks across User-generated Content Platforms in the Wild
Chapter 3
Towards Finding Code Snippets on a Question and Answer Website Causing Mobile App Vulnerabilities
Social Network Service
Video Sharing Service
Online Forum Service
Blogging Service
User Review Service
Chapter 5
Understanding Phishing Reports from Experts and Non-experts on Twitter

Figure 2.1: Overall Picture of This Thesis

code. Therefore, in order to determine whether the source code on the online forum was used to implement the application, a method is needed to calculate the similarity between the different forms of information, source code and byte code. We solve the aforementioned challenge for the purpose of investigating the existence of application vulnerabilities caused by UGC information.

Next, in Chapter 4, we investigate the actual state of malicious activities by attackers through a cross-sectional analysis of UGC platforms of different forms, such as social networking service, video sharing service, and online forum service. Attackers need to post contents that attract users' attention on multiple UGC platforms in order to induce a large number of users to malicious sites and obtain money and authentication information. However, since these UGC platforms have different characteristics, they must be analyzed in conjunction with UGC based on an understanding of their characteristics. We identify attackers who are exploiting UGC platforms to conduct malicious activities and solve the aforementioned challenges for the purpose of investigating the actual situation and characteristics of the attacks.

Chapter 5 then evaluates the effectiveness of cyber security threat information shared on the Social Networking Service. Due to the nature of UGC platforms, where users can freely share information, there are many cases of beneficial information sharing by well-meaning users. However, in many cases, users on UGC platforms share information in different media, such as text and images. Furthermore, there are many posts that are not related to cybersecurity at all, such as daily posts, and it is necessary to extract useful threat information for cybersecurity countermeasures from these large-scale posts. For the purpose of evaluating whether information on UGC platforms, where information is shared through diverse media, is useful for countermeasure techniques as a new observation point, we solve the aforementioned challenge.

Chapter 3

Towards Finding Code Snippets on a Question and Answer Website Causing Mobile App Vulnerabilities

3.1 Introduction

The popularity and spread of mobile devices have led to a huge number of mobile apps. Various mobile app developers, from professionals to amateurs, register their apps in app markets, and those apps are downloaded by a great number of users through the markets.

However, these developers, especially inexperienced ones, can create apps with serious vulnerabilities, for example, allowing malicious apps access to a user's personal information. On the other hand, market providers can take countermeasures against these vulnerable apps. For example, Google Play [19] warns developers about creating vulnerable apps and may ban vulnerable apps if the developers do not take proper precautions [20, 21, 22]. Despite such actions, many app markets have been reported to indeed offer numerous vulnerable apps [23, 24, 25].

We envision that one possible source of these vulnerabilities may be community websites for software developers. These websites, also called question-and-answer (Q&A) websites, provide the developers opportunities to discuss, ask, and answer questions regarding app developments and have grown in popularity. Indeed, the rich source of information given by the public discussions on these websites often provides quick solutions to the developers. Inexperienced developers especially tend to seek direct help and advice with ready-to-use code snippets from these websites. Moreover, they may simply copy such code snippets and use them in their own apps without checking their security. Thus, vulnerable snippets on the Q&A websites may be causing some apps' vulnerabilities.

In this paper, we investigate the reuse of code snippets from a Q&A website into the mobile apps to see if vulnerable snippets are indeed causing vulnerabilities of the apps. For the investigation, we first collected 243,589 code snippets from Stack Overflow [26], a representative Q&A website. From the collected snippets, we selected 209 vulnerable snippets that can cause app vulnerabilities. Additionally, we collected 61,910 apps from two Android app markets (Google Play and Qihoo 360 Mobile Assistant) including 47,081 free apps and 14,829 paid apps and identified 8,275,112 vulnerable classes in 48,333 apps by using a vulnerability scanner. Finally, we investigated the correspondence between the vulnerable code snippets and the collected apps by using a novel technique to compare bytecodes in each vulnerable class with the vulnerable snippets.

Our Contributions:

- We collected code snippets from Stack Overflow, analyzed a connection between code snippets and vulnerable apps, and investigated vulnerabilities attributable to reusing code snippets.

- We found that 15.8% of all evaluated apps that have SSL implementation vulnerabilities (improper host name verification), 31.7% that have SSL certificate verification vulnerabilities, and 3.8% that have WEBVIEW remote code execution vulnerabilities contain the possibly vulnerable code snippets from Stack Overflow.
- We designed and implemented a fully automated large-scale processing for analyzing correspondence between code snippets and Android apps.
- Our experimental results clarified that copy&paste-based reusing of code on a Q&A website is one factor causing app vulnerabilities.

3.2 Background

3.2.1 Code Snippets on Q&A Website

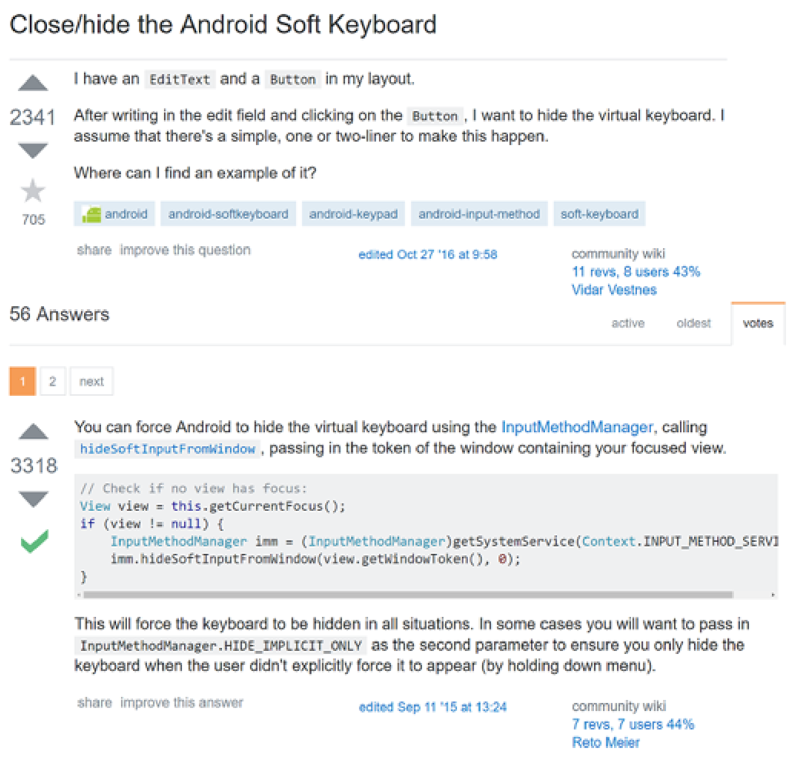
A code snippet is a fragment of code that is ready-to-use. By reusing code snippets, developers can create apps efficiently. Developers often use code snippets created by others as well as themselves.

A Q&A website for developers has a lot of code examples that developers can use easily. Developers can post questions on the Q&A websites, and other developers may answer these questions with code snippets for solving the questioners' problems. Additionally, it is possible to search already posted questions and answers by using keywords, tags, and so on. Also, users can refer to the evaluation of the answers with code snippets based on the votes by other users. Our study focused on Stack Overflow [26], which is one of the most popular Q&A websites for software developers. Code snippets on Stack Overflow are surrounded by `<code>` tags and can therefore easily be crawled. Figure 3.1 illustrates an example of a question and a corresponding answer posted on Stack Overflow. One user posted a question that arose during the development of an Android application, and another user posted the code for the answer.

3.2.2 Comparison of Android and iOS

We compare Android and iOS in terms of application development and market characteristics.

Application Development. Android applications are developed primarily in Java and Kotlin. These are object-oriented programming languages, many developers are already using these languages, and many developers are actively discussing them on Stack Overflow. iOS applications are developed in Swift and Objective-C. Swift is a new language developed by Apple that enables concise and efficient coding. The same has been discussed in iOS app development on Stack Overflow, although in smaller quantities than in Android. Android is an open source platform and developers have a great deal of freedom in the customization and flexibility of their apps. However, this requires additional effort to accommodate different device sizes, screen resolutions, and OS versions. This is known as fragmentation and is one major challenge for Android developers. iOS is a closed source platform and must follow Apple's guidelines and rules. This reduces the need to accommodate a diversity of device and OS versions, but on the other hand, it is characterized by limited freedom of customization.



Close/hide the Android Soft Keyboard

2341

705

I have an `EditText` and a `Button` in my layout.

After writing in the edit field and clicking on the `Button`, I want to hide the virtual keyboard. I assume that there's a simple, one or two-liner to make this happen.

Where can I find an example of it?

56 Answers

3318

You can force Android to hide the virtual keyboard using the InputMethodManager, calling `hideSoftInputFromWindow`, passing in the token of the window containing your focused view.

```
// Check if no view has focus:
View view = this.getCurrentFocus();
if (view != null) {
    InputMethodManager imm = (InputMethodManager)getSystemService(Context.INPUT_METHOD_SERVI
    imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
}
```

This will force the keyboard to be hidden in all situations. In some cases you will want to pass in `InputMethodManager.HIDE_IMPLICIT_ONLY` as the second parameter to ensure you only hide the keyboard when the user didn't explicitly force it to appear (by holding down menu).

Figure 3.1: Example of Question Post (Top) and Answer Post (Bottom) on Stack Overflow

Market Characteristics. Android is ahead of iOS in global market share, especially in emerging and developing markets [27]. However, users generally tend to pay less for apps and in-app purchases. On the other hand, iOS has a strong position in certain markets, especially in North America and Western Europe. Users generally tend to pay more for apps and in-app purchases. The review process for the Google Play Store, the official Android marketplace, is relatively quick, allowing new apps and updates to be brought to market quickly. However, this means that the quality of the apps will vary, and users may encounter apps with inappropriate content or bugs. In contrast, the Apple App Store's review process is known to be rigorous and time-consuming. This ensures that the quality of apps in the store remains high. However, when releasing new apps or updates, the review process needs to be included in the plan.

These perspectives show that Android and iOS have very different application development and market characteristics in terms of mobile operating systems. We consider Android to be more prone to security threats due to the freedom of app development and the looseness of market restrictions. Therefore, this study focuses on security threats to Android apps and analyzes the impact of source code on the UGC platform.

3.2.3 Android Apps Vulnerabilities

There are various categories of vulnerabilities in Android apps. Table 3.1 lists the Top Ten Mobile Risks in 2016 reported by the Open Web Application Security Project (OWASP) Mobile Security Project [28]. This table shows that client attacks, network attacks, and server attacks seriously threaten the security of Android apps. For example, vulnerabilities generated defects in `AndroidManifest.xml`, inappropriate implementations of Secure Sockets Layer (SSL) or Transport Layer

Table 3.1: Top 10 Mobile Threats in 2016

Kinds of Vulnerability	Kinds of Attack
M1 - Improper Platform Usage	Client Attacks
M2 - Insecure Data Storage	Client Attacks
M3 - Insufficient Transport Layer	Network/Traffic Attacks
M4 - Insecure Authentication	Client/Server Attacks
M5 - Insufficient Cryptography	Client/Network/Server Attacks
M6 - Insecure Authorization	Client/Server Attacks
M7 - Client Code Quality	Client Attacks
M8 - Code Tampering	Client Attacks
M9 - Reverse Engineering	Client Attacks
M10 - Extraneous Functionality	-

Security (TLS), inadequate implementations of `WebView`, and so on. Almost all these vulnerabilities were the fault of careless developers. Additionally, there are a lot of existing tools and services for detecting these vulnerabilities. In this paper, we use Vulnerability Scanner AndroBugs [29] and focused on three kinds of vulnerabilities attributable to the method of implementation.

3.3 Method

3.3.1 Overview

An overview of our approach is shown in Figure 3.2. In the collection phase, we collected code snippets in answer posts from Stack Overflow (Figure 3.3) and downloaded apps from multiple app markets. In the analysis phase, we selected potentially vulnerable code snippets from all collected snippets and extracted features from them. Additionally, we checked vulnerable classes in collected apps by using AndroBugs and extracted features from vulnerable classes flagged by AndroBugs. In the comparison phase, we calculated the similarity between features of code snippets and features of app and investigated code snippets that are possible causes of vulnerabilities. Output data include similarity between bytecode and code snippets and also kinds of vulnerability. We can check whether vulnerabilities are attributable to reusing vulnerable code snippets or not by analyzing output data, e.g., we can determine if codes in apps were reused from a Q&A website by setting a threshold of similarity value.

3.3.2 Method of Calculating Similarity between Code Snippets and Bytecode

We explain our proposed method for calculating similarity between code snippets and bytecode.

It is difficult to investigate whether code snippets have vulnerabilities or not because existing vulnerability scanners can only scan compilable source codes or bytecode after compilation. Likewise, source codes on websites are small pieces in many cases, so converting source codes into bytecode automatically is also

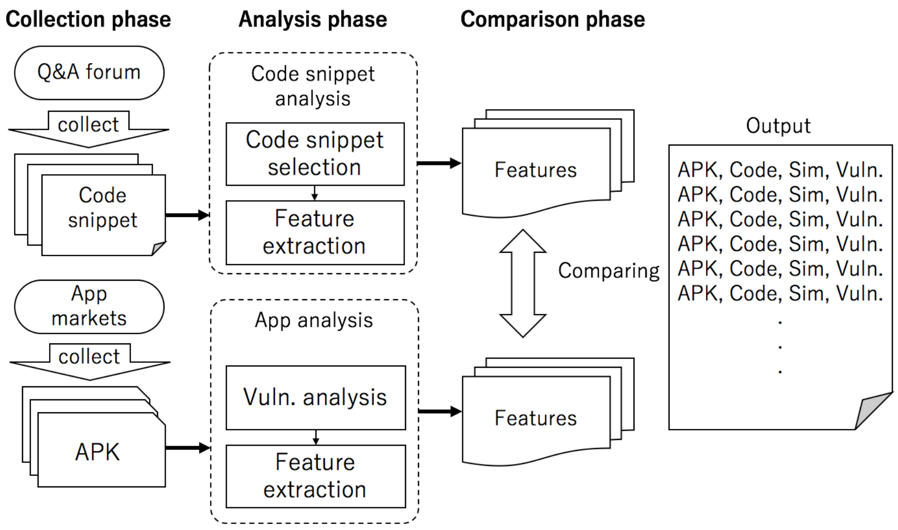
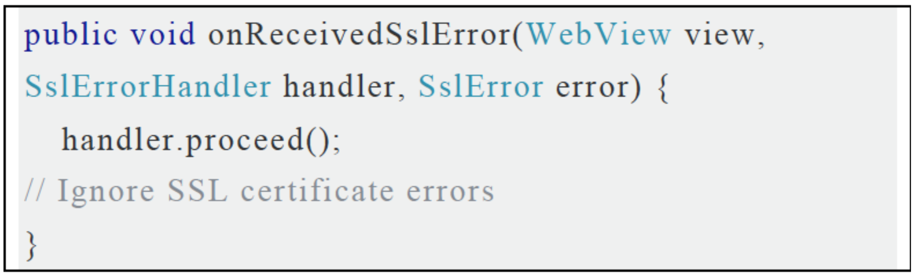


Figure 3.2: Overview of Our Approach



```
public void onReceivedSslError(WebView view,
SslErrorHandler handler, SslError error) {
   handler.proceed();
// Ignore SSL certificate errors
}
```

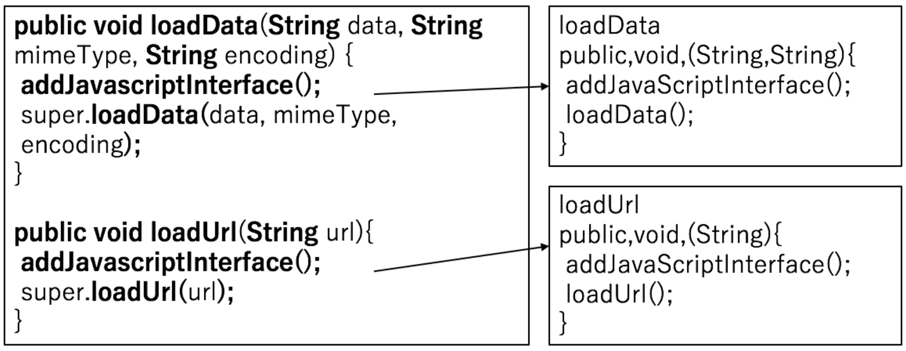
Figure 3.3: Example of Targeted Code Snippets

difficult. Furthermore, developers are assumed to modify code snippets and implement their apps using somewhat modified code snippets. A method of comparing and evaluating partially processing sequences or features needs to be used to specify code snippets that cause vulnerabilities of apps. Therefore, we evaluated similarities between code snippets and bytecode by using common features of method call sequences and method definitions.

We used Longest Common Subsequence (LCS) and Levenshtein Distance (LD) to evaluate method call sequences. Additionally, we also used the rate of concordance for types of the method definitions (TMD) and the similarity degree of the name of method definitions (NMD) to evaluate method definitions. Finally, we expressed the evaluation formula (1), which consists of LCS, LD, TMD, and NMD values.

The method of extracting features from code snippets is shown in Figure 3.4. We extracted method calls in the method body, the name of the method, and type modifier characters in method definitions from code snippets. The method of extracting features from bytecode in apps is shown in Figure 3.5. We also extracted information from bytecode in the same manner as above.

Comparative Approach to Information Group of Methods. The comparative approach to the information group of methods is shown in Figure 3.6. In this instance, the left side is a code snippet that has two method definitions, and the right side is a bytecode that has four method definitions. We compare each method sequentially and find the most similar method pairs between a code snippet and a bytecode. In this case, ① and ⑦ are the most similar pairs. We

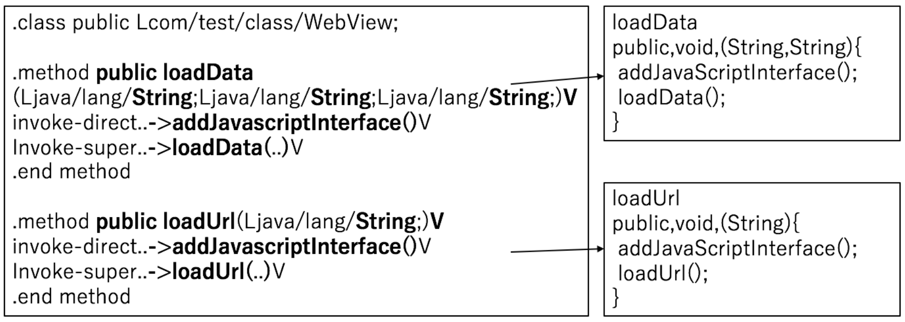


```
public void loadData(String data, String
mimeType, String encoding) {
 addJavascriptInterface();
 super.loadData(data, mimeType,
 encoding);
}

public void loadUrl(String url){
 addJavascriptInterface();
 super.loadUrl(url);
}
```

```
loadData
public,void,(String,String){
 addJavaScriptInterface();
 loadData();
}
```

```
loadUrl
public,void,(String){
 addJavaScriptInterface();
 loadUrl();
}
```

Figure 3.4: Method of Extracting Features from Code Snippets

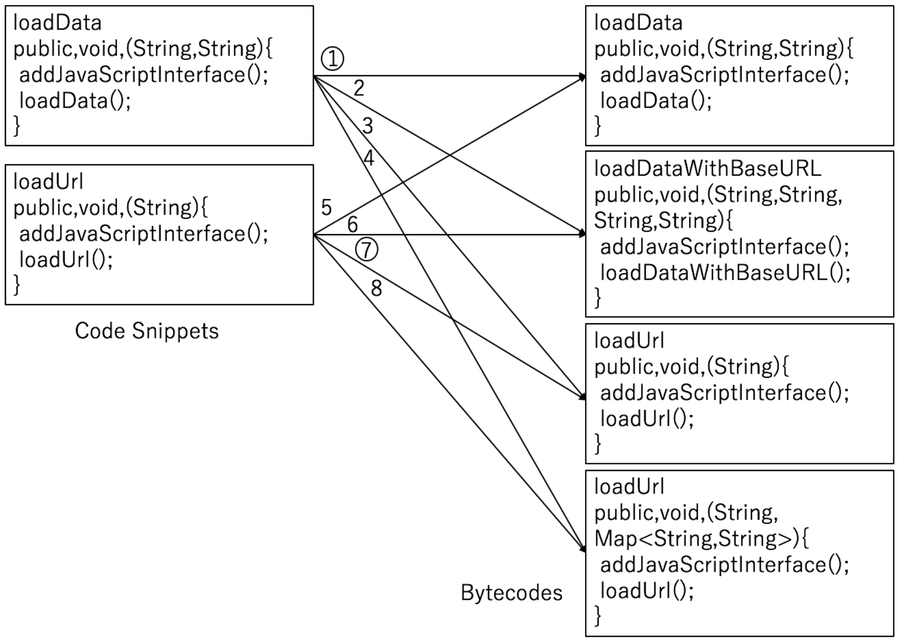


```
.class public Lcom/test/class/WebView;

.method public loadData
(Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;)V
invoke-direct..->addJavascriptInterface()V
Invoke-super..->loadData(..)V
.end method

.method public loadUrl(Ljava/lang/String;)V
invoke-direct..->addJavascriptInterface()V
Invoke-super..->loadUrl(..)V
.end method
```

```
loadData
public,void,(String,String){
 addJavaScriptInterface();
 loadData();
}
```

```
loadUrl
public,void,(String){
 addJavaScriptInterface();
 loadUrl();
}
```

Figure 3.5: Method of Extracting Features from Bytecode in Apps



```
loadData
public,void,(String,String){
 addJavaScriptInterface();
 loadData();
}
```

```
loadUrl
public,void,(String){
 addJavaScriptInterface();
 loadUrl();
}
```

```
loadData
public,void,(String,String){
 addJavaScriptInterface();
 loadData();
}
```

```
loadDataWithBaseURL
public,void,(String,String,
String,String){
 addJavaScriptInterface();
 loadDataWithBaseURL();
}
```

```
loadUrl
public,void,(String){
 addJavaScriptInterface();
 loadUrl();
}
```

```
loadUrl
public,void,(String,
Map<String,String>){
 addJavaScriptInterface();
 loadUrl();
}
```

Figure 3.6: Comparative Approach to Information Group of Methods

describe the method of calculating similarities later.

Evaluation Formula. We calculate similarities between code snippets and bytecode using the following equation (1). The number of method definitions in code snippets is n and the number of method definitions in one class of apps is m. We describe each distance, LCS, LD, TMD and NMD, respectively.

$$Score = \frac{\sum_{i=1}^{n} \max_{0 \leq j \leq m}(LCS_j + LD_j + TMD_j + NMD_j)}{4} \tag{3.1}$$

Note that specific features may greatly affect the equation because no similarity is independent of others. Our future work is to define the formula while considering the dependency of these similarity definitions.

Longest Common Subsequence (LCS). The LCS is an algorithm to find the longest subsequence common to all sequences in a set of sequences. It is suited to evaluate partial method call sequences and able to evaluate appropriate similarities if developers modify original code snippets, e.g., inserting other method calls or reordering call sequences.

The number of method calls in code snippets is S, and the longest common subsequence can be expressed as the following equation.

$$LCS_j = \frac{S_i}{\sum_{k=1}^{n} S_k} \times \frac{\max_{0 \leq l \leq m} LCS_l}{S_i} \tag{3.2}$$

Levenshtein Distance (LD). The LD is a string metric for measuring the difference between two sequences. Similar to the longest common subsequence, it is also suited to evaluate partial method call sequences and able to evaluate appropriate similarities if developers modify method call sequences. The number of method calls in one class of apps is A, and the LD can be expressed as the following equation.

$$LD_j = \frac{S_i}{\sum_{k=1}^{n} S_k} \times \max_{0 \leq l \leq m} (1 - \frac{LD_l}{S_i + A_l}) \tag{3.3}$$

Rate of concordance for types of method definitions (TMD). We calculate similarities between code snippets and bytecode from the viewpoint of modifiers, the type of return value, the type of argument, and method definitions. In many cases, method definitions in code snippets include characteristic patterns of a type modifier. Therefore, we can capture these features by using the following equation.

$$TMD_j = \frac{S_i}{\sum_{k=1}^{n} S_k} \times \frac{\max_{0 \leq l \leq m} TMD_l}{S_i} \tag{3.4}$$

Similarity degree of name of method definitions (NMD). Code snippets often have characteristic names of method definitions. We calculate similarities between code snippets and bytecode from the viewpoint of the NMD by using a 3-gram model that can capture features of partial match because developers may modify names of method definitions depending on their tastes.

$$NMD_j = \frac{S_i}{\sum_{k=1}^{n} S_k} \times \frac{\max_{0 \leq l \leq m} NMD_l}{S_i} \tag{3.5}$$

3.4 Experiments in Code Reuse Detection

In this section, we analyze the relevance between apps and code snippets by calculating similarities between them with the method proposed in Section 3.3. In Section 3.4.1, we describe our dataset. In Section 3.4.2, we calculate similarities between vulnerable code snippets and vulnerable apps using our proposed method explained in Section 3.3 to analyze their relationship.

3.4.1 Datasets

Collected Code Snippets. First, we extracted question posts tagged with "Android" from all pages of Stack Overflow. Second, we collected URLs viewed

Table 3.2: List of Evaluated Code Snippets

Detection Name by AndroBugs	Classification Condition	# Targeted Code Snippets
SSL_CN2	getSocketFactory getHostnameVerifier	43
SSL_X509	X509Certificate checkClientTrusted checkServerTrusted getAcceptedIssures	84
WEBVIEW_RCE	addJavascriptInterface	82

Table 3.3: List of Evaluated Apps

Market Name	# of Apps	Price	Operating Structure	Collection Date
Google Play (paid apps)	14,829	Paid	Official	October, 2015
Google Play (free apps)	16,532	Free	Official	October, 2015
Qihoo360 Mobile Assistant	30,549	Free	Third party	May, 2016

L or more times by users. In this paper, L is 1,000. Finally, we obtained 243,589 code snippets crawled from all answer posts on the above-mentioned collected URLs.

We extracted possibly vulnerable code snippets from all collected code snippets by using the classification conditions in Table 3.2. Specifically, we extracted code snippets that contain specific API calls related to the focused vulnerabilities. The number of code snippets used for the further evaluation is shown in Table 3.2. Note that code snippets outside method definitions are not covered in this experiment.

Collected Android Apps and Focused Vulnerabilities. We collected apps from Google Play, the official Android market, and Qihoo 360 Mobile Assistant [30], a third party market. Datasets of Google Play include both paid apps and free apps, but datasets of Qihoo 360 Mobile Assistant include only free apps. The breakdown of these datasets is in Table 3.3. Additionally, Table 3.4 presents the results of vulnerability scanning by AndroBugs.

As was mentioned in Section 3.2.3, there are various kinds of vulnerabilities in Android apps. We focus on three critical vulnerabilities named by the vulnerability scanner AndroBugs: SSL_CN2, SSL_X509, and WEBVIEW_RCE. SSL_CN2 and SSL_X509 are related to insecure communication by Secure Sockets Layer. That is, these vulnerabilities indicate that apps do not verify server certificates, which allows attackers to do man-in-the-middle (MITM) attacks. These vulnerabilities may leak sensitive information such as login credentials. WEBVIEW_RCE is related to `WebView` and Remote Code Execution. It causes apps to allow external JavaScript to control the host application. This critical vulnerability allows attackers to execute arbitrary codes by malicious JavaScript.

3.4.2 Evaluating Performance of Proposed Method

We evaluated the performance of the proposed method using apps created by ourselves using several code snippets. First, we selected five code snippets that are likely to have been reused by developers. Second, we simulated developer's behaviors of copying and pasting code snippets into their apps by actually cre-

Table 3.4: Results of Vulnerability Scanning of Collected Apps by AndroBugs

Market Name	# of Detected Apps / # of All Apps		
	CN2	X509	WEBVIEW
Google Play (Paid Apps)	875/14,829 (5.9%)	1,538/14,829 (10.4%)	3,036/14,829 (20.5%)
Google Play (Free Apps)	2,610/16,532 (15.8%)	3,732/16,532 (22.6%)	7,936/16,532 (48.0%)
Qihoo360 Mobile Assistant	8,569/30,549 (28.1%)	10,274/30,549 (33.6%)	12,102/30,549 (39.6%)
Total	12,054/61,910 (19.5%)	15,544/61,910 (25.1%)	23,074/61,910 (37.3%)

Table 3.5: Results of Performance Verification

Code Snippets	# of Defs	# of Calls	LCS	LD	TMD	NMD	Score
Code_A	7	10	1.000	0.900	0.847	1.000	0.937
Code_B	12	15	0.923	0.892	1.000	1.000	0.954
Code_C	1	20	1.000	0.741	1.000	1.000	0.935
Code_D	2	19	0.947	0.947	0.884	1.000	0.945
Code_E	4	20	1.000	0.928	0.983	0.950	0.965

ating five apps by using the code snippets with IDE autocomplete. Finally, we applied the proposed method to see if the code snippets and the five created apps could be correctly matched. Table 3.5 presents the results of the evaluation. We show that the proposed method ensures high performance in the five code snippets.

3.4.3 Experimental Results

We calculated similarities between the collected code snippets and apps using the proposed method in Section 3.3 with a similarity threshold of 0.8. In Section 3.4.2, we evaluated the performance of the proposed method. From that results, if developers copy and paste code snippets into their apps without modifications, the Score is not less than 0.9. As mentioned above, developers are assumed to modify code snippets and implement their apps using somewhat modified code snippets. Hence, we decided a similarity threshold of 0.8 considering somewhat modified code snippets. Note that we used code snippets with two or more method calls because our method cannot handle such small snippets properly and may produce false matches. We discuss this limitation in Section 3.5.3. Table 3.6 presents the results. We revealed that 31.7% of all evaluated apps that have X509 vulnerabilities contain the possibly vulnerable code snippets from Stack Overflow. Especially, we confirmed that a single problematic snippet has caused 4,844 apps to contain X509 vulnerabilities, which is 31.2% of all collected apps with that vulnerability.

The above results confirm that the possibly vulnerable code snippets indeed have a high chance of being contained in the vulnerable apps. However, the results do not confirm if these snippets are indeed causing the vulnerability of the apps. To clarify that, we analyze the relationship between the snippets and vulnerable apps in more detail.

We selected the “most popular” vulnerable code snippet from each of the three vulnerabilities. That is, for each of the three vulnerabilities, we selected the code

Table 3.6: Correspondence between Possibly Vulnerable Code Snippets and Vulnerable Apps

Market Name	# of Apps with Possibly Vulnerable Snippet / # of All Apps		
	CN2	X509	WEBVIEW
Google Play (Paid apps)	146/875 (16.7%)	234/1,538 (15.2%)	116/3,036 (3.8%)
Google Play (Free apps)	269/2,610 (10.3%)	816/3,732 (21.9%)	278/7,936 (3.5%)
Qihoo360 Mobile Assistant	1,489/8,569 (17.4%)	3,878/10,274 (37.7%)	513/12,102 (4.2%)
Total	1,904/12,054 (15.8%)	4,928/15,544 (31.7%)	907/23,074 (3.8%)

Table 3.7: Ratio of Vulnerable Classes with Potentially Vulnerable Code Snippet to All Matched Classes

Market Name	# of Vulnerable Classes with Snippet / # of All Classes with Snippet		
	CN2	X509	WEBVIEW
Google Play (paid apps)	82/87 (94.3%)	602/605 (99.5%)	14/14 (100%)
Google Play (free apps)	195/200 (97.5%)	1,766/1,767 (99.9%)	195/200 (97.5%)
Qihoo360 Mobile Assistant	1,820/1,903 (95.6%)	7,205/7,212 (99.9%)	139/139 (100%)
Total	1,904/2,190 (95.8%)	9,573/9,584 (99.9%)	348/353 (98.6%)

Table 3.8: Results of Application Update Status Investigation

Type of Application	#Total	#Updated	#Non-updated	#Non-existence
Google Play (paid apps)	487 (100%)	198 (40.7%)	188 (38.6%)	101(20.7%)
Google Play (free apps)	1,287 (100%)	681 (52.9%)	395 (30.7%)	211(16.4%)

Table 3.9: Results of Vulnerability Remediation Status Investigation of Applications with Updates

Type of Application	#Total	#No Vulnerabilities	#Vulnerable
Google Play (paid apps)	198(100%)	118(59.6%)	80(40.4%)
Google Play (free apps)	681(100%)	481(71.6%)	200(29.4%)

snippet that is used most frequently in the corresponding vulnerable apps. Then, we checked if these code snippets are indeed used only in the vulnerable classes in the vulnerable apps identified by AndroBugs. As a result, 95.8% or more vulnerable classes indeed contained the potentially vulnerable code snippets, as shown in Table 3.7. We believe that the above results show that reusing code snippets on Stack Overflow greatly contributes to vulnerabilities of apps.

3.4.4 Investigation of Application Updates

We investigated whether the apps were currently updated and whether the vulnerabilities had been remedied for those apps where we found vulnerabilities derived from code snippet reuse. Then, using the results of the experiment, we

Table 3.10: Classification Results of Developers' Responses to Vulnerabilities

Developer Type	#Devlopers
All apps have been updated and all vulnerabilities have been remedied	102 (42.3%)
App has been updated but not all vulnerabilities have been remedied	34 (14.1%)
Not updating all apps	105 (43.6%)

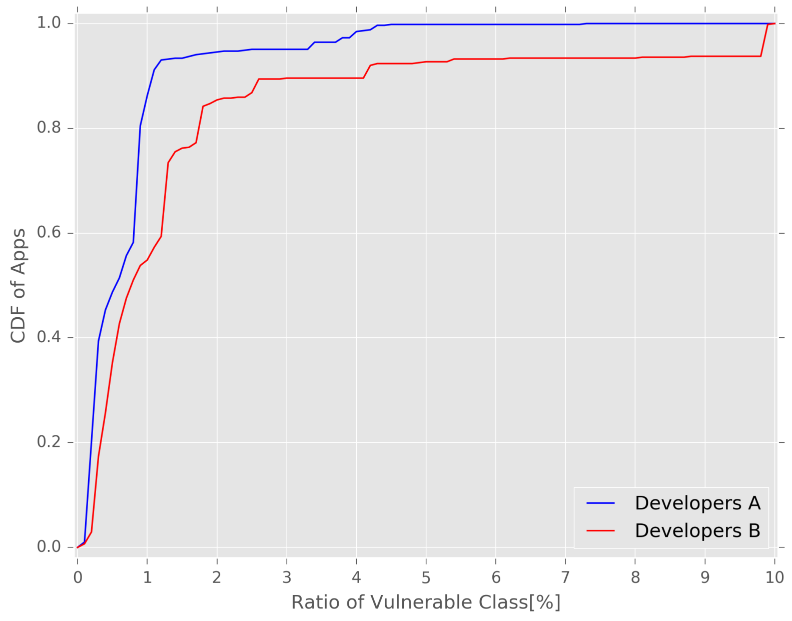


Figure 3.7: Cumulative Percentage of Vulnerable Classes Present

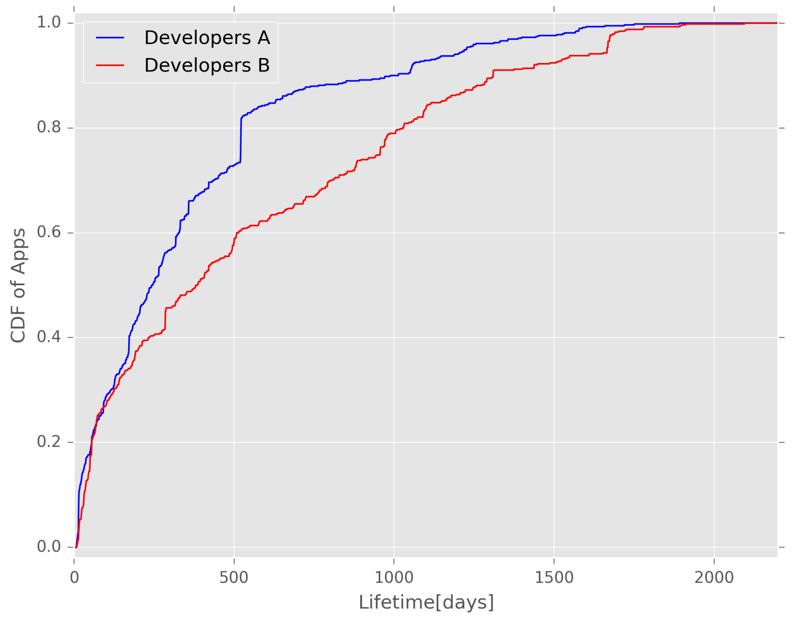


Figure 3.8: Cumulative Percentage of the Current Version of the App's Market Presence

classified the update status of developers of apps in which code snippet-derived vulnerabilities were found for developers.

Investigation of Vulnerable Apps for Remediation Status. We retrieved apps (487 paid apps and 1,287 free apps) with the same package names as the vulnerable apps in the experiments described in Section 3.4 from Google Play using the Google Play Unofficial Python API [31]. For the acquired apps, we investigated whether the relevant classes have been improved using the vulnerability scanning tool AndroBugs [29]. Table 3.8 shows the results of the investigation into whether apps with the same package name as the vulnerable apps have been updated on the current Google Play. We found that 188 paid applications (38.6 % of the total) and 395 free applications (30.7 % of the total) existed in the market without updates. In addition, we retrieved 198 paid apps (40.7 % of the total) and 681 free apps (52.9 % of the total) with updates from the market and investigated the vulnerabilities of the relevant classes, the results of which are shown in Table 3.9. Although many of the vulnerabilities had been remedied, 80 paid apps (40.4 % of apps with updates) and 200 free apps (29.4 % of apps with updates) had the same vulnerabilities. These findings suggest that there are still many apps on the market with code snippet vulnerabilities, and that developers themselves may not be aware of the existence of these vulnerabilities.

Characteristic Analysis by Developer. Using the experimental results of apps and vulnerability update status, we categorized the update status of developers of apps in which code snippet-derived vulnerabilities were found for developers who met the following criteria.

- Group of apps found with code snippet-derived vulnerabilities
- Developers with two or more apps in the above group of apps

The results of the classification are shown in Table 3.10. We confirmed that there are two groups of developers: one group of developers who update all the apps they publish and remedy all vulnerabilities, and the other group of developers who do not update all their apps at all despite the existence of vulnerabilities. The former group is referred to as developer group A and the latter as developer group B.

We randomly retrieved 500 different groups of apps currently published on the market by each group of developers, and plotted the cumulative ratios based on the following two indices in Figure 3.7 and Figure 3.8.

- Percentage of classes that AndroBugs warns are vulnerable relative to classes present in the app
- The period from the time the app was collected to the last update date listed on the market

Both indicators show that a large difference exists between developer group A and developer group B. These results reveal that developer group B tends not to update their apps at all and that the apps they publish are often more vulnerable than those of developer group A.

3.5 Discussion

3.5.1 Countermeasures by Market and Q&A Site Operators

Our experimental results confirmed that reusing code snippets on Stack Overflow strongly increases the likelihood of three vulnerabilities identified by AndroBugs:

SSL_CN2, SSL_X509, and WEBVIEW_RCE. Qihoo360 has a higher rate of vulnerable apps than Google Play, so Google Play can be said to focus more on countermeasures against vulnerabilities. Results also confirmed that Qihoo360 has a higher rate of vulnerable apps including potentially vulnerable code snippets. This may be because more unskilled developers are reusing potentially vulnerable code snippets on Stack Overflow in the Qihoo360 market. Additionally, we found that free apps are more vulnerable than paid ones. We speculate that some developers of free apps are non-professional and less able to eliminate vulnerabilities from their apps.

We found the same code snippets on many different pages, suggesting users of Q&A websites also copy and paste the code snippets. It would be problematic if potentially vulnerable code snippets spread over the sites in this way. To prevent such spreads, Q&A website operators should search for and remove these potentially vulnerable code snippets.

3.5.2 Enhancing Awareness for Developers

Developers should try to check whether code snippets are implemented correctly or not before they reuse them and they should not use code snippets that have vulnerabilities. Software modules can sometimes contain potentially vulnerable codes, and developers may use the modules by mistake without noticing the vulnerabilities.

3.5.3 Limitation

The proposed similarity calculation method has limitations. It is difficult to correlate code snippets with apps if code snippets do not have specific features, e.g., a snippet contains trivial functionalities and everyone will implement it in the same manner regardless of whether they reuse code snippets or not. This problem will also occur if a code snippet is small-scale.

Moreover, code obfuscation can also be problematic for calculating similarities. However, code snippets related to Android API are unaffected by obfuscations, thus we may be able to propose a new method for specifying vulnerable code snippets with obfuscations in our future work.

3.6 Related Work

Android apps vulnerabilities, mobile app developers, code clone detection, and developer community websites have been studied. However, few studies have focused on the effect of real code snippets on Android app vulnerabilities. In this paper, we collected code snippets from Stack Overflow, analyzed a connection between code snippets and vulnerable apps, and investigated vulnerabilities attributable to reusing code snippets.

3.6.1 Vulnerability Analysis

Fahl et al. [32] introduced a tool to detect potential vulnerability against MITM attacks. They analyzed 13,500 popular free apps downloaded from Google Play and revealed that 1,074 (8.0%) of the apps examined contained SSL/TLS code that is potentially vulnerable to MITM attacks.

Egele et al. [33] developed program analysis techniques to automatically check programs on the Google Play and found that 10,327 out of 11,748 apps that use cryptographic APIs - 88% overall - make at least one mistake. They then suggested specific recommendations on the basis of their analysis for improving overall cryptographic security in Android apps.

Furukawa et al. [34] investigate 15,064 of popular Android apps to identify their library version and to reveal statistical distribution of the versions. As a result, several apps even published on Japanese Google Play turn out to be using old version libraries warned their security risks.

3.6.2 Research on Mobile App Developers

Acar et al. [35] contacted developers in Google Play and investigated the resources the developers reference when they create apps, reference frequencies, developers' experience, and so on. Moreover, they performed an experiment in which their research team members develop apps by limiting referenced resources such as Stack Overflow, official documents, and books. Furthermore, they evaluated completed codes from the point of view of functional correctness and security. They also investigated how specific implemented codes are related to security in real apps. By doing these surveys, they analyzed the effects of resources referenced by developers creating apps on security problems. As a result, they clarified not only that Stack Overflow contains more handy resources than official documents do but also that a lot of resources are possible causes of vulnerabilities.

Wang et al. [36] presented a study of the mobile app ecosystem from the perspective of app developers. On the basis of over one million Android apps and 320,000 developers from Google Play, they analyzed the Android app ecosystem from different aspects. Their analysis shows that while over half of the developers have released only one app in the market, many have released hundreds of apps. Then they classified over 320,000 developers into four groups on the basis of the number of apps they released and analyzed the characteristics for different developer groups from the aspects of app quality, development behaviors, and privacy behaviors. The results revealed a wide variation among app developers. In particular, highly active developers, who have created more than 50 apps, tend to release low-quality, less popular, and high privacy risk apps.

3.6.3 Code Clone Detection

Various studies [37, 38, 39, 40] have been reported on code clone detection in Android apps. Hanna et al. [41] investigated present situations of code reuse using similarities between Android apps by focusing on specific operation codes included bytecode. Their results found some applications with confirmed buggy code reuse of Google-provided sample code that lead to serious vulnerabilities in real-world apps, some instances of known malware and variants, and some pirated variants of a popular paid games.

Zhou et al. [42] proposed a module decoupling technique to partition an app's code into primary and non-primary modules and developed a feature fingerprint technique to extract various semantic features. Their investigation shows that piggybacked apps are mainly used to steal ad revenue from the original developers and implant malicious payloads.

3.6.4 Research on Developer Community Website

Liu et al. [43] proposed a unified framework to tackle the challenge of detecting collusive spamming activities of Community Question Answering, which provides rich sources of information on a variety of topics. They also proposed a combined factor graph model to detect deceptive Q&As simultaneously by combining two independent factor graphs. Using a large-scale practical dataset, they found that their proposed framework can detect deceptive contents at an early stage and outperform a number of competitive baselines.

Fischer et al. [44] crawled Stack Overflow for code snippets and evaluated their security score using a stochastic gradient descent classifier. As a result, they revealed that 15.4% of the 1.3 million Android apps they analyzed contained security-related code snippets from Stack Overflow. Out of these, 97.9% contained at least one insecure code snippet.

3.7 Conclusion

We analyzed the relationship between Android apps vulnerabilities and code snippets on a Q&A website. As a result, we showed that a single problematic snippet has caused 4,844 apps to contain a vulnerability, which is 31.2% of all collected apps with that vulnerability. Moreover, we found the same potentially vulnerable code snippets are on many different pages and should be removed by the site operators.

In the future, we will try to comprehensively investigate code snippets that are possible causes of vulnerabilities besides the Q&A websites. Additionally, we will also analyze the mechanism of vulnerable app development using potentially vulnerable code snippets.

Chapter 4

Exploring Event-synced Navigation Attacks across User-generated Content Platforms in the Wild

4.1 Introduction

With the advent of user-generated content (UGC) platforms, people are using the search functions of such platforms as well as conventional search engines to search for information. UGC includes videos, blogs, bulletin board posts, images, and music that can be shared. The platforms that distribute UGC play an increasingly vital role in web communication among users, DataReportal estimated that there would be approximately 3.8 billion users worldwide as of 2020 [45].

Whereas users previously relied on conventional search engines to find information using keywords related to their interests, now they often obtain information and access links through UGC. Typically, a UGC platform provides users with ways to easily access interesting information from the large amount of UGC generated daily, e.g., trend searches and tagging as well as keyword searches. A UGC platform also attracts attackers who distribute malicious links that direct users to external malicious websites. This is a type of web-based social engineering (SE) attack that exploits user's psychological vulnerabilities. Studies have reported that users are more likely to click on links in UGC than in emails because users feel connected to each other with a sense of trust and camaraderie, and that 30%-66% of the subjects were tricked into clicking on links in tool-generated malicious posts that target specific users [46]. A web-based SE attack leverages various methods to navigate users to malicious websites to accomplish an attacker's objectives. Such methods include malware download attacks that mimic legitimate software to be installed [47], survey scams that mimic questionnaires and require users to input personal information [48], and telephone scams that claim to be from legitimate support centers [49]. Among web-based SE attacks, those that piggyback on real-life events are increasing. To increase the chances of malicious links being seen by users, attackers take advantage of extensive user interest to distribute related UGC with malicious links. It has been reported that numerous Apple-related phishing websites tend to appear around the times Apple announces new products [50], and many fake websites were discovered during the 2019 Rugby World Cup [51].

In this paper, we focus on *event-synced navigation attacks*, a type of web-based SE attack that generates UGC with links to an external malicious website and is distributed synced with a real-life event that attracts user attention at a specific time. Event-synced navigation attacks have been deployed across multiple UGC platforms; however, existing research [52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66] only targets a single UGC platform. Thus, a full picture of these attacks has not yet been presented. Therefore, we propose a system to detect event-synced navigation attacks in real time to comprehensively understand the attacks in the wild. The proposed system leverages three intrinsic properties of attacks; (1) UGC are posted at specific times corresponding to real-life events, (2) UGC are posted across multiple UGC platforms and (3) UGC are posted with links to an external malicious website. These properties are inevitable footprints left by attackers to attract user attention and lead users to a malicious website. Finally, we detected a significant amount of malicious UGC in the wild using the proposed system and investigated malicious entities involved in the attacks. We found many cases where the same attacker spread attacks across multiple platforms. Especially, those attacks originating from multiple platforms were accessed more than four times as frequently as attacks originating from a single platform. We also found that 87.8% of the fully qualified domain names (FQDN) survived for more than 100 days and that, one week after being posted, 69.0% of malicious UGC had not been removed by UGC platform providers.

Our primary contributions are summarized as follows.

- We defined a new threat, i.e., event-synced navigation attacks, that can affect many users across multiple UGC platforms in relation to real-life events.
- We proposed an innovative system that detected event-synced navigation attacks in real time with 97% accuracy.
- Using the proposed detection system, we found 39,646 malicious UGC posted over a period of 40 days.

This paper is an extended version of a paper presented at IEEE COMPSAC 2021 [67]. The paper [67] proposes a new system for early detection of event-synced navigation attacks, and evaluates its ability to detect attacks with high accuracy against test data. However, we did not conduct a detailed analysis of the malicious content detected in the wild, nor did we evaluate the current countermeasure techniques of each UGC platform providers. In this study, we analyzed the detected event-synced navigation attacks from five new perspectives (Section 4.5). The new contributions of this paper are as follows.

- We found that 34.1% of event-synced navigation attacks are spread across two or more popular UGC platforms.
- We revealed UGC platform providers often miss malicious UGC and that attackers abuse the context of real-world events, particularly sporting events to which people pay attention, especially sporting events which attract significant user attention.
- We confirmed a number of cases where event-synced navigation attacks have led users from popular social platforms to malicious sites (i.e., information theft, survey scams, suspicious browser plugin installations, etc.), as identified in a number of existing studies.

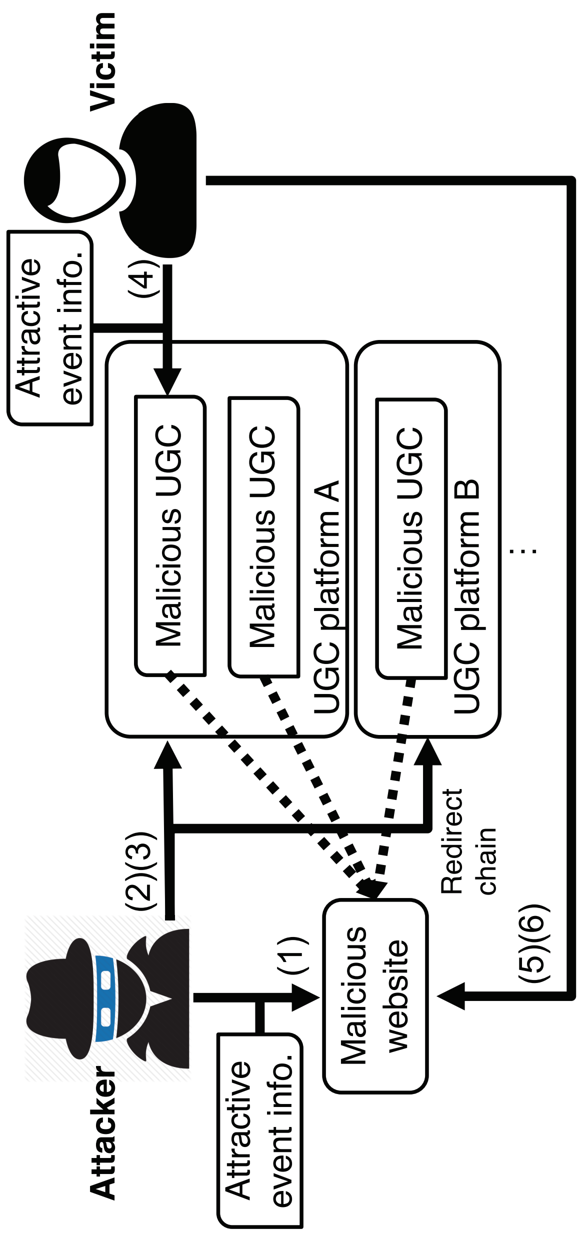


Figure 4.1: Overview of Threat Model

4.2 Guiding Users through UGC

4.2.1 Type of Guidance

UGC is used as a medium to direct users to external sites or different platforms in a variety of contexts.

Link Sharing. Users post links to their favorite websites and articles on UGC platforms, directing other users to those links.

Online Review. Users post reviews of products and services to encourage other users to visit their websites.

Contests and Campaigns. Companies conduct photo contests or hashtag

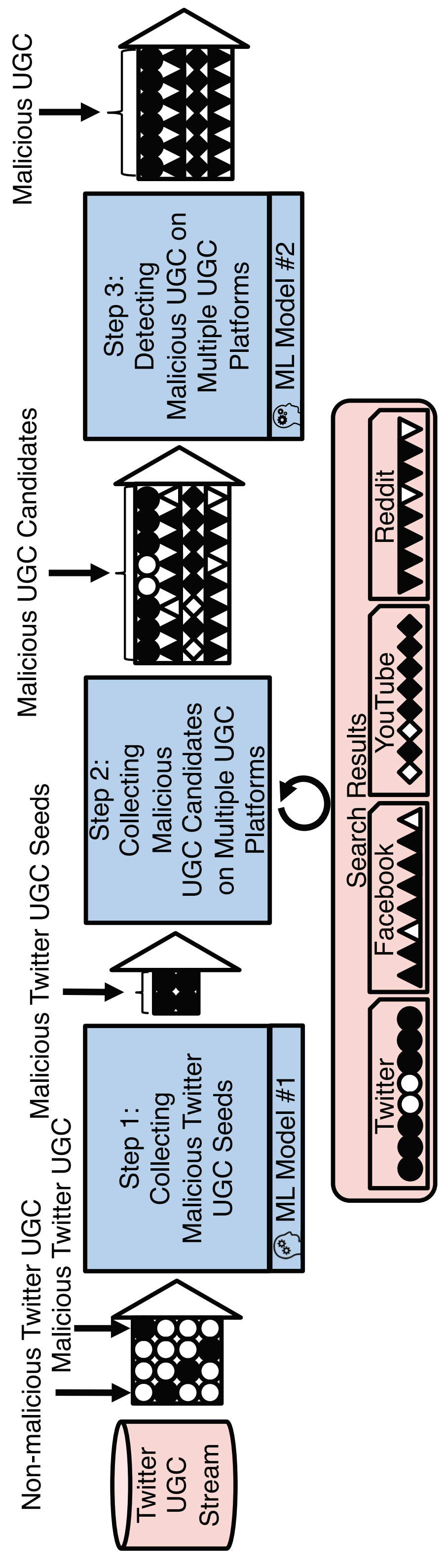

Twitter UGC Stream
Non-malicious Twitter UGC
Malicious Twitter UGC
Step 1: Collecting Malicious Twitter UGC Seeds
ML Model #1
Malicious Twitter UGC Seeds
Step 2: Collecting Malicious UGC Candidates on Multiple UGC Platforms
Malicious UGC Candidates
Step 3: Detecting Malicious UGC on Multiple UGC Platforms
ML Model #2
Malicious UGC
Search Results
Twitter
Facebook
YouTube
Reddit

Figure 4.2: System Overview

campaigns on UGC platforms to direct participants and viewers to their websites or specific platforms.
Celebrity Marketing. Celebrities recommend specific products on their own accounts on UGC platforms and direct followers to product sales sites.
Forums and Q&A sites. Users provide information to solve specific problems

and direct other users to the site by sharing links to their sources and related information.

The activity of directing users from well-known UGC platforms to other platforms and websites in the aforementioned contexts raises security threats. For example, user-generated links and files may contain malware, either intentionally or unintentionally. It could also lead to phishing scams in which users provide personal information believing it comes from a reliable source. In this study, we target these types of malicious activities that lead users away from well-known UGC platforms for detection and analysis.

4.2.2 Threat Model

Our threat model for an event-synced navigation attack is shown in Figure 4.1. An attacker posts content that includes keywords related to an exciting real-life event that attracts a large number of users/victims who use UGC during the event. The posts contains links to a prepared malicious website. For example, during matches in the 2019 Rugby World Cup matches, users searched UGC platforms for keywords related to the tournament. The key to a successful attack is to distribute malicious UGC posts at the best time to attract the users' attention and lead a large number of users to the malicious website. In this study, we focus on attacks that target pre-scheduled events except for sudden incidents or scandals as a context (e.g., celebrity scandals, earthquakes, and riots) because the context and method of spreading differ significantly. We then describe the attacker and victim's perspectives in our threat model.

4.2.3 Attacker's Perspective

The attacker executes the attack in three steps as follows.
Preparing Malicious Websites. The attacker selects an event that can attract a large number of users (1). For example, the attacker can select an upcoming sporting event or a product launch. Then, the attacker constructs an attack scenario based on the event to target the users' psychological vulnerabilities. Take the Rugby World Cup as an example. In this case, the attacker may create a phishing website that mimics an official streaming website with a form asking for credit card and address details.
Creating Malicious UGC. The attacker creates UGC in a context that makes the user want to access the prepared malicious website (2). The URL in the UGC is designed to reach the malicious website via multiple websites. Note that attackers often use URL shortening services or redirects to intermediate websites where the destination is unknown until users actually access the website. This method is employed to evade detection by malicious URL detection algorithms deployed on each UGC platform.
Distributing Malicious UGC. The attacker disguises themselves as a legitimate user and spreads the UGC on multiple UGC platforms with a large number of users (3). As mentioned previously, the attacker distributes the trap when the user is expected to search for keywords related to the event using the search functions on the UGC platforms. For example, malicious UGC spreads on UGC platforms like Twitter and Facebook when users are most interested in the targeted events, e.g., immediately after the release of a new iPhone or the start of the Rugby World Cup.

4.2.4 Victim's Perspective

A user encounters the attack in three steps as follows.
Searching for Event Information. A user searches for relevant keywords on UGC platforms based on their interests (4). For example, if the user wants to watch a live Rugby World Cup match on the web, they may input the keywords "rugby world cup free live streaming." In addition, trending words, e.g., "rugby world cup 2019", and related hashtags e.g., `#RWC2019`, may also be used in search terms. The search results will present both non-malicious content and malicious UGC posted by the attacker. Previous studies [68, 69] have stated the results of search engines (e.g., Google, Bing and Baidu), are polluted by black hat search engine optimization techniques, and the results link to malicious websites. UGC platform users frequently search for UGC in chronological order to obtain the most up-to-date information; therefore, an attacker's posts may appear in large numbers depending on the timing.
Reaching Malicious Websites. When the user clicks a URL in malicious UGC spread by the attacker, they are redirected to the prepared malicious website (5). Note that some malicious websites contain a large number of redirects, or go through different UGC on the same platform or UGC on different platforms.
Becoming a Victim of an Attack. The user may be asked for their personal information or to install fake software to gain access to the target website, which is actually a fake and malicious website (6). Once the user enters their personal information (e.g., email address, password, credit card details, etc.), the entered information is sent to the attacker; however, the user cannot access the desired content.

4.3 Proposed System

In this paper, we propose a system to identify malicious UGC spread by event-synced navigation attacks in real time. Figure 4.2 shows an overview of the proposed system. Given a Twitter UGC stream as input, the proposed system automatically identifies malicious UGC spread on multiple UGC platforms. The malicious UGC identified by the proposed system include those that lead to malware download websites, phishing websites, and fake websites. The proposed system involves three steps: (1) collecting malicious UGC seeds on Twitter, (2) collecting malicious UGC candidates on multiple UGC platforms, and (3) detecting malicious UGC on multiple UGC platforms. The following sections describe these steps in detail.

4.3.1 Step 1: Collecting Malicious Twitter UGC Seeds

The first step takes a Twitter UGC stream as input. We selected Twitter as the first input to the proposed system because it is better suited than any other UGC platforms in terms of generating information to search for similar malicious UGC (i.e., malicious UGC seeds) due to its ability to retrieve large amounts of data in real time. Then, we employed a supervised machine learning approach to identify malicious Twitter UGC seeds and output them as a starting point for subsequent detection of malicious UGC on multiple platforms. This step comprises three sub-steps: *extracting features*, *labeling*, and *applying machine learning*.

Table 4.1: List of Features

Type	Model	Features	Dimensions
UGC Context Features	ML Model #1 & #2	UGC Word Embedding	700
UGC Timing Features	ML Model #1	# of Selected UGC	1
		Average Time Interval	1
		Average # of URLs per Post	1
		Unique # of URLs	1
		Unique # of Users	1
Infrastructure Features	ML Model #2	# of Selected UGC	1
		# of Distributed Platforms	1
		Average # of URLs per Post	1
		Unique # of Landing URLs	1
		Unique # of Users	1
Web Content Features	ML Model #2	HTML Word Embedding	700
		HTML Tag Counts	30
		# of Redirects	1

Step 1.1: Extracting Features

In this sub-step, features that contribute to the identification of malicious Twitter UGC are extracted from the Twitter UGC stream. Twitter is an extremely high-volume UGC stream; thus, designed features can quickly identify malicious Twitter UGC. To realize this, we design the features under the limitation that we do not need to access the URLs contained in the Twitter UGC, which allows us to identify malicious Twitter UGC quickly after an attacker posts it. Here, we utilize two features (Table 4.1), i.e., *UGC context features* and *UGC timing features*.

UGC Context Features. We generated distinguishable features for the characteristic text in malicious Twitter UGC (e.g., the context in which the attacker attempts to direct users to a malicious website). Specifically, we created 700-dimensional features corresponding to the text from text portions contained in each input UGC using word embedding. Word embedding is a feature learning technique in natural language processing that is used to extract fixed-dimensional feature vectors by converting words and phrases in the lexicon to real vectors. Among the many existing word embedding algorithms, Sent2Vec [70] was used in this study because it has been reported to perform the best for relatively short sentences used in UGC platforms, which are the target of this study. The various parameters were set to the same values as in the experiments in the previous study [70], and the dimensions of the feature vectors were set to 700 dimensions as well.

UGC Timing Features. We generated UGC timing features to capture the characteristics of malicious Twitter UGC that is widely spread at a specific time. These features allow us to capture the unique concurrences of malicious Twitter UGC caused by events not captured in only the textual context considered by the UGC context features. Here, we first extract multiple UGC posted within the last 60 minutes of the input UGC from the Twitter UGC stream. The threshold was set to 60 minutes because we considered that if attackers target the beginning of a sports festival or a new product launch, similar posts will cluster within 60 minutes. Then, we selected only the UGC that is similar to the text of the input UGC from the extracted UGC. Here, we calculated the degree of similarity between the input UGC and the extracted UGC using the Jaccard coefficient, and if the calculated similarity was greater than or equal to a predetermined threshold, it was considered similar. In this study, the threshold for the Jaccard coefficient was set to 0.7, which is similar to that used in a previous study [71]. The selected UGC is the one posted at similar times that has text that is similar to the input UGC. Then, we generated five features from the selected UGC (Table 4.1). The first feature is the number of UGC in the selected UGC, which captures the characteristics of similar malicious Twitter UGC itself spread at specific times. The second feature is the average posting time interval of the selected UGC. The same attacker or an attacker with the same target can post many malicious UGC simultaneously; thus, the shorter the average posting time interval, the more characteristics of malicious Twitter UGC can be captured. The third feature is the average number of URLs contained in each UGC, as selected by the Jaccard coefficient described above. Attackers include many URLs in UGC to tempt users to click on a URL [52], and this feature can capture that characteristic. The fourth feature is the unique number of all URLs in the selected UGC. Attackers use URL shortening services and intermediate websites to evade URL-based blocklist detection, which leads users to the same malicious website with a different URL [53, 72]. In other words, the larger this feature is, the more likely it is to be malicious Twitter UGC. The fifth feature is the number of unique users who have posted their selected UGC. It has been reported that attackers post simultaneous malicious Twitter UGC on many accounts to avoid detection on the platform [52]. Similarly, the larger this feature is, the more likely it is to be malicious Twitter UGC.

Step 1.2: Labeling

In the labeling sub-step, ground truth labels are assigned to the input UGC to be used as training data for supervised machine learning. Here, the ground truth label is binary: malicious Twitter UGC and non-malicious Twitter UGC. Generally, a large amount of training data with ground truth labels is essential to the success of supervised machine learning. Therefore, we design a method to obtain the ground truth labels for many unlabeled UGC without manual labeling. Here, we first select the UGC that contains URLs that lead to websites outside of Twitter from the input UGC. To label malicious Twitter UGC, we adopted the settings from a previous study [53] and our own settings to capture trends. We labeled UGC that satisfies the following criteria as malicious Twitter UGC: (1) the UGC deleted by Twitter after a certain period from the posted date or posted by accounts suspended by Twitter, and (2) UGC that includes at least one keyword that is listed in Google Trends [73] or Twitter Trends [74] at the same timing as the post. We labeled the UGC as non-malicious if it had text similar to that contained in the UGC previously labeled as malicious and if it contained unpopular URLs. UGC with popular URLs are not considered in this study since they can be easily determined to be non-malicious by the URL list. UGC that have similar contexts to malicious UGC but have not been removed by the respective UGC platform providers are considered non-malicious in this study. The similarity metric of UGC text is based on the Jaccard coefficient. In this study, popular websites were considered those in the top 10,000 of the most recent Alexa top sites [75] at that time. Note that we did not simply define non-malicious Twitter UGC as that containing popular URLs. Our proposal is to accurately detect both malicious Twitter UGC and similar but non-malicious Twitter UGC when subsequently creating machine learning models. For example, if we label Twitter UGC as non-malicious simply because it contains the URLs of popular websites, we will detect UGC containing unpopular URLs as malicious Twitter UGC. Thus, we would not be able to detect malicious Twitter UGC involved in event-synced navigation attacks.

Step 1.3: Applying Machine Learning

In this sub-step, the model is trained using the training data, and the trained model is used to identify malicious Twitter UGC seeds from a Twitter UGC stream. Here, we first constructed a supervised machine learning model (ML Model #1, in Figure 4.2) using training data comprising the extracted features and the corresponding ground truth labels. Supervised ML includes various algorithms (e.g., random forest, decision tree, support vector machine, logistic regression, and naïve Bayes). We performed preliminary evaluation and selected random forest, which exhibited the highest and most stable accuracy on the training data. Next, we used the constructed training model to predict the binary classification of malicious Twitter UGC or non-malicious Twitter UGC for each input UGC obtained from the real-time Twitter UGC stream. The identified malicious Twitter UGC was used as the malicious Twitter UGC seeds in the following steps.

4.3.2 Step 2: Collecting Malicious UGC Candidates on Multiple UGC Platforms

In the second step, the malicious Twitter UGC seeds (obtained in Step 1) are taken as a starting point, and similar malicious UGC candidates are collected from multiple UGC platforms (Twitter, Facebook, YouTube, and Reddit). We targeted these four UGC platforms because they have a large number of users [45] and are likely to be targeted by attackers. This step involves two sub-steps, i.e., *generating search queries* and *collecting malicious UGC candidates*.

Step 2.1: Generating Search Queries

This sub-step generates a search query to find malicious UGC candidates that are similar to malicious Twitter UGC seeds in the four UGC platforms. For example, consider the malicious Twitter UGC text "Rugby World Cup Japan vs Russia Free Live Streaming Click This Link [URL]." Here, to search for similar malicious UGC candidates from the UGC platforms, we must generate a set of words extracted from the malicious Twitter UGC, e.g., "Japan vs Russia" and "Rugby World Cup" as a search query. This is required because, in our event-synced navigation attack, multiple attackers create multiple malicious UGC independently for an event on multiple UGC platforms: thus, they are not covered by exact match searches with the known malicious Twitter UGC. In addition, the search algorithm differs greatly depending on the UGC platform. For example, some UGC platforms, e.g., Twitter, display all UGC results that match even if the query is short, while others, e.g., YouTube, calculate the degree of similarity on the basis of the search query using their own algorithm and only display results with high similarity. Thus, we design a search query generation method to collect malicious UGC candidates efficiently considering each UGC platform's search algorithm. Here, two methods were employed to generate queries from different perspectives, i.e., *context-based query generation*, which generates a query based on the context of the words, and *frequency-based query generation*, which generates a query based on the frequency of occurrence of words. Then, in *query selection*, we select the most suitable query to collect malicious UGC candidates on each UGC platform using the collection results of known malicious UGC on each UGC platform.

Context-based Query Generation. We leverage text summarization techniques in natural language processing to capture contextual features, e.g., “live streaming” and “free giveaways,” which commonly appear in malicious UGC text. Text summarization techniques are used to automatically select a set of words suitable to summarize a given text. Although various text summarization algorithms have been proposed, we generated a contextual search query using the EmbedRank algorithm [76], which is a fast algorithm that can detect event-synced navigation attacks early. EmbedRank automatically extracts a set of words that represent the characteristics of a sentence from the target sentence without supervised data. EmbedRank comprises three steps, i.e., key phrase candidates are generated according to certain rules, key phrase candidate and the target sentence are converted to corresponding vectors using the same embedding model, and the cosine similarity between the key phrase candidate and the target sentence is calculated to rank key phrase candidates. In our case, we first generate key phrase candidates based on the word-by-word 2-gram (e.g., Live Streaming), 3-gram (e.g., Japan vs Russia), and 4-gram (e.g., Streaming Click This Link) from the collected malicious Twitter UGC seeds (Step 1). Then, we convert the key phrase candidates and original malicious UGC sentences into 700-dimensional feature vectors using the word embedding model (Step 1.1). Finally, we compute the cosine similarity between the feature vectors and rank the key phrase candidates. From the ranking results, we generate search queries that are greater than a predetermined rank such that we obtain search queries specific to the malicious UGC sentences.
Frequency-based Query Generation. Keywords like “Rugby World Cup” and “iPhone release” appear frequently in malicious UGC in relation to the event-synced navigation attack. We employ the Z-score method [77] to capture the increase in occurrence frequency of these words as an anomaly. The Z-score is the number of standard deviations by which the value of a raw score (i.e., an observed value or data point) is greater or less than the mean value of what is being observed or measured. The Z-score is defined as an observed value minus the mean value divided by the standard deviation. This makes it a robust anomaly detection method as this group of data has a mean of 0 and a standard deviation of 1, thereby enabling the comparison of values with different units. As a result, anomalies can be identified with nearly the same accuracy even when the flow rate of UGC per unit time increases or decreases significantly. Here, we first generate candidate search queries based on word-by-word 2-gram, 3-gram, and 4-gram from the malicious Twitter UGC seeds collected in Step 1 as in the case of the *context-based query generation*. Then, the Z-score of the search query candidate was calculated, and if its value exceeds a threshold, it was generated as a search query.
Query Selection. We select a search query for each UGC platform from the search queries generated by the two methods described previously. We design a preferential selection in consideration of the search queries identified as malicious UGC on each UGC platform in the past. We compute word malicious score on a word-by-word basis for each UGC platform using the words contained in the known malicious UGC search queries, and we finally compute query maliciousness score for the overall query. Here, all scores have a range of 0.0–1.0. If a word does not exist in a past search query, it is set to a new word with a word maliciousness score of 1.0.

The word maliciousness and query maliciousness scores are calculated as follows. First, we calculate word maliciousness score for each UGC platform using the UGC collected in the past for each word. For example, consider the situation where we determine whether to perform a search with the query "rugby streaming japan" on Twitter. Here, assume we have only "rugby world cup streaming," "free live streaming," and "rugby free live," for Twitter's past search queries containing each word (rugby, streaming, and japan) in the search query at that time. Their past search queries and corresponding results are summarized as follows. Among the UGC collected by the query "rugby world cup streaming" on Twitter, 20 were malicious UGC and 50 were non-malicious UGC. Among the UGC collected by the query "free live streaming" on Twitter, 100 were malicious UGC and 100 were non-malicious UGC. Among the UGC collected by the query "rugby free live" on Twitter, 10 were malicious UGC and 100 were non-malicious UGC. There were no previous search queries and corresponding results containing "japan" on Twitter. In this case, the query maliciousness score for "rugby streaming Japan" is calculated as follows. First, the word maliciousness score is $\frac{\frac{20}{20+50}+\frac{10}{10+100}}{2} = 0.19$ for "rugby" and $\frac{\frac{20}{20+50}+\frac{100}{100+100}}{2} = 0.39$ for "streaming" , and 1.0 for "japan" because it is a new word. Second, we calculate the query maliciousness scores of the candidate search queries using the average of the calculated word maliciousness score for each word. Third, the query maliciousness score of the search query candidate "rugby streaming japan" on Twitter is $\frac{0.19+0.39+1.00}{3} = 0.53$. Then, we calculate the word maliciousness score for each word on Facebook, YouTube, and Reddit as well as the query maliciousness score for the search query candidates. Finally, we select search queries up to the threshold value for each UGC platform in order from the highest to lowest query maliciousness scores of the candidate queries. For ethical consideration (Section 4.6.2), we limited the number of search queries to a maximum of 100 on each platform per *query selection.*

Step 2.2: Collecting Malicious UGC Candidates

This sub-step collects malicious UGC candidates from the four UGC platforms using the generated search query. Here, we collect UGC from the four UGC platforms as follows. Note that we exclude posts without URLs to external websites for all services because such posts are beyond the scope of this study.
Twitter. With Twitter, we search for the most recent UGC by combining the URLs and queries. We then retrieve information from the search results, including the sentences contained in the UGC, user names, and posting times.
Facebook. On Facebook, we search for three types of UGC, i.e., posts, videos, and events, by combining the URLs and queries. First, for posts, we obtain the sentences, user names, and post times from the top 20 results in chronological order. Then, for videos, we obtain the video title, summary, user name, and time of posting from the top 20 results in chronological order. Here, the text of the UGC is the combined string of the video title and summary. Finally, for events, we obtain the event title, event details, and user names from the top 20 results in chronological order. Here, the text of the UGC is a string combining the event title and event details.
YouTube. On YouTube, we search for two types of UGC, i.e., video posting and video distribution, by combining the URLs and queries. We then obtain the URLs of the videos from the top 20 results in chronological order. In addition, we access each URL to obtain the video title, summary, and user name. Here, the text of the UGC is the combined string of the video title and summary.
Reddit. On Reddit, we search for one type of UGC in a thread by combining the URLs and queries. Then, we obtain the URLs of the threads from the top 20 results in chronological order. In addition, we access each URL and obtain the

thread title, content, user name for posting, and posting time. Here, the text of the UGC is the combined string of the thread title and content.

4.3.3 Step 3: Detecting Malicious UGC on Multiple UGC Platforms

The third step takes malicious UGC candidates collected from each UGC platform as input. Then, supervised ML is employed to identify malicious UGC on the four UGC platforms as the output. Note that Step 1 only targets malicious Twitter UGC seeds, whereas Step 3 collects malicious UGC on all four UGC platforms. This step comprises three sub-steps, i.e., *extracting features*, *labeling*, and *applying ML*.

Step 3.1: Extracting Features

In this sub-step, features that contribute to identifying malicious UGC on the four UGC platforms are extracted. We design features from multiple UGC platforms that can identify malicious UGC quickly and are not dependent on a specific platform. The UGC features for multiple UGC platforms, i.e., UGC context features, infrastructure features, and web content features (Table 4.1), can be classified as malicious or non-malicious UGC using a single training model.
UGC Context Features. We generated distinguishable features for the characteristic text contained in common malicious UGC. As discussed in Section 4.3.1, we created a 700-dimensional feature corresponding to the text from the text portions contained in the UGC on each of the input platforms using Sent2Vec in word embedding. Note that the features have the same number of dimensions even if the platforms differ.
Infrastructure Features. We created infrastructure features to capture the infrastructure, e.g., diverting accounts on UGC platforms for attackers to spread links and eventually reach malicious websites. These features allow us to capture resources that cannot be captured by textual context alone. Here, we first access the URLs contained in the input UGC using a web crawler. Then, we obtain the IP address of the finally reached website. We extract multiple UGC from the input UGC with the same IP address as the finally reached website. The UGC is posted by the same user or attacker even though there may be UGC on different platforms or with different usernames. Then, we create the five features from the UGC (Table 4.1). The first feature is the number of unique UGC in the extracted UGC. This feature can capture the characteristics of attacks that are spread at a specific time that reuse the same hosting server or domain name, which are spread at a specific time. The second feature is the number of platforms on which the extracted UGC was posted. We can capture the characteristic of an attacker who posts to multiple platforms to lead a large number of users to a malicious website. The third feature is the average number of URLs listed in each UGC among the extracted UGC. As discussed in Section 4.3.1, here, we can capture the characteristic that attackers include many URLs in UGC to make users click on the URLs. The fourth feature is the number of unique URLs in the extracted multiple UGC. As discussed in Section 4.3.1, the attacker uses different URLs to evade URL-based blocklists. Therefore, the larger the feature, the more likely it is to be a more malicious UGC. The fifth feature is the number of unique users in the extracted multiple UGC. Here, we can capture the characteristic of an attacker creating multiple accounts for each platform to post malicious UGC simultaneously.
Web Content Features. We generate web content features to capture the

characteristics of malicious websites commonly seen in attacks, and we classify these features as malicious or non-malicious. Here, we first access the URL in the malicious UGC candidates using a web crawler. We then obtain the web content of the final destination website by handling redirections. Then, we create the three features shown in Table 4.1 based on the obtained information. The first feature is to generate text-identifiable features to trigger information input and button clicks on a website. Here, we use a word embedding technique to generate 700-dimensional features corresponding to the text, excluding HTML tags, from the obtained web content. To capture the characteristic text, we employ Sent2Vec [70] in the word embedding (Section 4.3.1) because websites that trick users by disguising themselves as legitimate websites often contain similar text [78], and many such websites also contain text that is similar to malicious UGC. The second feature is the number of the top 30 frequently appearing HTML tags in malicious websites, which were identified in a preliminary investigation. The attacker simultaneously creates many similar malicious websites; thus, we can capture the characteristic that malicious websites have similar HTML structures. The third feature is the number of redirects that occurred prior to reaching the final destination website. It has been reported that attackers attempt to evade detection by platform providers using URL shortening services and various redirection paths [53]. The larger this feature, the UGC is more likely to be classified as malicious.

Step 3.2: Labeling

Similar to Section 4.3.1, this sub-step assigns ground truth labels to malicious UGC candidates to be used as training data for the supervised ML. Here, the ground truth label is binary, i.e., malicious or non-malicious. Note that we can obtain the ground truth labels semi-automatically without manually labeling everything. To label malicious UGC on each platform, we essentially labeled them in the same manner described in Section 4.3.1. In this sub-step, the labeling covers multiple UGC platforms; thus, each platform has different criteria to remove malicious UGC. Therefore, we expand the training data, especially malicious UGC, by taking advantage of the multiple platforms. We obtain the IP address derived from accessing a URL contained in UGC labeled as malicious UGC on one of the platforms and then we label UGC that contains a URL with the same IP address as malicious UGC (even if it is from a different platform). As a result, we can efficiently label the malicious UGC of the same attacker on diffrent platforms. To label non-malicious UGC on each platform, we apply the approach described in Section 4.3.1 to multiple platforms. From the remaining UGC not labeled as malicious, we label the UGC as non-malicious UGC if the URLs are not included in the popular websites. The non-malicious UGC is collected by a search query due to its similarity in context to malicious UGC; however, here, it is assumed to be non-malicious. For example, the events abused by the attackers have official websites, related websites, and news websites.

Step 3.3: Applying Machine Learning

In this sub-step, the model is trained using the training data, and the trained model is used to identify malicious UGC on multiple UGC platforms from the malicious UGC candidates. Differing from the process described in Section 4.3.1, here, we construct a model that can handle the UGC of multiple platforms as the input data for classification. Even if the posted UGC platform differs, we only

Table 4.2: Datasets for Evaluation

Steps to use / Collected Time	Dataset Name	# UGC
Twitter Stream Dataset	All Twitter	63,442,349
Step 1	Malicious Twitter	73,146
09/15/2019 – 10/14/2019	Non-malicious Twitter	165,813
Search Result Dataset	Malicious Twitter	4,343
Step 2 and Step 3	Non-malicious Twitter	1,386
10/22/2019 – 10/23/2019	Malicious Facebook	2,002
	Non-malicious Facebook	485
	Malicious YouTube	517
	Non-malicious YouTube	369
	Malicious Reddit	1,723
	Non-malicious Reddit	1,163

require a single trained model because the designed features are fixed-dimensional with the same criterion that can be used with different UGC platforms. We also construct a supervised machine learning model (ML Model #2, Figure 4.2) with random forest (Section 4.3.1). We use the training model to predict the binary classification (malicious or non-malicious UGC) for each input UGC obtained from the search results on the four UGC platforms. We then output the identified malicious UGC on multiple platforms as the final output.

4.4 Evaluation

We evaluated the performance of each step of the proposed system. Here, we first describe the dataset used in the evaluation. We then describe the results of each step of the evaluation.

4.4.1 Datasets

We prepared the UGC Dataset with the ground truth labels required to evaluate the performance of the proposed system.
Twitter Stream Dataset. We collected 63 million UGC with external URLs in English from Twitter for the period from 09/15/2019 to 10/14/2019 (30 days), which we refer to as *All Twitter*. Then, we evaluated the collected UGC again on 10/21/2019 using the two criteria described in Section 4.3.1. We found that 73,164 UGC were removed by Twitter and included at least one trending word from Google Trends or Twitter Trends for the collection period, and we labeled them *Malicious Twitter*. We then labeled 165,813 UGC from the remaining UGC as *Non-malicious Twitter* in the same manner described in Section 4.3.1.
Search Result Dataset. We split the UGC string of Malicious Twitter in the *Twitter Stream Dataset* using line breaks, and we obtained 136,243 different word combinations. Then, we randomly selected 10,000 search queries from 136,243 different strings to search for similar UGC. Here, we collected UGC from Twitter, Facebook, YouTube, and Reddit covering the period from 10/22/2019 to 10/23/2019 (2 days) using the 10,000 selected queries. As shown in Table 4.2, we collected 5,729 Twitter UGC, 2,487 YouTube UGC, 886 Facebook UGC, and 2,886 Reddit UGC. We assigned ground truth labels to the collected UGC. First, we checked on 10/30/2019 to determine whether the collected UGC was removed on each UGC platform, resulting in 432 Twitter UGC, 214 Facebook UGC, 405 YouTube UGC, and 312 Reddit UGC being labeled as malicious UGC. We manually clicked on the URLs in the UGC to confirm that the URLs were related to the SE attack rather than misinformation, pornography or cyberbullying. Second,

Table 4.3: Evaluation

Method		TPR	TNR	Precision
Step 1 of Proposed System (C+T)		**0.969**	**0.989**	**0.975**
Step 1 of Proposed System (C)		0.940	0.955	0.904
Step 1 of Proposed System (T)		0.905	0.931	0.967
Existing System 1 [52]		0.879	0.941	0.874
Existing System 2 [53]		0.712	0.930	0.889
Method			Coverage	Toxicity
Step 2 of Our System (F+C)			**0.823**	**0.785**
Step 2 of Proposed System (F)			0.552	0.771
Step 2 of Proposed System (C)			0.784	0.604
Google Trends			0.312	0.285
Twitter Trends			0.293	0.223
Features	Platforms	TPR	TNR	Precision
Step 3 of Proposed System (C+W+I)	**All Platforms**	**0.980**	**0.972**	**0.983**
Step 3 of Proposed System (C+W)	Twitter	0.967	0.955	0.966
	Facebook	0.955	0.950	0.975
	YouTube	0.967	0.962	0.963
	Reddit	0.967	0.968	0.964
	All Platforms	0.961	0.951	0.971
Step 3 of Proposed System (C)	Twitter	0.898	0.882	0.929
	Facebook	0.900	0.822	0.923
	YouTube	0.773	0.830	0.881
	Reddit	0.884	0.889	0.903
	All Platforms	0.809	0.792	0.920

C: UGC Context Features, T: UGC Timing Features, I: Infrastructure Features
F: Frequency Based Query Generation, C: Context Based Query Generation

from the remaining collected UGC, we labeled the UGC as malicious if the IP address of the destination site matched any of the labeled malicious UGC. Third, from the remaining collected UGC, we labeled UGC whose website's screenshot was visually similar to that of any of the labeled malicious UGC as malicious UGC. Finally, we labeled the UGC as non-malicious UGC. As shown in Table 4.2, we prepared a dataset with a minimum of 369 and maximum of 4,343 labeled malicious and non-malicious UGC for the four UGC platforms.

4.4.2 Detection Accuracy of Malicious UGC on Twitter

We compared the accuracy of Step 1 of the proposed system and two existing systems [53, 52] in terms of the binary classification of UGC on Twitter (malicious and non-malicious). We prepared two versions of Step 1, i.e., one version only used the UGC context features, and the other versions used both the UGC context features and UGC timing features (Section 4.3.1). Note that the existing systems [52, 53] are not open source; thus we reimplemented these systems in reference to the literature. We used Malicious Twitter and Non-malicious Twitter from the Twitter Stream Dataset to perform 10-fold cross-validation. Here, we considered three evaluations metrics, i.e., the true positive rate (TPR), true negative rate (TNR), and precision. The TPR is the ratio of correctly detected malicious UGC among all malicious UGC. TNR is the ratio of correctly detected non-malicious UGC among all non-malicious UGC. Precision is defined as the ratio of those detected as malicious UGC that actually is malicious UGC. Table 4.3 shows the results. The proposed system achieved a TPR value of 0.969, a TNR value of 0.985, and a precision value of 0.975, which are all higher than both existing systems using either feature set. These results demonstrate that our feature sets were more effective in terms of detecting malicious UGC than existing systems and features using one of feature sets (UGC context features and UGC timing features).

4.4.3 Collection Performance of Malicious UGC Candidates

We compared how efficiently each method collected malicious UGC on each platform with Step 2 of the proposed system and a baseline system that uses trending words on the web. Here, we prepared three versions of Step 2 of the proposed systems. The first version used *context-based query generation*, the second version used *frequency-based query generation*, and the third version used a combination of the two methods. For all three versions, we selected 1,000 queries generated using the *Search Results Dataset*, each in order of score. We also prepared two systems as baselines for using trending words on the web based on Google Trends, Twitter Trends, respectively. The Google Trends [73] baseline was obtained daily from the top 10 keywords trending in the United States from 09/15/2019 to 10/14/2019, and 1,000 keywords were selected randomly. The Twitter Trends [74] baseline was also obtained daily from the top 10 keywords trending in the United States from 09/15/2019 to 10/14/2019, and 1,000 keywords were selected randomly. We prepared two criteria, i.e.,*coverage* and *toxicity* to evaluate the collection efficiency of malicious UGC for search queries generated by each system. Coverage is defined as the number of prespecified malicious UGC among the collected UGC divided by the number of prespecified malicious UGC, and toxicity is the number of prespecified malicious UGC divided by the number of collected UGC. Here, greater values for these criteria indicate more efficient collection of malicious UGC. In this evaluation, the malicious UGC for each platform in the Search Result Dataset was considered to be prespecified malicious UGC. The criterion for prespecified malicious UGC is that all words in the search query are included in the text of the UGC. We defined coverage and toxicity this way because TPR, TNR and precision are not applicable in this case. Coverage and toxicity are suitable criteria because they allow us to calculate how leakproof the systems are and how much of an entire set of search queries is truly malicious, respectively. Table 4.3 shows the results. As can be seen, the proposed system achieved a coverage value of 0.823 and a toxicity value of 0.785, which indicates the best efficiency among the compared systems.

4.4.4 Detection Accuracy of Malicious UGC on Multiple UGC Platforms

We evaluated the accuracy of malicious UGC detection using the *Search Result Dataset*. Sections 4.4.2 and 4.4.3 compared parts of the proposed system and their baselines; however, here, we did not have a baseline to compare because, to the best of our knowledge, this is the first study to develop a system to detect malicious UGC of multiple UGC platforms in event-synced navigation attacks, which is the output of the proposed system. In this section, we evaluate the proposed system from two perspectives using the prelabeled *Search Result Dataset*. First, we prepared three versions of the three feature sets discussed in Section 4.3.3. The first version used only UGC context features, the second version used web content features and context features, and the third version used all feature sets, including the infrastructure features. Then, we prepared two versions of training data. One version used only the UGC of each individual platform, and the other version used all UGC of the four platforms. We combined them and compared them in terms of features and training data for all systems. Here, we use the Search Result Dataset to perform 10-fold cross-validation in addition to TPR, TNR, and precision. Table 4.3 shows the results. As can be seen, the proposed system achieved the best accuracy of 0.980 TPR, 0.972 TNR, and 0.983 precision

Table 4.4: Results of Detected Malicious UGC

	Nov. 2019	Feb. 2020	Total
# Collected UGC on Twitter	55,144,729	52,312,433	107,457,162
# Malicious Twitter UGC Seeds	73,218	85,895	159,133
# Context Based Search Queries	88,234	95,432	171,545
# Frequency Based Search Queries	92,321	99,623	182,467
# Selected Search Queries	52,867	59,431	105,423
# Malicious Twitter UGC	7,715	8,816	16,531
# Malicious Facebook UGC	5,840	6,710	12,550
# Malicious YouTube UGC	2,152	2,344	4,496
# Malicious Reddit UGC	3,124	2,945	6,069
# Unique Landing URLs	8,123	7,948	15,855
# Unique Malicious URLs	6,688	6,164	11,844
# Unique Malicious FQDNs	850	746	1,439

for both feature sets and the training data. We found that the infrastructure features improved the detection accuracy when the same attacker spreads attacks from multiple UGC platforms because these features can capture the characteristics of simultaneous postings of attacks. In addition, using the entire platform as training data, if the collected UGC was dissimilar to the malicious UGC in the training data for the given UGC platform but was similar to the malicious UGC in the training data of another platform, it could be classified as malicious UGC.

4.5 Measurement

We performed a measurement study to explore malicious UGC on four UGC platforms in the wild using the proposed system. Here, we continuously detected and collected malicious UGC with the proposed system for a total of 40 days during two different time periods (11/01/2019 to 11/20/2019 (20 days) and 02/08/2020 to 02/27/2020 (20 days)). The results are summarized in Table 4.4. We collected 107,457,162 UGC written in English and with external URLs from Twitter for 40 days. Then, Step 1 of the proposed system detected 159,133 malicious Twitter UGC seeds. In addition, Step 2 of the proposed system generated 105,423 search queries and retrieved UGC from each UGC platform. Step 3 of the proposed system successfully detected 16,531 malicious UGC on Twitter, 12,550 on Facebook, 4,496 on YouTube, and 6,069 on Reddit. These malicious UGC had 15,855 unique external URLs, and the final website had 11,844 unique URLs and 1,439 unique FQDNs. We found that 157 FQDNs of malicious websites were common for November 2019 and February 2020, and 139 (88.5%) of them showed no change in IP addresses linked to the FQDN even after three months. We also found that the proposed system detected malicious UGC within two hours after it was posted and before UGC platform providers took measures against the malicious UGC.

4.5.1 Analysis of Platforms

Abused Platforms. We aggregated malicious UGC whose final FQDNs were the same, and the number of FQDNs per UGC platform in Table 4.5. The number of FQDNs that originated from each UGC platform was 446 on Twitter, 531 on Facebook, 476 on YouTube, and 522 on Reddit. The number of FQDNs that originated from a single UGC platform was 210 on Twitter, 236 on Facebook, 164 on YouTube, and 338 on Reddit. In addition, the number of FQDNs that

Table 4.5: Number of FQDNs Distributed Each Platform

Sets	Description	# of FQDNs
T ∪ F ∪ Y ∪ R	All	1,439 (100%)
T	Originated from T	446 (31.0%)
F	Originated from F	531 (36.9%)
Y	Originated from Y	476 (33.1%)
R	Originated from R	522 (36.3%)
T − F − Y − R	Included only in T	210 (14.6%)
F − T − Y − R	Included only in F	236 (16.4%)
Y − T − F − R	Included only in Y	164 (11.4%)
R − T − F − Y	Included only in R	338 (23.5%)
T ∩ F	Shared on two platforms	60 (4.17%)
T ∩ Y	Shared on two platforms	82 (5.70%)
T ∩ R	Shared on two platforms	51 (3.54%)
F ∩ Y	Shared on two platforms	154 (10.7%)
F ∩ R	Shared on two platforms	51 (3.54%)
Y ∩ R	Shared on two platforms	45 (3.13%)
T ∩ F ∩ Y	Shared on three platforms	11 (0.76%)
T ∩ F ∩ R	Shared on three platforms	2 (0.14%)
T ∩ Y ∩ R	Shared on three platforms	3 (0.21%)
F ∩ Y ∩ R	Shared on three platforms	5 (0.35%)
T ∩ F ∩ Y ∩ R	Shared on four platforms	12 (0.83%)

T: # of FQDNs on Malicious Twitter UGC, F: # of FQDNs on Malicious Facebook UGC,
Y: # of FQDNs on Malicious YouTube UGC, R: # of FQDNs on Malicious Reddit UGC

originated from two UGC platforms was 60 on Twitter and Facebook, 82 on Twitter and YouTube, 51 on Twitter and Reddit, 154 on Facebook and YouTube, 51 on Facebook and Reddit, and 45 on YouTube and Reddit. The number of FQDNs that originated from three UGC platforms was 11 on Twitter, Facebook, and YouTube; two on Twitter, Facebook, and Reddit; three on Twitter, YouTube, and Reddit, and five on Facebook, YouTube, and Reddit. The number of FQDNs that originated from all four UGC platforms was 12. Note that 491 (34.1%) of the 1,439 FQDNs of the malicious websites we found were spread from at least two UGC platforms, which agrees with our expectations. In addition, by focusing on each UGC platform, we found that the number of FQDNs originating from multiple UGC platforms was 236 (52.9%) of 446 on Twitter, 295 (55.6%) of 531 on Facebook, 312 (65.5%) of 476 on YouTube, and 184 (35.2%) of 476 on Reddit. While 65.6% of the attacks on YouTube were also found on other UGC platforms, only 35.2% of the attacks on Reddit were found on other UGC platforms. Among the four UGC platforms, YouTube was most commonly found to allow the same attacker to spread attacks on multiple platforms, and Reddit was the least used UGC platform for attacks on multiple platforms. Reddit has fewer users than the other UGC platforms [45], and it is assumed that attackers do not often target Reddit when selecting multiple UGC platforms to spread attacks. In addition, the UGC platform combination with the highest co-occurrence of the same FQDN was Facebook and YouTube with 154 (10.7%) FQDNs. In terms of the number of users, Facebook and YouTube are the world's largest and second largest UGC platforms, respectively; thus, we assume they are likely to be targeted by attackers when selecting multiple UGC platforms to spread attacks. We found a few attacks that originated from three and four UGC platforms, most attacks originated from only one or two UGC platforms and were spread at specific times.

Deleted by Platform Providers. We also investigated whether malicious UGC detected by the proposed system was removed by the UGC platform providers after a certain period. Here, we analyzed how differently each UGC platform

Table 4.6: Number of Malicious UGC Deleted by Providers

Platform	One week later	One month later	Three months later
Twitter	2,992 (18.1%)	9,241 (55.9%)	10,298 (62.3%)
Facebook	1,144 (9.12%)	5,032 (40.1%)	5,271 (42.0%)
YouTube	4,104 (91.3%)	4,185 (93.1%)	4,298 (95.6%)
Reddit	4,035 (66.5%)	4,855 (80.0%)	4,928 (81.2%)

responds to malicious UGC and how this changes over time. Here, the criterion used to determine if UGC was removed was if its URL was replaced with a page that contains a freeze or deletion string after a certain period. Note that such pages are not displayed if they are deleted by the authors themselves; thus, we can distinguish between deletion by the UGC platform and deletion by attacker. In addition, malicious UGC does not appear in the search results after it is removed by a UGC platform; thus, its existence in the search results of the proposed system means that it was correctly detected earlier than it would have been by the UGC platform. Table 4.6 shows the percentage of UGC removed from each UGC platform one week, one month, and three months after being posted. For example, after three months, the UGC platform providers had removed 62.3% (Twitter), 42.0% (Facebook), 95.6% (YouTube), and 81.2% (Reddit) of the malicious UGC detected by the proposed system. From the results, we found that the percentage of malicious UGC removed differed greatly among UGC platforms. In particular, we found that one week after the malicious UGC was posted, 91.3% were removed on YouTube, while only 9.12% were removed on Facebook, which indicates a significant difference in the platforms' responses. One reason for this difference among UGC platforms is the difference in characteristics between malicious and non-malicious UGC posted to each UGC platform. For example, on YouTube, content is generated mainly from videos; however, malicious UGC has distinctive characteristics, e.g., low playback time (i.e., the videos are meaningless), a large amount of text in the description, and URLs to external websites. Therefore, we consider that malicious UGC can be detected more efficiently on YouTube compared to other UGC platforms because malicious UGC on YouTube has more indicators that can be assessed as malicious. In addition, focusing on the transitional period, we found that the number of deletions increased from 18.1% to 55.9% on Twitter and 9.12% to 40.1% on Facebook, although most malicious UGC was not deleted within one week of being posted. However, we can confirm that the percentage of deletions one month and three months after being posted did not change significantly on any UGC platform compared to the percentage of deletions one week after being posted. This means that UGC that had passed through the detection logic of each UGC platform was no longer supported, and the UGC platform provider may check again when the same attacker account remains active and newly generated content is detected. Thus, we conclude that no UGC platforms respond sufficiently to threats related to the event-synced navigation attack (Section 4.2.2), considering there is a high probability that a user will reach a malicious website if the threats are not responded to during the period of a given event or within a few hours.

4.5.2 Analysis of Detected FQDNs

Lifetime. We analyze the lifetime of FQDNs found in this experiment. Here, we used the passive DNS database [79] to investigate how long the combination of FQDNs and IP addresses obtained during the crawl by the proposed system

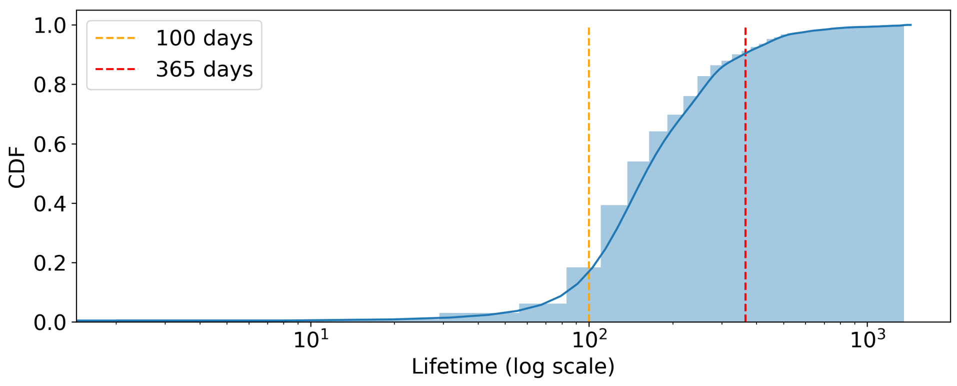


Figure 4.3: Lifetime of Detected FQDNs

Table 4.7: FQDN Accesses

Platform	# FQDNs	Mean	Median	Max	Total
Single	948	1,040	212	52,356	985,598
Multiple	491	4,184	654	453,282	2,054,232

survived. The passive DNS database [79] allows us to obtain the period of time and number of name resolutions of each FQDN. Note that this investigation was limited to the combination of FQDNs and IP addresses obtained from crawling and did not consider other IP addresses to which the FQDNs were linked in the past. Therefore, we define the survival period as the period during which the FQDN was linked to a single IP address. Figure 4.3 shows the results of the survival analysis where the x-axis is the log-scale lifetime (in days), and the y-axis is the value of the cumulative distribution function of FQDNs. The yellow line shows that 1,264 (87.8%) of the FQDNs survived for more than 100 days, and the red line shows that 133 (10.8%) of the FQDNs survived for more than 365 days. These results demonstrate that FQDNs associated with a single IP address survived for a long time and were not being taken-down.

Accesses. We analyzed the impact of FQDNs on users we found in our measurement. Here, we used the passive DNS database in the same manner. The number of user accesses was defined as the number of name resolutions for a combination of FQDNs and IP addresses obtained by crawling the UGC at the time it was posted. As shown in Table 4.7, in FQDNs derived from multiple platforms, the mean accesses was 4,184, the median was 654, the maximum was 453,282, and the total was 2,054,232. We found that FQDNs derived from multiple platforms had more than four times as many accesses as those derived from a single platform. Note that these accesses are based on the passive DNS database; thus, it is assumed that the actual number of accesses is much higher because passive DNS is the number of accesses for some access routes.

4.5.3 Analysis of Detected Website Categories

We investigated how the malicious websites that we detected behaved in relation to users. First, we used VirusTotal [80] to check if malicious websites' URLs were positive for any type of malware in any of the scanning engines and found only 9.2% (1,090 URLs) to be positive. The SE attack does not show clear malicious behavior; thus, automatically detecting them early with high accuracy is difficult. Then, we randomly sampled 1,439 URLs, one from each of the FQDNs

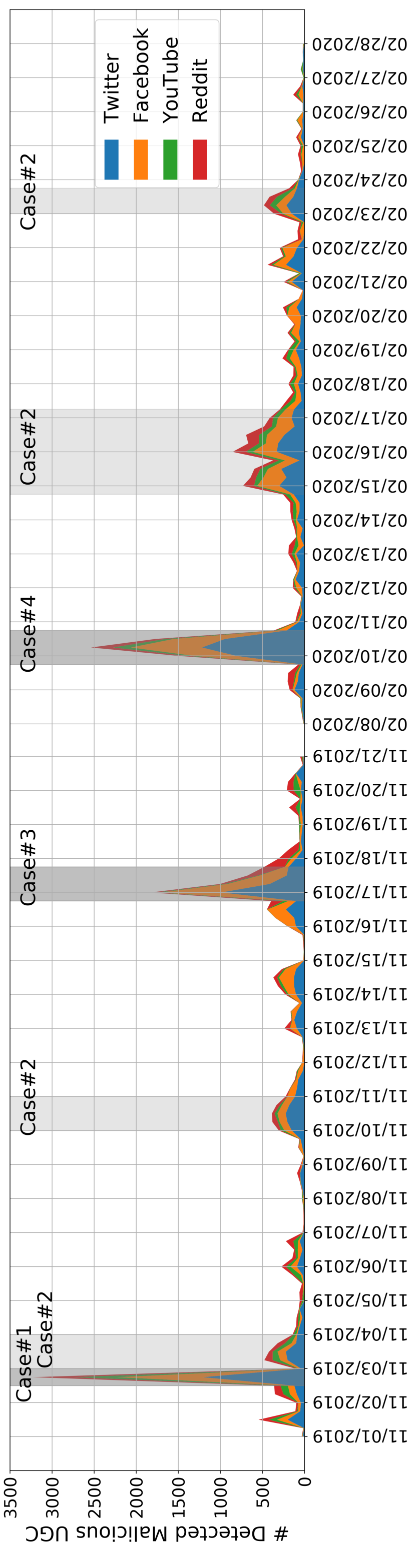


Figure 4.4: Transition in the Number of Detected Malicious UGC Every 6 Hours

we detected in this study and checked how a malicious website behaves toward users by manually reviewing screenshots and clicking on buttons in the malicious website. As a result, it was found that the sites could be classified into seven different categories. Note that there are cases where a single FQDN has multiple malicious behaviors. For example, there were malicious websites that request users to enter both residential address and credit card information on a single

web page.
Steal credit card information. We found 411 (28.6%) FQDNs that asked users to enter credit card information. Many sites attempted to steal information by saying that they do not charge a fee but are required to verify personal information.
Steal street address information. We found 53 (3.68%) FQDNs that asked users to enter residential address information. Attackers entice users with discounted goods to convince them to enter personal information on fake shopping sites.
Steal email address and password. We found 525 (36.5%) FQDNs requested users to enter a combination of an email address and password. If the user enters a combination that they use frequently, the attacker will succeed in stealing user credentials. In this study, the most common pattern was to deceive people into entering this information by imitating a legitimate video streaming website.
Install browser extensions. We found 98 (6.81%) FQDNs prompting users to install a Google Chrome extension plugin. There are reports that a large number of extended plug-ins threaten the security and privacy of users [81]. Since malicious FQDNs can be detected by VirusTotal, it is highly likely that the extension plugin contains embedded code that steals information.
Complete surveys. We found 102 (7.09%) FQDNs that are more likely to be related to survey scams. There are reports [48] of malicious websites that trick users with the phrase "answer a number of questions posted and get an iPhone." We confirmed that none of the FQDNs contain any well-known survey sites (e.g., Google Forms [82] and Survey Monkey [83]).
Allow web notifications. We found 84 (5.84%) FQDNs that requested web notification permission. It has been reported that if users allow such notifications, they will receive numerous malicious advertisements [84]. As the detected FQDNs do not include popular websites (i.e., Alexa Top 10,000 sites), it is possible that suspicious push notifications will be delivered to users who have authorized web notification if user continues to observe push notifications for a long period of time.
No obvious malicious behaviors. We found 198 (13.8%) FQDNs that did not show any malicious behavior. Many sites were adult sites, and the attacker's objective is to increase traffic and generate advertising revenue. In addition, it is possible that the attacker has removed the malicious behavior or that a valid referrer or user-agent is needed to trigger the attack.

4.5.4 Case Study: Abused Events

We investigated what real-world events are used in the context of inducement and to what extent the attacks are spread on each platform. Figure 4.4 shows the number of malicious UGC detected by our proposed system. The x-axis is the date and time at 6-hour intervals, and the y-axis is the number of malicious UGC detected on each platform in 6 hours. We can confirm that when a lot of malicious UGC is detected on one UGC platform, a lot of malicious UGC is also detected on other UGC platforms. The amount of malicious UGC increases when attacks by the same attacker, attacks with the same target, and attacks that reach the same malicious websites are spread from multiple UGC platforms. We now focus our analysis on four real-life events (Case#1, Case#2, Case#3, and Case#4). Note, each case has a different period or periods.

Case#1 (11/02/2019 12:00 – 11/02/2019 24:00) involved the 2019 Rugby World Cup Final between England and South Africa. The number of malicious

UGC detections increased significantly on all UGC platforms, and more than 3,000 malicious UGC were detected on all UGC platforms within 6 hours. We investigated malicious UGC containing the strings "rugby," "world cup," "England," or "South Africa" and found 1,014 on Twitter, 928 on Facebook, 495 on YouTube, and 412 on Reddit. We found malicious websites created specifically for the Rugby World Cup. These websites had FQDNs such as *watchrugbyworldcup[.]com* and *rugbyworldcuplive2019[.]com*. In addition, we found cases of malicious websites that mimic general sports websites, such as *sport1.sportplay-live[.]club* and *espn.officialpage[.]us* These websites used different destinations in the same FQDN depending on the context, such as rugby-worldcup[.]php?*live=england+vs+south+africa* in the path section.

Case#2 (11/03/2019 00:00 – 11/03/2019 24:00, 11/10/2019 00:00 – 11/10/2019 24:00, 02/14/2020 18:00 – 02/17/2020 06:00, 02/23/2020 00:00 – 02/23/2020 18:00) considered the period when soccer matches in the English Premier League were being held. We found that the number malicious UGC detected increased periodically on all UGC platforms, although the amount of malicious UGC detected is not as high as in Case#1. We investigated malicious UGC containing the strings "premier" and/or "league" and found 1,854 on Twitter, 1,725 on Facebook, 412 on YouTube, and 988 on Reddit. Among the detected UGC, we observed 88 FQDNs of malicious websites in general sports-related contexts such as *sport1.sportplay-live[.]club* and *sportivi-play[.]club* in both Nov. 2019 and Feb. 2020. In other words, the same FQDN is used and continues to be spread as an actual attack without countermeasures.

Case#3 (11/16/2019 18:00 – 11/17/2019 18:00) involved periods when NCAA Football and High School Football games in the United States occurred at the same time. We investigated malicious UGC containing the strings "ncaa," "high school," or "league," and found 982 on Twitter, 789 on Facebook, 55 on YouTube, and 289 on Reddit. There were 83 FQDNs with "ncaa" in the UGC text and 77 FQDNs with "high school" in UGC text, with 53 FQDNs in common. These results reveal that the same attacker disseminates attacks in similar contexts for similar events. In this context, the spread on YouTube is relatively low, and it is inferred that the UGC platform on which attacks are spread varies depending on the attacker or the characteristics of the context.

Case#4 (02/09/2020 18:00 – 02/10/2020 18:00) focused on the Academy Awards, also known as The Oscars. We investigated malicious UGC containing the strings "academy," "awards," or "oscars" and found 1,432 on Twitter, 1,123 on Facebook, 205 on YouTube, and 242 on Reddit. These results indicate that the attackers will use any event that is likely to attract a large number of people.

4.5.5 Case Study: Directory Listings

We found 49 FQDNs for which directory listings could be viewed by anyone due to a misconfiguration on the attacker's server side. UGC represents a part of the entire attack; however, to clarify the actual situation and consider countermeasures the scale of the attack must be understood from the infrastructure side. Therefore, we regularly monitored the information in the directory listings to investigate the attack ecosystem. The monitoring procedure is as follows. First, we accessed "http://[detected FQDN]/". Next, we determined if the string "Index of /" is in the HTML title tag (i.e., <title>). Finally, when directory listings were available, we monitored changes in web structure by retrieving web content every hour. Here, we consider the website to have been updated if a new date and time for the last modification have been added. When the website has been updated, the interval between the newly added date and the previous date is defined as the *update interval*.

As a result of monitoring changes for the 49 FQDNs, we observed a total of 1,361 website updates from the time each FQDN was initially found until it could no longer be found. First, we investigated the update interval. The minimum value of the update interval was one minute, the maximum value was 1,439 minutes, the mean was 388.1 minutes, the median was 170.0 minutes, and the standard deviation was 441.0. Specifically, the attacker updated 467 times within 60 minutes, and the maximum update interval was 1,439 minutes. In other words, we found that the attacker frequently updated the content of the malicious website within a few hours of the original posting and updated it at least once a day. We assume that malicious website are updated frequently to change the content in accordance with the context in order to lead users to the malicious website, and the attack is carried out with the content in accordance with the target of the attack.

Next, we monitored the web structure every hour and investigated the number of directories shown in each FQDN's website. The minimum value of the number of directories for each FQDN was one, the maximum value was 89, the mean was 17.14, the median was 11, and the standard deviation was 20.0. Considering the fact that one path is one induction context, we found that one FQDN was able to induce users with up to 89 different contexts of malicious websites. Specifically, we found that 89 different directory sections existed for the two FQDNs among the 49 FQDNs available for directory listing. However, the observation periods differed between Nov. 2019 and Feb. 2020, and the same FQDN led users to malicious websites in various contexts. Furthermore, since the average number of directory parts is 17.14 per FQDN, it is evident that the attacker reuses their server resources to lead users to malicious websites in various contexts. In other words, we conclude that early detection of these websites is effective for eradicating attacks.

4.6 Discussion

In this section, we outline limitations of our method and experiment, ethical considerations related to our experimental setting, and possible countermeasures for stakeholders against event-synced navigation attacks.

4.6.1 Limitations

Detection evasion. The key idea of the proposed system is to focus on attacker activity: to direct users to malicious websites, an attacker produces UGC related to a specific context and synchronizes the distribution of malicious UGC with an actual event. However, if an attacker generates UGC in a context where it has not appeared in previous attacks or distributes UGC at a time other than the time of the actual event, the proposed system will overlook the attack. However, if an attacker uses such strategies to evade detection, the chances of malicious UGC attracting user attention will be reduced and the attack will be less efficient. In addition, attackers can employ techniques to evade our website crawling process: i.e., IP cloaking and user interaction. Because the proposed system is assigned a single specific IP address for crawling external websites linked by UGC, malicious websites owned by the same attacker can cloak themselves against an IP address that accesses several of them. Our system for website crawling manipu-

lates HTTP protocol-level redirects (e.g., HTTP 301 status code), HTML-level redirection (e.g., iframe and meta tags), and JavaScript-level redirection automatically; however, a user-interaction-based website transition (e.g., clicking a button or link) is beyond the scope of our crawling. Although the proposed system has the aforementioned limitations, our measurement procedure enables us to estimate a lower bound for the number of event-synced navigation attacks.
Changes over time. We generated a training model on the basis of malicious UGC features and corresponding malicious websites collected as of October 2019. Our measurement study demonstrates that the proposed system can detect a large number of malicious UGC for at least several months (i.e., the end of February 2020). However, if the strategy of event-synced navigation attacks changes in the future (e.g., changing trends in writing style used for navigation or using images), the training model will need to be retrained. We leave the evaluation of the robustness of training and retraining the model as future work.

4.6.2 Ethical Considerations

We were aware that there were ethical considerations related to collecting a large amount of UGC from various UGC platforms. We searched for and collected UGC as an end user of UGC platforms because there is no alternative way to do so. To respect the acceptable use policies of UGC platforms and reduce potential harm to such platforms, we followed the best practices of related studies to conduct our experiment. We believe our experiment never affected the availability of platforms (i.e., production services) because our system throttled the requests to each UGC platform (100 requests per hour at most). To send requests in a relevant manner, we utilized the UGC platforms' public application programming interfaces or commonly used open-source tools for accessing UGC platforms. Among the four UGC platforms we investigated, only Facebook requires an account to access UGC. Thus, we used our own legitimately registered accounts for Facebook.

4.6.3 Countermeasures

UGC Platform Providers. Although UGC platform providers are making efforts to detect and eliminate malicious UGC, we have shown that they fail to detect a lot of it. The features leveraged by the proposed method can help UGC platform providers detect malicious UGC. We found that numerous attacks originate from multiple UGC platforms and lead to the same malicious website. To counter such attacks, if UGC platform providers share the incomplete malicious UGC lists that each platform has individually, they may be able to expand detection coverage and improve the security of each other's platforms.
End Users. End users on UGC platforms can take measures to protect themselves. First, users should consider the risks involved in clicking on URLs in UGC. Users can reduce the chances of being victimized by simply checking that the account posting the UGC is official before clicking. Not all links are malicious; the option to connect to an external website, e.g., video feeds related to sports events, are often presented by official accounts. Second, users should proactively report suspicious content they come across to the point of contact provided by each UGC platform. In fact, many attacker accounts have been deleted on user notifications [85].

4.7 Related Work

In this section, we summarize previous studies related to UGC, attack scenarios, and efficient malicious URL detection.

User-generated Content (UGC). A significant number of studies have analyzed malicious UGC from various perspectives [52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]. For example, Gao et al. analyzed Facebook for malicious UGC and reported that 70% of the posts that include a URL led to phishing websites and 97% of the accounts that posted them were compromised [59]. Gao et al. also proposed a system that enables real-time detection of malicious UGC using features common to Twitter and Facebook [52]. Lee et al. proposed a method, specifically for Twitter, to detect malicious UGC and pointed out that, unless they are accessed with a real browser, many URLs in malicious UGC redirect to legitimate websites rather than malicious websites [53]. However, all previous studies were based on the analysis and detection in individual UGC platforms; the relationship between UGC on multiple UGC platforms has not been analyzed.

Attack Scenario. Attackers lead users to malicious websites in various contexts. Kharraz et al. developed a classifier to accurately distinguish between legitimate surveys and survey scams that lead users to malicious websites under the guise of receiving a benefit from answering a questionnaire [48]. They reported that 40% of the pages that led to malicious survey scams were in the top 30,000 Alexa sites. Rafique et al. created a classifier that accurately identified free live streaming services, which are commonly used to spread deceptive advertisements [78]. They observed attacks that led to malicious browser extensions or fraudulent sites and reported that those websites were hosted in Europe and Belize. Miramirkhani et al. conducted a large-scale analysis of technical support scams that defraud users of exorbitant amounts of money by imitating a malware infection on their computers. They actually contacted 60 fraudulent technical support groups and were able to identify the tools and scenarios that were used in the attacks [49]. However, neither the attacks induced in relation to real events nor the malicious websites related to specific events were analyzed.

Efficient Malicious URL Detection. Invernizzi et al. showed that malicious pages related to drive-by download attacks can be efficiently collected by generating search queries from known malicious pages [68]. Thomas et al. designed a feature selection method to detect email and Twitter-derived spam with high accuracy [86]. They found that the characteristics of these spams differ significantly, and that using the opposite characteristics, for example, detecting email spam using the characteristics of Twitter spam detection, does not improve the accuracy. These studies primarily focus on search engines and compare email and UGC. To the best of our knowledge, no studies have analyzed attacks on multiple UGC platforms using the search function of UGC platforms.

4.8 Conclusion

In this paper, we defined a threat model in which an attacker leads users to a malicious website from multiple UGC platforms using real-life events to which people pay attention at specific times (i.e., event-synced navigation attacks). We proposed an innovative three-step system to collect and analyze event-synced navigation attacks from UGC platforms in real time. We evaluated the collection efficiency and detection accuracy of malicious UGC in the three steps of the proposed system and confirmed that the system can classify malicious and non-malicious UGC with 97% accuracy [67]. In addition, we investigated event-synced navigation attacks using the proposed system and found that 39,646 malicious UGC were spread over a period of 40 days in the wild. We found that 34.1% of the malicious UGC were spread from multiple UGC platforms and that attacks originating from multiple platforms were accessed more than four times as frequently as attacks originating from a single platform. We also found that 87.8% of FQDN associated with malicious websites survive for more than 100 days and that countermeasures taken by the UGC platform only covered 31.0% of the malicious UGC we detected in this study even though the malicious websites were accessed frequently. In this study, we conducted experiments under specific conditions (a single IP address), thus our system may not be able to reach the malicious website due to cloaking when the user is not the target of the attack. In addition, we cannot guarantee the detection accuracy of the proposed system against new attacks that will appear in the future because we conducted experiments on data from a specific period of time. We selected four UGC platforms with many users and conducted experiments on them. The features of the proposed system are agnostic to UGC platforms and can be applied to new ones that may appear in the future. We hope that the results of these investigations will be useful for future research and the development of countermeasures to event-synced navigation attacks.

Chapter 5

Understanding Characteristics of Phishing Reports from Experts and Non-experts on Twitter

5.1 Introduction

A phishing attack involves an attempt by an attacker to deceive a user into believing that a harmful website is authentic, with the aim of acquiring valuable information like account credentials or credit card details. Recently, phishing attacks have increased globally [87, 88, 89, 90], especially attacks targeting mobile devices, with a 3.28-fold increase from 2020 Q2 to 2020 Q3 [89]. In addition to the traditional phishing attacks via e-mail and short message service (SMS) have been especially on the rise [91]. Smishing, a portmanteau of "SMS" and "phishing," refers to phishing attacks that specifically exploit smartphone SMSs to deceive users into providing sensitive information or clicking on malicious links. Attackers are exploiting SMS features for phishing: it can be sent with a phone number, with a much smaller namespace than an email address; it can be reliably pushed to cell phone subscribers when they are in range; and SMS is used for legitimate notifications and two-factor authentication, making it impossible to ignore completely.

The first step in timely combatting this ever-increasing number of phishing attacks is to collect a wider range of phishing cases that reach end users and continue understanding their characteristics. In fact, to that end, numerous studies have been conducted to measure and analyze phishing attacks [92, 93, 94, 95]. The facts about phishing and the weaknesses of the countermeasures revealed by these studies at that time have helped improve the coverage of spam filters in email services (e.g., Gmail and Outlook), web browser blocklists (e.g., Google Safe Browsing [96] and Microsoft Defender SmartScreen [97], threat feeds (e.g., PhishTank [98] and OpenPhish [99]), and security analysis engines (e.g., VirusTotal [80] and urlscan.io [100]).

However, existing countermeasures are still insufficient when phishing messages reach end users and users encounter phishing sites. This raises the following question for us. *How can we collect phishing that reaches users bypassing existing countermeasures?*

In this study, we propose an approach that uses Twitter as a new observation point to immediately collect *actual phishing situations* encountered by users that have bypassed existing countermeasures and to understand the characteristics of such phishing. Some previous studies have also used Twitter as a source to extract *non-phishing* cyberattack information (e.g., vulnerability information and malware behavior information) [101, 102, 103, 104] and limited phishing cyberattack information (e.g., search by fixed keywords or monitor only specific users) [105, 103, 106]. Specifically, these previous studies used Twitter posts of the cyberattack information by *security experts*, which allowed them to identify vulnerability information and indicator of compromises (IOCs) before they were published on databases that share vulnerability information (e.g., Common Vulnerabilities and Exposures numbers) and threat information (e.g., malicious URLs) such as the National Vulnerability Database [107] and VirusTotal [80]. While at first glance these studies appear to be close to what our study aims to do, they differ significantly in that our goal is to extract and analyze *phishing*-related information even from the actual situations that reach *non-experts*. Indeed a large number of non-experts have posted suspicious phishing attack-related cases on Twitter as alerts [108]. We are eager to immediately analyze the content of alerts they report as cases where phishing has reached users because existing countermeasures have been bypassed. These reports have the benefit of being more victim-centered and comprehensive than posts by security experts and potentially being used as new information for anti-phishing technology. Our challenge is to extract only phishing attack reports from a large number of irrelevant tweets in their everyday lives.

To this end, we propose CrowdCanary, a system capable of structurally and accurately extracting phishing information (e.g., URLs and domains) from tweets of experts and non-experts who have actually discovered or encountered phishing. CrowdCanary is a system that employs pre-selected keywords (e.g., phishing and scam) as input to identify and output phishing attack-related user reports. Additionally, CrowdCanary can collect a diverse set of tweets by automatically identifying and extracting new keywords that are often seen in such reports and adding them to the system. We evaluate the effectiveness of our malicious URL collection in CrowdCanary against security engines [80], as well as existing systems that collect attack information from Twitter [105, 109]. We also analyzed the differences between experts and non-experts and considered what approach should be taken to collect the information shared by non-experts. Finally, we discussed how the phishing information extracted by CrowdCanary could be analyzed to help protect actual end users.

Our primary contributions are as follows.

- We proposed CrowdCanary, a system that identifies reports of phishing attacks by both English and Japanese Twitter users with a high accuracy rate of 95% for evaluation data.

- We operated CrowdCanary for three months and were able to identify 38,935 phishing reports out of 19 million tweets and extract 35,432 phishing URLs. We confirmed that 31,960 (90.2%) of these phishing URLs were later detected by anti-virus engines, demonstrating the high accuracy of CrowdCanary's threat intelligence extraction

- We analyzed users who shared phishing reports and discovered that the majority of phishing reports detected by CrowdCanary were shared by non-experts. We showed that the threat intelligence reported by non-experts includes many URLs not included in the intelligence shared by experts, making it useful as a new observation point for phishing attacks from a more victim-friendly perspective.

This paper is an extended version of our paper presented at ARES 2023 [110]. Our previous paper proposed a system to detect reports of phishing attacks on

Twitter by both experts and non-experts, evaluated its ability to detect them with high accuracy, and analyzed the differences between expert and non-expert reports. However, we did not perform a comprehensive analysis on the intelligence within the detected reports, such as comparing the extracted useful information to actual phishing-specific data feeds, examining the actual attack infrastructure based on the detected phishing attacks. In this study, we analyzed information on phishing attack reports from a new perspective and uncovered previously unidentified insights (Section 5.7). The new contributions of this paper are as follows.

- We compared the phishing URLs detected by CrowdCanary to those from two data feeds specialized in phishing attacks and found that more than half of the phishing URLs detected by CrowdCanary represented unique threat intelligence. Furthermore, we discovered that CrowdCanary was able to identify about 80% of the common URLs more rapidly than the other two feeds, demonstrating its superior detection speed compared to existing technologies.
- We conducted an analysis of the domain names and hosting providers that attackers typically use to deploy phishing sites, using the collected phishing URLs. We found that the phishing sites in our study have a bias toward certain top level domains that are generally regarded as malicious, and that the hosting providers to which they are deployed are biased toward a small number of IP addresses, many of which are controlled by organizations in the United States.

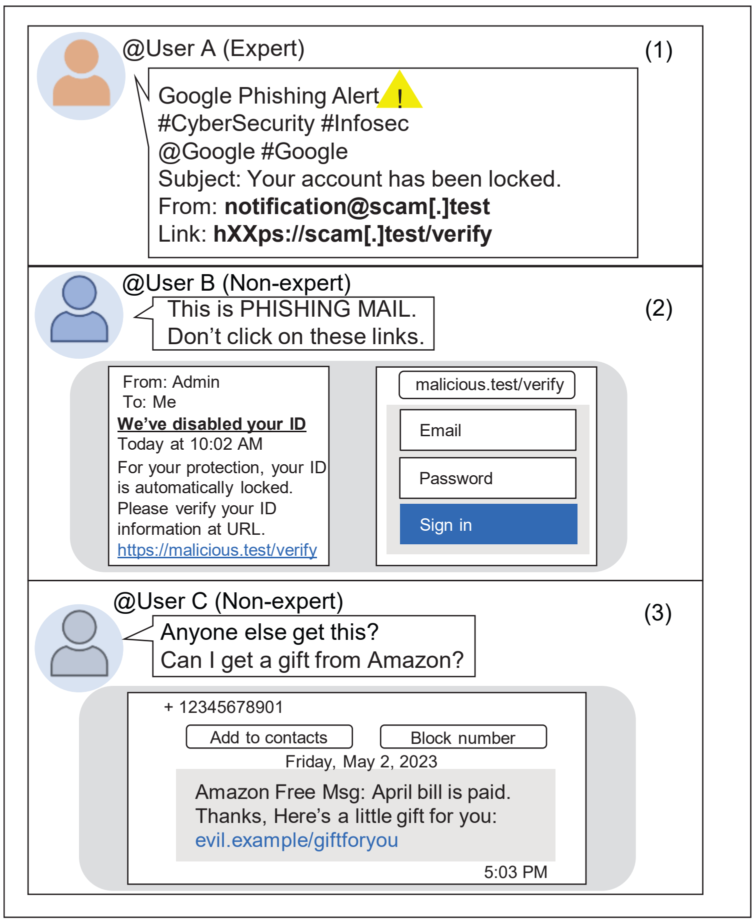
5.2 Motivating Examples

In this section, we discuss examples of user-reported phishing attacks and the challenges of extracting URLs and domain names related to phishing attacks.

5.2.1 Reports on Phishing Message

With the increased usage of social media platforms and smartphones, people post phishing emails and SMSs content they discover or encounter [108]. Figures 5.1 (1), (2), and (3) show reports of phishing attacks posted by users on Twitter, which we refer to as cases (1), (2), and (3), respectively. These are examples where Twitter users discover or encounter a phishing email or SMS and share that information along with the tweet's text or a screenshot taken with their smartphone.

In case (1), a user discovered Google phishing emails. He/she used hashtags and mentions to alert Twitter users to the email title, the sender's email address, and the phishing URL. It's relatively easy for us to collect reports and extract information if the report includes alerting hashtags or mentions the company's official account, and if the threat intelligence is in the body of the tweet. In case (2), a user clicks on a URL in a phishing email, understands that he/she has arrived at a phishing site, and shares a screenshot of the email and his/her browser. You can find the URL and domain name related to the phishing site in the information. In case (3), a user shares a phishing SMS he/she received to get feedback because he/she are unsure if the information is real or fake. In addition to the URL in the SMS, the text of the tweet and SMS contains the company string "Amazon," which was abused in the phishing attack. Compared to case (2), this case lacks keywords such as "PHISHING". Therefore, to collect


@User A (Expert) (1)

Google Phishing Alert ! #CyberSecurity #Infosec @Google #Google
Subject: Your account has been locked.
From: **notification@scam[.]test**
Link: **hXXps://scam[.]test/verify**

@User B (Non-expert) (2)

This is PHISHING MAIL. Don't click on these links.

From: Admin
To: Me
We've disabled your ID
Today at 10:02 AM
For your protection, your ID is automatically locked. Please verify your ID information at URL.
https://malicious.test/verify

@User C (Non-expert) (3)

Anyone else get this? Can I get a gift from Amazon?

Friday, May 2, 2023
Amazon Free Msg: April bill is paid. Thanks, Here's a little gift for you: evil.example/giftforyou
5:03 PM

Figure 5.1: Reports on Phishing Messages

such phishing reports, we need to monitor Twitter at the right time and with the relevant keywords. Specifically, we need a system that can extract the keyword "Amazon" when phishing attacks with context related to "Amazon" are prevalent and promptly collect phishing reports from Twitter using that keyword. We will have important information about phishing attacks if we can extract URLs, domain names, and exploited company brand names as character strings from collected reports. Because this information is based on live phishing attacks that bypassed existing countermeasure technologies and reached end users, it is valuable to consider better countermeasure technologies to detect and prevent phishing attacks before they reach users.

5.2.2 Challenges

Collecting phishing-related posts from users and extracting only phishing-related information from them presents three challenges.
Collection of posts from various users on Twitter. There are a lot of tweets on Twitter, including phishing reports from security experts and non-experts. To examine them realistically, we need to collect the tweets as narrowly as possible. However, keywords commonly used by security experts in their reports, such as "#phishing.", are not always included in the reports of security non-experts. Therefore, we need to dynamically determine keywords to include in phishing reports and collect tweets at the right time to collect reports from a wide range of users.

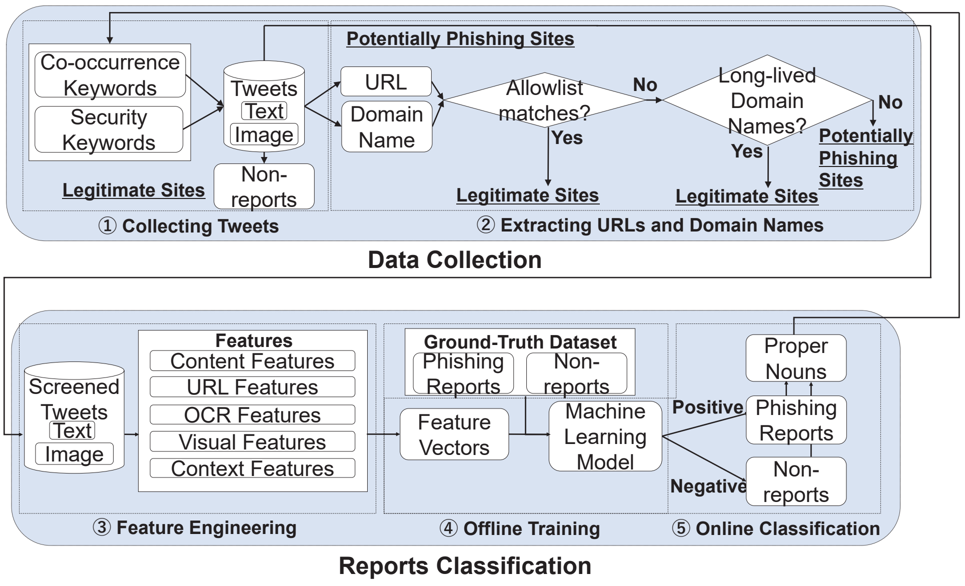


Figure 5.2: Overview of CrowdCanary

Extraction of information from collected user posts. Phishing reports from non-experts are often presented in more diverse formats than those used by security experts. For example, phishing-related information may only be included in the image of a tweet, not in the body of the tweet. Without human intervention, it is difficult to determine whether the tweet is a report related to sharing information about phishing attacks from texts and images. Since we cannot manually analyze all tweets, we need a mechanical way to extract information from both the texts and images of a large set of tweets.
Validation of extracted information. It is necessary to extract only information about URLs and domain names related to phishing attacks from user reports. Some of the information we collect may be user-generated misinformation about legitimate sites or entirely unrelated to phishing attacks. As a result, we need to confirm the accuracy of the information extracted from the texts and images of the collected reports.

5.3 Proposed System: Data Collection

We propose CrowdCanary, a system that collects large-scale reports of phishing attacks in English and Japanese from Twitter users, including experts and non-experts, and allows for structured and accurate extraction of phishing information. We selected English and Japanese as the languages for our analysis because they are the top two languages used by Twitter users and thus likely to share information on phishing attacks using those languages [111]. Figure 5.2 shows an overview of CrowdCanary. CrowdCanary has two core components, *Data Collection* and *Reports Classification*. In this section, we describe the first component of CrowdCanary, *Data Collection*. This component takes keywords as input for searching tweets, collects data for *Report Classification*, and outputs them at one-hour intervals. The one-hour collection interval is a customizable system parameter. This component is designed to collect a wide range of tweets related to phishing attacks from different users. In addition, this component extracts information about URLs and domain names that are candidates for phishing sites from the collected tweets, and excludes information that is in a no-

Table 5.1: Selected Security Keywords (English)

Keywords Related to Security Threats	Cyber Attack, Fake Site, Fraud, Scam, Malicious Site, Phishing, Opendir, Spam, Social Engineering, Smishing
Keywords with Frequent Shared Security Threats	#CyberCrime, #CyberSecurity, #CyberThreat, #IdentityTheft, #InformationSecurity, #InfoSec, #EmailSecurity, #ThreatHunting, #Threat, #Security

tationally invalid form or related to legitimate sites. This component consists of the following steps: *Collecting Tweets* and *Extracting URLs and Domain Names*

5.3.1 Collecting Tweets

In this step, we collect tweets using two types of keywords, *Security Keywords*, which are often used to share security information, and *Co-occurrence Keywords*, which co-occur with *Security Keywords* only at certain times. We use the Twitter Search API [112] as a means of collecting tweets. Otherwise, equivalent analysis can be performed using a stream of tweets as input, such as the firehose API [113] or the Decahose API [114] (10% random sampling of the firehose API). Since a large number of users on Twitter routinely post tweets that are unrelated to phishing reports, we considered that a search approach using appropriate keywords would be more efficient in collecting candidate reports of phishing attacks than an analysis of all such tweets or a random sampling of tweets.

Security Keywords. *Security Keywords* in this paper refers to keywords that are regularly posted on Twitter for cybersecurity-related information. *Security Keywords* allows us to collect tweets from security experts and tweets from non-security experts sharing phishing attacks they have discovered. Specifically, we select multiple keywords from two perspectives: related to the attack type (e.g., phishing) and information sharing (e.g., #infosec). The keyword defined as attack type (e.g., phishing) is sometimes used as a hashtag (e.g., #phishing), which is also included in the search. Finally, we selected the 20 security keywords in Table 5.1 for the following experiments. Based on previous researches [103, 104] and our preliminary study, we selected keywords most likely to be shared on Twitter for information about phishing sites. We also selected the same number of *Security Keywords* in Japanese as those translated from English.

In our preliminary study, we collected and analyzed 100,000 tweets using these common keywords (e.g., "attack" and "email,") and found that more than 95% of the tweets were unrelated to phishing attacks. On the other hand, we also found that most tweets related to phishing attacks contained 20 selected security keywords. Specifically, 4,921 tweets, or 4.92% of the 100,000 tweets mentioned above, contained information about phishing attacks that had one of *Security Keywords*. Therefore, the security keywords selected in this study are reasonable for collecting and analyzing as many reports of phishing attacks as possible from many tweets on Twitter while reducing the number of false positives.

Co-occurrence Keywords. *Co-occurrence Keywords* in this paper are not directly security-related keywords, but keywords (e.g., Amazon and ATT) that co-occur with *Security Keywords* at certain times and are included in non-expert tweets. The purpose of designing Co-occurrence Keywords is to collect as many phishing report attacks as possible that would otherwise be missed by Security Keywords. Specifically, *Co-occurrence Keywords* are extracted using the following procedure. First, we consider the tweets collected during the last period when the system is running as the *Co-occurrence Keywords* extraction target. The strength of association (SoA) is then calculated using the idea of pointwise mutual information (PMI). We define P(X) and P(Y) as the probability of the occurrence of a proper noun X and a type of tweet Y, respectively, in a given tweet. The probability that X and Y co-occur is P(X, Y). In this case, PMI is represented by the following:

$$PMI(X,Y) = \log(\frac{P(X,Y)}{P(X)P(Y)}) \tag{5.1}$$

Next, we use positive pointwise mutual information (PPMI) as in the following equation to avoid the case where PMI goes to negative infinity (i.e., where P(X,Y) = 0).

$$PPMI(X,Y) = \max(0, PMI(X,Y)) \tag{5.2}$$

If X and Y do not occur at all in a single tweet, the PPMI will be 0. If X and Y are likely to occur in a single tweet, the PPMI will be positive or negative. Then, given a pair of proper nouns W in a tweet and a binary label L in the tweet (i.e., a phishing report or non-report), the SoA is given by the following equation:

$$SoA(W,L) = PPMI(W,L) - PPMI(W,\neg L) \tag{5.3}$$

If W appears only in phishing reports or non-reports, $PPMI(W,\neg L)$ is zero, then SoA is equal to PPMI ($SoA(W,L) = PPMI(W,L)$). Furthermore, W, which appears frequently in phishing and non-reports, has $PPMI(W,L)$ and $PPMI(W,\neg L)$ almost equal. As a result, $SoA(W,L)$ takes on a value close to zero. In other words, given a proper noun in a tweet for a given time period and a binary label of a phishing report or not, it is possible to extract keywords that are frequently found only in the user's report for that time period. Since the common duration of the same phishing attack is 21 hours [115], we calculate the PMI for tweets within the previous 21 hours in our study. For the proper noun extraction task, we use the English model [116] and the Japanese model [117], which have been pre-trained on a large amount of data and confirmed to be highly accurate for this task. We evaluated whether we could extract as many brand names (e.g., Amazon, ATT, Microsoft 365) as possible from the aforementioned 100,000 tweets, and finally set the SoA threshold to 4. Then, the top 10 keywords that exceed the threshold are selected as *Co-occurrence Keywords*. The default state is no *Co-occurrence Keywords*, and *Co-occurrence Keywords* will be selected each time this step is performed.

5.3.2 Extracting URLs and Domain Names

This step extracts URLs and domain names potentially associated with phishing attacks from the collected tweets. The extraction targets include both the texts and images contained in the tweets.

Image Analysis. We extract URLs and domain names from the images in the collected tweets by identifying the body area of the SMS or email. Specifically, we used YOLOv5 [118] as in the previous study [105], to identify body text areas in email or SMS screenshots, annotated with 3,000 images in the dataset described in Section 5.4.3. For the 3,000 images used for training, we analyzed valid thresholds with confidence scores ranging from 0.0 to 1.0 for the body text areas extracted by YOLOv5. As a result, all areas with a confidence score of 0.8 or higher corresponded to the body text area in the image. Therefore, in this study, if YOLOv5 extracts an area with a confidence score of 0.8 or higher, it is considered to be the body text area. Then, we use Tesseract [119] to extract character strings from the body text areas in both English and Japanese. If the body text area is not identified, we apply Tesseract to the entire image. We extracted text from English tweets using models pre-trained in English, while we extracted text from Japanese tweets using models pre-trained in both English and Japanese. This is because Japanese phishing emails/SMSs also contain English words.

Text Analysis. Next, we extract URLs and domain names from the text of images and tweets. Our study focuses on URLs and domain names that non-experts are likely to post as phishing attack information. Using regular expressions, we retrieved only the matches of URLs and domain names as candidate phishing sites from both the text of tweets related to the reports and the text derived from images. In particular, if there are defanged strings (e.g., example.com to example[.]com and http to hXXp) in a text, we refang the text (e.g., example[.]com to example.com and hXXp to http) and extract the URL and domain name matched by the regular expression.

Screening Phishing-related URLs and Domain Names. Finally, we exclude URLs and domain names that are incorrectly formatted or related to legitimate sites. Specifically, we check that it conforms to the format specified by RFC 3986 [120] and RFC 1035 [121]. If the URL or domain name that passed format validation is not included in both the image and the text, the tweet will be excluded from further analysis.

Then, we also exclude as legitimate sites any domain name in the top 10,000 on the Tranco list [122] and that does not match the shortened URL list [123]. Existing research [115] has shown that the registration of a domain name and the execution of a phishing attack can occur within a few days or tens of days at most. Therefore, we obtain domain name information from WHOIS and eliminate legitimate sites registered more than 365 days ago. CrowdCanary focuses on fresher domain names to detect newer phishing attacks, thus phishing sites that are more than one year old are excluded from our study. Furthermore, we exclude from our analysis false reports due to attackers sharing obviously legitimate sites or users mistakenly sharing legitimate sites. We output any tweets with at least one or more domain names that remain after the screening as *screened tweets*.

5.4 Proposed System: Reports Classification

We describe the second component of CrowdCanary, *Reports Classification*, in this section. For the screened tweets obtained in the first component, we extract features in the tweets. Using supervised learning, we train a classifier to identify highly relevant reports of phishing attacks with high accuracy. From the created features, we select some features for training to achieve highly accurate and efficient classification. This step includes the following steps: *Feature Engineering*, *Training and Classification*, and *Evaluation of Classification Accuracy*.

5.4.1 Feature Engineering

We extract features from the screened tweets that help us identify user reports. This component classifies a single tweet as either a phishing report or a non-report. Specifically, we generated vectorizable features from Twitter user information, tweet body text, and images. Then, we selected helpful features from the generated features that improve the classification accuracy of phishing reports and non-reports using Boruta SHAP [126]. Boruta SHAP is a method that uses Shapley values for feature selection in Boruta, allowing for more accurate calculation of feature contributions and increasing the robustness of the Boruta

Table 5.2: List of Features

Feature Type	No.	Features Name	Vector Type	Dimensions
Content	1	# of characters	Integer	1
	2	# of words	Integer	1
	3	# of hashtags	Integer	1
	4	# of images	Integer	1
	5	Defanged type	Category	9
URL	6	Total # of characters	Integer	1
	7	# of characters in domain name	Integer	1
	8	# of digits	Integer	1
	9	Top level domain	Category	10
OCR	10	Number of characters	Integer	1
	11	# of words	Integer	1
	12	# of symbols	Integer	1
	13	# of digits	Integer	1
Visual	14	EfficientNet Vector [124]	Embedding	16
Context	15	BERT Vector [125]	Embedding	58
Total				104

algorithm [127]. Finally, we use the five types of features shown in Table 5.2: *Content Features*, *URL Features*, *OCR Features*, *Visual Features* and *Context Features*.

Content Features. From the content of the tweets collected in the previous component, we extract features relevant to identifying sharing related to phishing attacks, focusing mainly on the text. Our idea is straightforward: identify the actual content of the user's tweet. We extract five features from the information in a user's tweet. Specifically, we designed the following six types: number of characters (No. 1), number of words (No. 2), number of hashtags (No. 3), number of images (No. 4), and defanged type (No. 5).

Features No. 1 to No. 4 are each a vector of integer values obtained from tweets. Defanged type (No. 5) is a 9-dimensional feature vector with the one-hot encoding of 9 types of defanged types ("example .com", "example[.]com", "example(.)com", "example{.}com", "example\.com", "hxxp://example.com", "hXXp://example.com", "http[:]//example.com" and "http://example.com[/]"). We believe that the number of characters and words in a warning-only post is relatively small. In addition, when users post reports, they often include numerous screenshots of emails and SMSs, and these features can efficiently identify user reports. Related studies [103, 128] have shown that these similar features can effectively determine whether a string contains warning information.

URL Features. We extract phishing site-specific features from the URLs contained in the texts and images of the screened tweets. Phishing sites often include characteristic strings in the domain name or path portion of the URL (e.g., abuse of subdomain names and long domain names) compared to legitimate sites [90]. It is possible to classify whether URLs are associated with phishing attacks by capturing the differences between the strings in the URLs of phishing sites and legitimate sites. Specifically, we designed the following four types: total number of characters (No. 6), number of characters in the domain name (No. 7), number of digits (No. 8) and top level domain (TLD) (No. 9).

No. 6 to No. 8 are the respective vectors of integer values calculated from the URLs (domain names) contained in the texts or images of the tweets. We conducted a preliminary survey of the TLDs in the ground-truth dataset (Section 5.4.3) and found 841 different TLDs. We investigated whether TLDs contribute to the identification of phishing sites using Boruta SHAP and identified

10 TLDs (“com”, “org”, “top”, “info”, “xyz”, “online”, “net”, “shop”, “cn” and “vip”) as important. TLD (No. 10) is a 10-dimensional feature vector with the one-hot encoding of 10 types of TLD, as mentioned above. For example, the fully qualified domain names (FQDNs) of phishing sites have more characters than those of legitimate sites, indicating subdomain abuse (e.g., login.security.account.example.com). In addition, Spamhaus reports that in 2023, TLDs such as “cn” and “top” have many cases of abuse [129] and may not be reviewed by registrars. As a result, TLDs abused by phishing sites tend to cluster in the same TLD.

OCR Features. We use Tesseract [119] to extract texts from the images in screened tweets. Reports of phishing attacks shared by people in images are typically screenshots of people’s smartphones, significantly different from other images commonly posted on Twitter. We can determine if the images in the tweets are related to the report of a phishing attack by performing OCR on the images and capturing differences in the extracted strings. If there is no image in a tweet, all OCR features are set to 0. If a tweet has multiple images, split it, create OCR features for each image, and classify all split tweets using the same other features.

Specifically, we designed the following four types: number of characters (No. 10), number of words (No. 11), number of symbols (e.g., !, ? and &) (No. 12) and number of digits (No. 13). No. 10 to No. 13 are the respective integer vectors calculated from the texts extracted by applying OCR to the tweet images. In addition to the URL and domain name, the image that the user shares as a phishing report includes the email or SMS text. In other words, texts and words that deceive users into clicking on URLs are also included in the extracted strings. Phishing SMSs and emails that deceive people have a predetermined amount of characters in a similar context (e.g., Your account has been suspended! Verify now [URL]), and hence the features differ significantly from strings extracted from other images.

Visual Features. We construct a fixed dimensional feature vector if the tweets obtained in the previous component contain images. Then, if there is no image in a tweet, the visual features vectors are set to 0. If a tweet has multiple images, split it, create visual features for each image, and classify all split tweets using the same other features. This feature captures the similarity in appearance of common phishing emails and SMSs.

Specifically, because emails, SMSs, and browser screenshots are usually images with a specific appearance, this feature is useful for classifying such images from other images. These images are essential for distinguishing phishing reports from non-reports, as they are included when users post information in the form of images. We use EfficientNet [124] as our visual feature generation model. We selected EfficientNet as the model for generating visual features since it is one of the state-of-the-art methods in image classification [130, 131]. We fine-tuned the model pre-trained on ImageNet (EfficientNet model) in English and Japanese with images related to the report (e.g., phishing email images and SMS phishing images) and images unrelated to the report (e.g., food images and landscape images). We successfully improved the feature generation to decide whether or not to include images related to the report.

We generate a 1,280-dimensional image feature vector from tweets using a retrained model. Then, we compressed the dimensions to achieve a cumulative contribution rate of 99% using TruncatedSVD [132], and the result was 16 dimensions for both English and Japanese. Here, we employ a fixed-dimensional vector, a compressed version of the vector created by the optimized EfficientNet model (No. 14).

Context Features. The contextual information from the tweet sentences obtained in the previous component is represented as a fixed-dimensional feature vector. When people share reports of phishing attacks, they often include alarming and angry statements, and are usually in a specific context. We cannot adequately capture these contexts based on the number of characters or words in a tweet. To this end, we use vectors created by a model trained on a large amount of text to capture the context of a tweet's text.

Specifically, we use BERT [125] as the context feature generation model. BERT and BERT-based methods are state-of-the-art for several natural language processing tasks [133, 134, 135]. We fine-tuned the sentences of tweets related to reports in both English and Japanese using the ground-truth dataset (Section 5.4.3). We optimized feature generation for a pre-trained model with many words to determine whether a tweet is related to user reports or not. In certain scenarios, a user who receives a phishing attack alerts, suspects, or incites the attacker. As a result, the contextual characteristics are different from other people's daily posts.

We create a 768-dimensional context feature vector from tweets using a re-trained model. Then we also compressed the dimensions to achieve a cumulative contribution rate of 99% using TruncatedSVD [132], and the result was 58 dimensions for both English and Japanese. Here, we use a fixed-dimensional vector, a compressed version of the vector generated by the optimized BERT model (No. 15).

5.4.2 Training and Classification

Using the many features we have created so far, we train a model for binary classification of whether a tweet is a report of a phishing attack or not.

Method. Given labeled positive or negative training data, a supervised learning model can be trained that uses the characteristics of each tweet to predict the binary value of tweets associated with phishing reports or non-reports. We then aim to predict with a high degree of accuracy whether new tweets are similar to previous phishing reports or non-reports. We compared and evaluated eight commonly used supervised learning algorithms: Random Forest, Neural Network, Decision Tree, Support Vector Machine, Logistic Regression, Naïve Bayes, Gradient Boosting, and Stochastic Gradient Descent. To account for the influence of some algorithms on accuracy loss, all feature vectors were preprocessed to set the mean to 0 and the variance to 1. Here, we train and evaluate using a ground-truth dataset labeled with phishing or non-phishing reports, which will be explained later in Section 5.4.3.

Results. We adopted Random Forest as the training and classification algorithm for the following three reasons. (1) Random Forest showed the best binary classification accuracy for the ground-truth data among the eight algorithms. (2) Random Forest performed consistently well with stable speed in both the training and inference phases for large amounts of data. (3) The importance of the features in the Random Forest was distributed among *Content Features*, *URL Features*, *OCR Features*, *Visual Features*, and *Context Features*, thus the classifier does not depend on any particular feature in its decision. We perform a classification accuracy evaluation on the ground-truth datasets (Section 5.4.3) and, in the live operation using CrowdCanary (Section 5.5), a model trained with the Random Forest algorithm, to perform the binary classification of phishing reports and non-reports.

Table 5.3: Ground-truth Dataset for Evaluating the Accuracy of Machine Learning Models

Language	Collected Time	Label	# of Tweets
English	May. 1, 2021 – Jul. 19, 2021	Phishing Reports	5,000
	(80 days)	Non-Reports	15,000
Japanese	May. 1, 2021 – Jul. 19, 2021	Phishing Reports	5,000
	(80 days)	Non-Reports	15,000

Table 5.4: Classification Accuracy Evaluation Results

Language	Features	Accuracy	TPR	TNR	Precision	F-measure
English	**All Features**	**0.957**	**0.952**	**0.962**	**0.962**	**0.957**
	Content+URL+OCR	0.838	0.829	0.847	0.845	0.837
Japanese	**All Features**	**0.949**	**0.948**	**0.960**	**0.951**	**0.943**
	Content+URL+OCR	0.798	0.754	0.843	0.827	0.789

5.4.3 Evaluation of Classification Accuracy

Before taking measurements with CrowdCanary in live operation, we evaluated the classification accuracy of phishing reports and non-reports in CrowdCanary.
Ground-truth Datasets. Table 5.3 shows the dataset used for the evaluation. First, we used the 20 English keywords from Table 5.1 and the 20 translated Japanese keywords. Then, we searched on Twitter using the keywords for 80 days from May. 1, 2021 – Jul. 19, 2021, and collected 1,543,245 and 1,023,368 tweets in English and Japanese, respectively. Existing studies or publicly available datasets do not provide ground-truth datasets for the correct answers to phishing reports and non-reports, which are our research goals. As a result, we have to annotate them ourselves. Therefore, we randomly sampled the collected tweets and manually labeled them with a binary value of either phishing reports or non-reports. We excluded from our annotations tweets that do not have a URL or domain name in the text or image of the tweet. We then accessed the URLs and domain names in the text and images of the collected tweets from the experimental environment, examined the collected web content, and performed a similarity analysis with legitimate sites. Four security engineers conducted this annotation, and we labeled each of the tweets that we all agreed were reports of phishing attacks and non-reports. As a result of the annotations, we labeled the tweets as “phishing reports” when we determined they were related to phishing attacks and “non-reports” when they were not. Finally, we created 5,000 “phishing reports” and 15,000 “non-reports” in English and Japanese, respectively. To account for the effect of temporal bias, we split the training and testing data 7:3 in time order for the evaluation experiment.
Evaluation Results. The evaluation results are shown in Table 5.4. When combining all features (Content+URL+OCR+Visual+Context) for the English case, Accuracy was 0.957, True Positive Rate (TPR) was 0.952, True Negative Rate (TNR) was 0.962, Precision was 0.962, and F-measure was 0.957. The results show that the accuracy is sufficient to classify phishing reports from the large volume of tweets collected. We also found that it is difficult to detect user reports of phishing attacks with high accuracy using only simple features generated from meta information on Twitter. The same result is obtained for the Japanese case. We conclude that feature vectors with information embedded in a fixed dimension, pre-trained on many languages and images, significantly improve classification accuracy. To summarize, in subsequent evaluations for Section 5.5,

Table 5.5: Overview of Datasets for Evaluation

System	Period	Datasets	#
CrowdCanary	Nov. 1, 2022 – Jan. 31, 2023 (3 months)	Collected Tweets	18,765,699
		Screened Tweets	324,589
		Phishing Reports	38,935
		Detected Threats	42,987
		Detected URLs	35,432
SpamHunter	Jan. 1, 2018 – Aug. 31, 2022 (56 months)	Detected Threats	15,553
		Detected URLs	15,269
Twitter IOC Hunter	Aug. 1, 2021 – Jul. 31, 2022 (12 months)	Detected Threats	10,092
		Detected URLs	9,344

we will use a machine learning model trained by combining five types of features: Content+URL+OCR+Visual+Context.

5.5 Evaluating User Reports in the Wild

We used CrowdCanary, which was confirmed to detect user reports with high accuracy in Section 5.4.3, to classify unknown tweets in the wild. We then performed a comparative evaluation with two existing systems [109, 105] that collect and publish malicious URLs and domain names from Twitter.

5.5.1 Operating Environment

We operated the proposed system in a virtual machine (VM) on Azure. Specifically, we used the Linux OS Ubuntu 20.04 on a Standard D32as v4 (32 vCPU, 128GB Memory) VM. We used twscrape [136], an open source tool, as the means of data retrieval from Twitter, and scikit-learn [137] for the analysis process related to machine learning. We employed luigi [138], a Python-based pipeline framework, to ensure that each task can be efficiently scheduled and processed in all sub-steps. With the operational environment described above, the proposed system operated without error during all the experimental periods in this study.

5.5.2 Datasets for Evaluation

A summary of the datasets for CrowdCanary and the two existing systems for comparison is shown in Table 5.5. These two existing systems collect information from Twitter, but the information they collect is not limited to phishing attacks. Although CrowdCanary focuses specifically on phishing attacks, we demonstrate that the quantity and quality of information collected by CrowdCanary outperforms the two existing systems. While CrowdCanary is a newly implemented system that works perfectly on the current version of Twitter, the existing systems rely heavily on older Twitter APIs and are unable to analyze the latest tweets. Therefore, we used datasets [139, 109] from when these systems were publicly available for our evaluation.
Proposed System (CrowdCanary). We ran CrowdCanary continuously every hour for three months, from Nov. 1, 2022 – Jan. 31, 2023. We set the *Security Keywords* to 20 English and 20 Japanese words in Table 5.1, and the initial state of the *Co-occurrence Keywords* to none. CrowdCanary selected new *Co-occurrence Keywords* every hour from the collected user reports. During the two-month experiments, we collected 18,765,699 tweets, screened 324,589 tweets, and identified 38,935 phishing reports. For domain names included in user reports,

Table 5.6: Overview of Comparison Results between CrowdCanary and Existing Systems

System	VT≧1	VT≧5	Total	VT≧1 /day	VT≧5 /day
CrowdCanary (I+T, E+J)	**31,960 (90.2%)**	**15,768 (44.5%)**	**35,432 (100.0%)**	**347**	**171**
CrowdCanary (I, E+J)	17,633 (84.2%)	7,267 (37.8%)	19,205 (100.0%)	187	83.2
CrowdCanary (T, E+J)	15,260 (88.6%)	8,452 (49.1%)	17,231 (100.0%)	164	88.5
CrowdCanary (I+T, E)	17,126 (91.2%)	7,558 (40.2%)	18,779 (100.0%)	186	82.2
CrowdCanary (I+T, J)	14,834 (89.1%)	8,210 (49.3%)	16,653 (100.0%)	161	89.2
CrowdCanary (I, J)	7,528 (83.4%)	4,107 (45.5%)	9,026 (100.0%)	81.8	44.6
CrowdCanary (T, J)	7,081 (87.5%)	4,014 (49.6%)	8,093 (100.0%)	77.0	43.6
CrowdCanary (I, E)	9,048 (88.9%)	3,583 (35.2%)	10,178 (100.0%)	98.3	38.9
CrowdCanary (T, E)	8,286 (90.8%)	4,244 (46.5%)	9,126 (100.0%)	90.1	46.1
SpamHunter (I, E) [105]	8,266 (59.8%)	1,718 (10.9%)	15,269 (100.0%)	4.85	1.01
Twitter IOC Hunter (T, E) [109]	5,228 (56.0%)	2,172 (23.2%)	9,344 (100.0%)	14.3	5.95

I: Image analysis, T: Text analysis, E: English analysis, J: Japanese analysis

we considered them to be URLs by appending the protocol "https" to the domain name. Finally, we merged these URLs with the extracted URLs to obtain 35,432 unique URLs extracted by CrowdCanary.

Existing System (SpamHunter). We selected the dataset of the previous study [105] as our existing system for comparison. Their "SpamHunter" system collects tweets with SMS-related keywords, performs image analysis, and extracts phishing-related URLs. Spam
Hunter comes closest to our motivation in terms of the information we want to collect, however their method of collecting tweets is very limited. This is because SpamHunter only analyzes tweets when the keyword "sms" is included in the body of the tweet, which results in a large number of non-expert tweets that should be analyzed being missed. They published the collected URLs [139], and obtained 15,553 threats from Jan. 1, 2018 – Aug. 31, 2022. In addition, we added "https" to threats that lacked protocol information, excluded URLs with formatting deficiencies, and finally prepared 15,269 detected URLs.

Existing System (Twitter IOC Hunter). Next, we selected the existing system [109] for comparison because it extracts cybersecurity-related information (e.g., malicious URLs, IP addresses, etc.) from Twitter and allows us to obtain data for a specified time period through its API. We obtained 10,092 threats using the API of Twitter IOC Hunter [109] from Aug. 1, 2021 – Jul. 31, 2022. Similar to SpamHunter, we added "https" to threats that lacked protocol information, excluded URLs with formatting deficiencies, and finally prepared 9,344 detected URLs.

5.5.3 Comparison of Maliciousness using VirusTotal

We analyzed how VirusTotal(VT) [80] flags the URLs detected by CrowdCanary and the two existing systems [105, 109]. When we request VirusTotal to scan a URL, it evaluates the maliciousness of about 90 different types of anti-virus software and returns the results to us. Several studies [140, 141, 103, 142, 143] used VirusTotal as a metric for evaluation. Then it is appropriate for our study to evaluate how much of the information collected from Twitter are actually malicious URLs.

VirusTotal provides five types of results for scanned URLs: malicious, suspicious, harmless, undetected and timeout. Because CrowdCanary immediately collects/outputs phishing attacks shared by Twitter users, sometimes VirusTotal does not detect them even though the URLs are malicious. We then requested scans and obtained results at least one week after detection in CrowdCanary. Since the URLs of the existing systems had already mainly been analyzed by VirusTotal, we obtained the results of the scans. If VirusTotal had no previous scan results, we requested a scan and obtained the scan results. VirusTotal has also seen cases of false positives from anti-virus vendors [141]; therefore, URLs identified as malicious/suspicious by one anti-virus vendor are not necessarily phishing URLs. As a result, in our study, we compared CrowdCanary and the two existing systems in terms of the number of URLs flagged as malicious/suspicious by at least one and as many as five anti-virus vendors in VirusTotal.

The comparison results are shown in Table 5.6. We conducted the analysis with images and text as the threat information extraction targets, and with English and Japanese as the tweet collection languages. Focusing on URLs that were flagged as positive by five or more antiviruses in VirusTotal, 15,768 (44.5%) were positive for CrowdCanary (Image+Text), 7,267 (37.8%) were positive for CrowdCanary (Only Image), 8,452 (49.1%) were positive for CrowdCanary (Only Text), 1,718 (10.9%) were positive for SpamHunter and 2,172 (23.2%) were positive for Twitter IOC Hunter. We confirmed that CrowdCanary was superior to the proposed and existing systems in terms of both the absolute number and detection rate of URLs later detected by multiple antiviruses in VirusTotal. Early detection of URLs that will later be detected by VirusTotal is important for the future development of countermeasure technology. SpamHunter is a system that extracts information from tweet images, and Twitter IOC Hunter is a system that extracts threats from tweet texts. Even if we target only images and texts for CrowdCanary's threat extraction, we can see that it can extract more URLs detected by VirusTotal. Due to the different experimental periods of the proposed system and the two existing systems, we compared the average per day of URLs detected by VirusTotal. In this case as well, the results showed that CrowdCanary was superior to the existing systems. When CrowdCanary's threat information extraction is limited to images and English (equivalent to SpamHunter's analysis target with CrowdCanary (I, E) in Table 5.6), CrowdCanary extracted 20 times and 39 times the amount of malicious URLs for VT$\geqq$1/day and VT$\geqq$5/day, respectively, compared to SpamHunter. Additionally, when CrowdCanary's threat information extraction is limited to texts and English (equivalent to Twitter IOC Hunte's analysis target with CrowdCanary (T, E) in Table 5.6), CrowdCanary extracted 6 times and 8 times the amount of malicious URLs for VT$\geqq$1/day and VT$\geqq$5/day, respectively, compared to Twitter IOC Hunter.

In addition, we manually investigated the remaining 3,472 (35,432-31,960) URLs that VirusTotal did not detect during the experimental period. We identified malicious URLs that could be identified as phishing sites based on the content of tweets, website content, screenshots, WHOIS information, etc. As in Section 5.4.3, this investigation was conducted by four security engineers and took a total of 30 hours to check for undetected URLs in VirusTotal. As a result, we found that 2,635 (7.44%) URLs were truly phishing sites (false negatives by Virus-

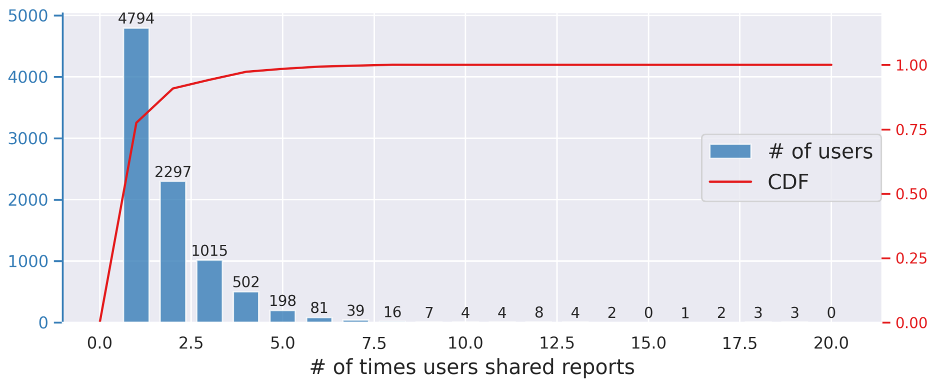


Figure 5.3: Correlation between Users and Number of Times Reports Were Shared

Table 5.7: User Categorization Results

User Type	# Users	# Reports	# Shared min	median	mean	max
Expert	25	15,263	1	280	610	3,900
Non-expert	9,000	17,577	1	1	1.95	73

Total). Most of these URLs were used for redirects under the domain names of duckdns.org, which abused the dynamic DNS provider, and cutt.ly, which abused the URL shortening service and made it difficult to determine the maliciousness of the URLs mechanically. On the other hand, 482 (1.36%) URLs were incorrect information due to OCR misidentification (e.g., misidentifying “l” as “1”), 160 (0.45%) URLs were not phishing site URLs included in the user’s report (e.g., minor legitimate sites that users cannot accurately determine whether they are phishing or not), and 195 (0.56%) URLs were misclassified by the machine learning model (e.g., legitimate SMSs or emails). The next Section 5.6 analyzes a total of 34,595 (31,960+2,635) URLs detected by VT or manually identified as malicious URLs.

5.6 Comparison of Experts and Non-experts

We analyze the reports collected by CrowdCanary with a focus on the characteristics of the users (i.e., security experts or non-experts). In this section, we use 34,595 URLs (32,813 phishing reports) containing malicious information related to phishing attacks identified by VirusTotal and manual investigation in Section 5.5.3.

5.6.1 Analysis of Users who Shared Reports

Of the 32,813 phishing reports, the number of unique users was 9,025. We identify the users who shared these reports as experts or non-experts. Specifically, users who satisfy either of the following two conditions are considered experts, and users who satisfy neither of the two conditions are considered non-experts. (1) The user has security-related keywords (e.g., phishing, threat hunter) in their Twitter profile. (2) The user has posted more than half of their last 10 tweets related to cybersecurity.

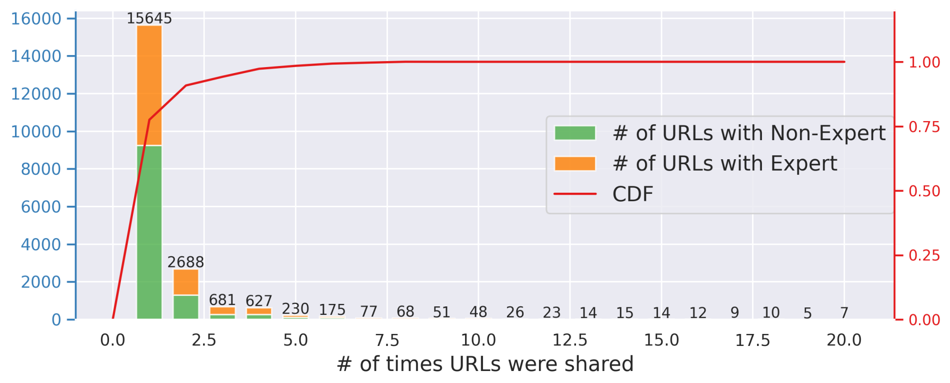


Figure 5.4: Correlation between Users Types and Number of Times URLs Were Shared

Table 5.8: Comparison of URLs Characteristics

User Type	# URLs	# Shortened URLs	# Services
Expert	16,778	102 (0.61%)	7
Non-expert	18,654	2,896 (15.5%)	13

As a result, we categorized users in the method described above, resulting in 25 users (2.77%) as experts and 9,000 users (97.23%) as non-experts, as shown in Table 5.7. We reviewed the results as a manual and verified that they were categorized as intended. We found that experts share phishing reports an average of 610 times, while non-experts share phishing reports an average of 1.95 times. In particular, we confirmed that many expert shares appeared to be mechanical, with some accounts only posting phishing attack threats up to 3,900 times during the experimental period. Most non-experts shared phishing emails and SMS messages they received only a few times. However, in rare cases, we found some non-experts who shared phishing emails and SMS messages they received 73 times during the experimental period.

Additionally, Figure 5.3 shows the correlation between users and the number of times reports are shared. The x-axis represents the number of times a user shared a report, the blue bar on the y-axis represents the number of reports based on the number of times the report was shared, and the red line on the y-axis represents the cumulative distribution function (CDF) value of the reports. From Figure 5.3, users who shared only one report accounted for 53.1% of the total, while users who shared two reports accounted for 78.6% of the total. In other words, if we collect information from Twitter limited to accounts of users who frequently share, as in existing studies [102, 103], we would miss phishing reports from numerous users. We demonstrated that CrowdCanary can collect not only the limited information shared by security experts, but also information posted by a large number of users, including reports of phishing attacks by non-experts.

5.6.2 Analysis of the Detected URLs' Characteristics

We analyzed the value of the URLs included in the phishing reports. Specifically, we analyzed the number of times each URL was shared as a phishing report. The correlation between user types (i.e., experts or non-experts) and the number of

Table 5.9: Comparison of FQDNs Characteristics

User Type	# FQDNs	# Dynamic DNSs	# Providers
Expert	6,530	668 (10.2%)	1
Non-expert	8,699	3,612 (41.5%)	3

Table 5.10: Comparison of Report Sharing Methods

User Type	# URLs in Images	# URLs in Texts	#Hashtags median	#Hashtags mean	#Mentions median	#Mentions mean
Expert	1,523 (10.0%)	13,740 (90.0%)	4	3.83	0	0.21
Non-expert	16,659 (94.8%)	918 (5.22%)	0	0.73	0	0.15

times reports containing that URL were shared is shown in Figure 5.4. The x-axis represents the number of times a URL has been shared, the green and orange bars on the y-axis represent the number of URLs found that match, and the red line represents the CDF value of the unique URLs. From Figure 5.4, URLs extracted from phishing reports shared only once by users accounted for 77.5% of the total, while URLs extracted from user reports shared twice by users accounted for 90.8%. As shown in our results, we found that extracting information from the tweets of a fixed set of users with a limited observation target would miss the majority of high-value malicious URLs that are shared a few times at most.

We then analyze the characteristics of the URLs and FQDNs shared by security experts and non-experts. The results are shown in Tables 5.8 and 5.9. The unique URLs included in the expert and non-expert reports were 16,778 and 18,654, respectively. Attackers sometimes use redirects from the landing URL to the phishing site where they ultimately want to direct the user [115, 144]. Specifically, we investigated how many URLs exploited the dynamic DNS providers [145] and URL shortening services [123] used to redirect phishing attacks. Among dynamic DNS providers, duckdns.org was found to be abused 99.3% in total, and among URL shortening services, cutt.ly and bit.ly were abused 70.5% in total. Because these services and providers are free, can generate a large number of URLs, and have no countermeasures to exploit for phishing attacks, it is believed that attackers use them to evade detection (i.e., spam emails and SMSs detection) of phishing sites they have created. Many of the threats shared by non-experts are URLs that are actually spread in phishing e-mails and SMSs. These URLs can be used as a starting point for analyzing the full picture of attacks, or as intelligence for block lists that automatically detect spam e-mails and SMSs. Many experts share URLs after redirects, which sometimes cannot be analyzed because they are inaccessible without the proper referrer [146], and are not suitable information to prevent the spread of phishing emails and SMSs.

5.6.3 Analysis of Report Sharing Methods

We analyze the differences in the way experts and non-experts share information. First, we compared experts and non-experts on how users share information about phishing attacks. The results are shown in Table 5.10. We found a significant difference in how information was shared: 90% of expert reports included URL information in the text of their tweets. In contrast, 95% of non-expert reports included URL information in the images of their tweets. Experts identify threats through their own investigation rather than by encountering them, and they often share the information in a formatted text (in the text of a tweet). On

Table 5.11: Top 10 Keywords Collected Phishing Reports

Rank	Keywords (Expert)	Type	#	Keywords (Non-expert)	Type	#
1	#phishing (E+J)	S	3,982	#phishing (E+J)	S	1,545
2	#scam (E+J)	S	2,733	National Tax Agency (J)	C	1,175
3	#phishingmail (J)	C	1,406	Fraud (J)	S	1,035
4	#infosec (E)	S	1,283	#Amazon (E+J)	C	889
5	#cybersecurity (E)	S	1,280	#scam (E+J)	C	820
6	#Amazon (E+J)	C	1,119	Softbank (J)	C	712
7	#phishingsite (J)	C	1,079	Docomo (J)	C	688
8	National Tax Agency (J)	C	1,022	American Express (E+J)	C	653
9	SMBC (J)	C	894	Google (E+J)	C	512
10	#bank (E)	C	822	Please retweet (J)	C	488

E: English only, J: Japanese only, E+J: Contains identical semantic words in both languages
S: Security Keywords, C: Co-occurrence Keywords

the other hand, non-experts often store the phishing attacks they encounter it (receiving an email or SMS, or reaching the site with a browser) as screenshots from their smartphones, etc., and attach the images directly to their tweets and share them. Although it is difficult to collect a large number of these reports from non-experts and extract information properly, CrowdCanary was able to extract as many threats as experts and more, as shown in Figure 5.4.

We also found significant differences in features between experts and non-experts in the context of the text when sharing reports. The median and the mean number of hashtags and mentions in the phishing reports of experts and non-experts are shown in Table 5.10. Hashtags are referred to as "#phishing" and are primarily used by users on Twitter to share information. People looking for information can find tweets containing the hashtag relatively easily using the search function. In this case, the expert report shows an average of 3.83 hashtags in the tweets, while the non-expert report shows an average of only 0.73 hashtags. As a result, collecting non-expert reports with appropriate keywords is more difficult than collecting expert reports shared using fixed hashtags. Similarly, we examined user reports that included mentions that could be posted to a specific user account on Twitter and found no significant differences between experts and non-experts.

Finally, we discuss query keywords that were useful in collecting phishing reports. The top 10 keywords that resulted in the collection of expert and non-expert reports are listed in Table 5.11. Among the top 10 keywords for experts, 8 were hashtagged and 4 were security (as defined in Section 5.3.1) keyword types. In particular, we found that a large number of experts shared their information using the hashtags "#infosec" and "#cybersecurity", which are not commonly used by non-experts. On the other hand, only 3 of the top 10 non-expert keywords were hashtagged. Although "#phishing" was sometimes the most effective keyword for collecting phishing reports, as it was for experts, many of the non-experts shared reports using the name of the company brand that was exploited in the phishing attack. However, simply searching for a company's brand name will return a number of irrelevant tweets. Therefore, either a search using appropriate keywords at the right time, as in this study, or a highly accurate detection mechanism from among the tweets continuously collected by company brand name is required.

Table 5.12: Comparative Dataset of Phishing URLs

Datasets	#
CrowdCanary's Phishing URLs	34,595
OpenPhish's Phishing URLs [99]	82,963
PhishTank's Phishing URLs [98]	28,164
CrowdCanary ∩ OpenPhish	11,589
CrowdCanary ∩ PhishTank	9,213
OpenPhish ∩ PhishTank	12,748
CrowdCanary ∩ OpenPhish ∩ PhishTank	4,620

5.7 Analyzing Phishing Attacks in User Reports

To deepen our understanding of phishing attack reports shared on Twitter, we evaluate the effectiveness of information regarding countermeasure techniques and analyze the actual phishing attack infrastructure.

5.7.1 Analysis of Common URLs with Existing Data Feeds

We collected two types of datasets for comparative evaluation, OpenPhish [99] and PhishTank [98], both specialized for phishing attacks. OpenPhish is an open feed of large-scale data on phishing, and various existing countermeasure technologies reference the OpenPhish dataset. PhishTank is a crowdsourcing service that stores phishing data from users via URL submission and phishing verification. PhishTank determines whether a URL submitted by one user is phishing or not depending on the criteria of other users, and if the URL exceeds a specified PhishTank criterion, it is classified as a phishing site. In this comparative evaluation, we used only data of PhishTank labeled as phishing sites.

These data feeds are widely used in existing researches [147, 115, 128, 142] for the evaluation of phishing attacks as open threat intelligence that anyone can use. The two data feeds are explicitly indicated as providing information to APWG [148] and national CSIRTs; thus, when phishing URLs are published in the data feeds, the CSIRT in each country moves to handle takedowns of them. We continuously collected the latest data feeds from OpenPhish and PhishTank hourly during the same three-month period of Nov. 1, 2022 – Jan. 31, 2023. As a consequence, we collected 82,963 and 28,164 URLs from OpenPhish and PhishTank, respectively.

We evaluated the ratio of common phishing URLs and the latency of the same URLs across two data feeds using CrowdCanary. The target time for evaluation is the posting time of the extracted user reports on Twitter for CrowdCanary, the discover_time in the data available in the API for OpenPhish, and the verification_time in the data available in the API for PhishTank.

Ratio of Common URLs. As shown in Table 5.12, CrowdCanary's phishing URLs and OpenPhish URLs have 11,589 URLs in common, CrowdCanary's phishing URLs and PhishTank URLs have 9,213 URLs in common, OpenPhish and PhishTank URLs have 12,748 URLs in common, and all three types of data have 4,620 URLs in common. We discovered that less than half of the phishing URLs in each dataset had anything in common, and that the observed targets differed greatly across them. Especially, among the phishing URLs extracted independently by CrowdCanary, only 13.4% of the total URLs were listed in both OpenPhish and PhishTank. This comparison revealed that the phishing URLs extracted by CrowdCanary had a large amount of unique information that was not present in other data feeds. In other words, reports of phishing attacks by

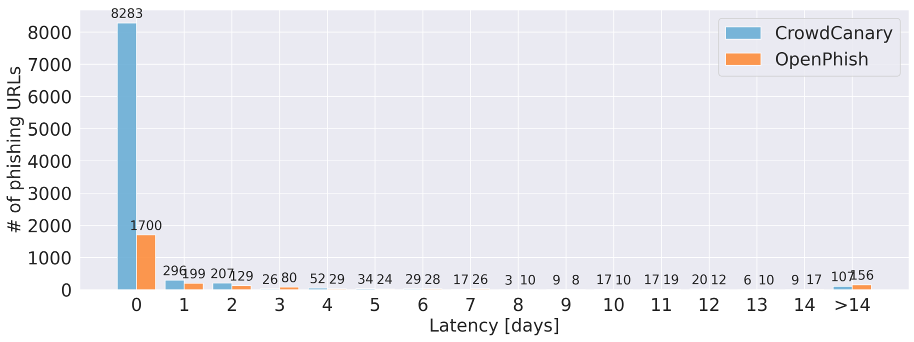


Figure 5.5: Latency Comparison of Phishing URLs in CrowdCanary and OpenPhish

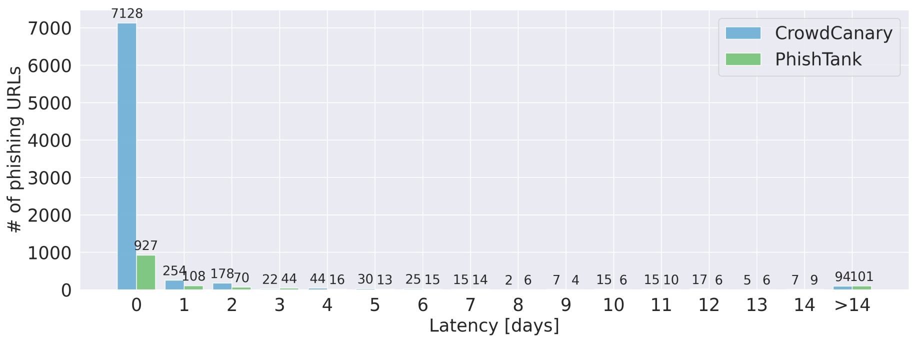


Figure 5.6: Latency Comparison of Phishing URLs in CrowdCanary and PhishTank

users on Twitter are worth analyzing and extracting, and may be utilized as a new data feed for countermeasure techniques in the future.

Latency Comparison with OpenPhish. We compared and evaluated 11,589 phishing URLs common to CrowdCanary and OpenPhish. A summary of the results is shown in Figure 5.5. The x-axis is the difference in latency in days, and the y-axis is the number of relevant phishing URLs. The blue bars represent the number of phishing URLs collected faster by CrowdCanary, whereas the orange bars represent the number of phishing URLs collected faster by OpenPhish. The number of phishing URLs that were collected faster by CrowdCanary was 9,132, which was 78.8% of the total common phishing URLs. From Figure 5.5, most latency differences were less than one day. Because phishing sites have a short survival time [115], it is possible to improve existing countermeasure techniques with information from user reports, as the majority of common phishing URLs were collected earlier by CrowdCanary.

Latency Comparison with PhishTank. We compared and evaluated 9,213 phishing URLs common to CrowdCanary and PhishTank. A summary of the results is shown in Figure 5.6. The x-axis is the difference in latency in days, whereas the y-axis is the number of relevant phishing URLs. The blue bars represent the number of phishing URLs collected faster by CrowdCanary, whereas the green bars represent the number of phishing URLs collected faster by PhishTank. The number of phishing URLs collected faster by CrowdCanary was 7,853, which was 85.3% of the total common phishing URLs. From Figure 5.6, we found that Twitter users more often report information about phishing attacks earlier than security experts who use PhishTank routinely. These results mean more users can be protected from phishing attacks by using CrowdCanary information

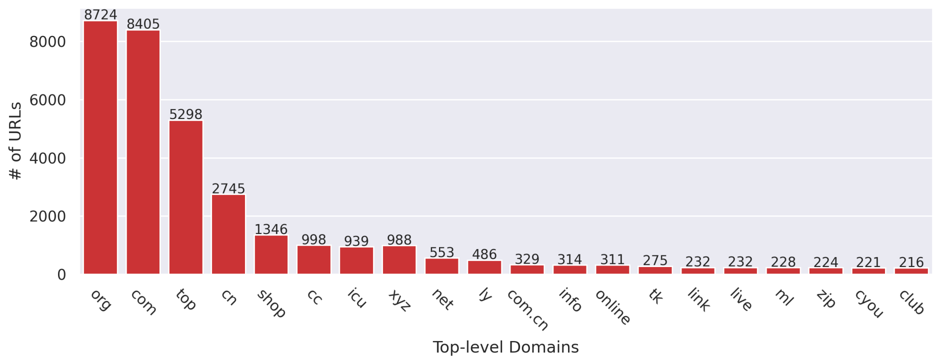


Figure 5.7: Distribution of Top Level Domains

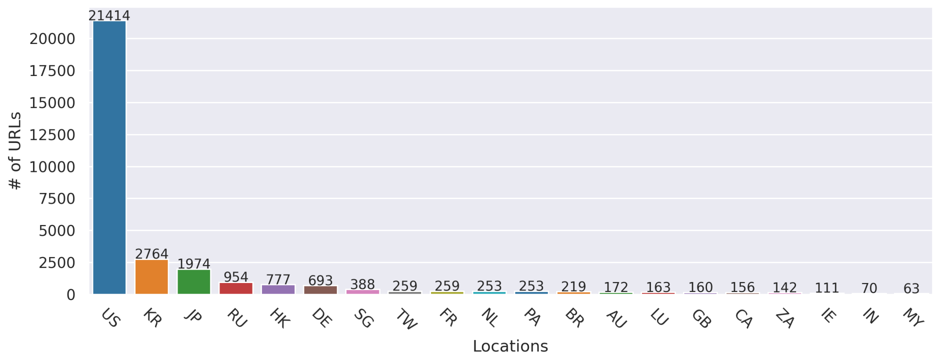


Figure 5.8: Distribution of IP Address Locations

as a countermeasure than by referring to PhishTank to detect phishing attacks.

5.7.2 Analysis of Phishing Infrastructure

We analyzed the structure of website infrastructure commonly used by attackers for phishing sites by extracting URL and domain name information using CrowdCanary. We focused on what trends exist in the web resources (e.g., domain names and web servers) that attackers use to deploy phishing sites.

Distribution of Top Level Domains. We aggregated the top level domains (TLDs) of 34,595 phishing URLs detected by CrowdCanary. The top 20 most frequently occurring TLDs and the number of URLs are shown in Figure 5.7. In total, we found 279 TLDs. As shown, the TLD “org”, which was found in the largest number this time, is often used as a domain name for organizations and associations. In this case study, 8,591 URLs (98.5%) with a top level domain of “org” were found to be exploited by “duckdns.org”, a dynamic DNS provider described in Section 5.6.2. To address this issue, it is essential to collaborate with the operators of these URLs. The top level domain “com” is the most commonly registered domain name and was found to be frequently abused by numerous phishing sites in this survey. The following TLDs, “top”, “cn”, “shop”, “cc”, “icu”, and “xyz”, were reported to have a high rate of abuse [129], and were shared on Twitter as many phishing attack reports as well. There is a clear trend in the TLDs that are exploited in phishing attacks, and this information is valuable in determining maliciousness.

Distribution of IP Addresses Locations. We analyzed the location information of IP addresses by querying the GeoIP2 database [149]. Out of 34,595 phish-

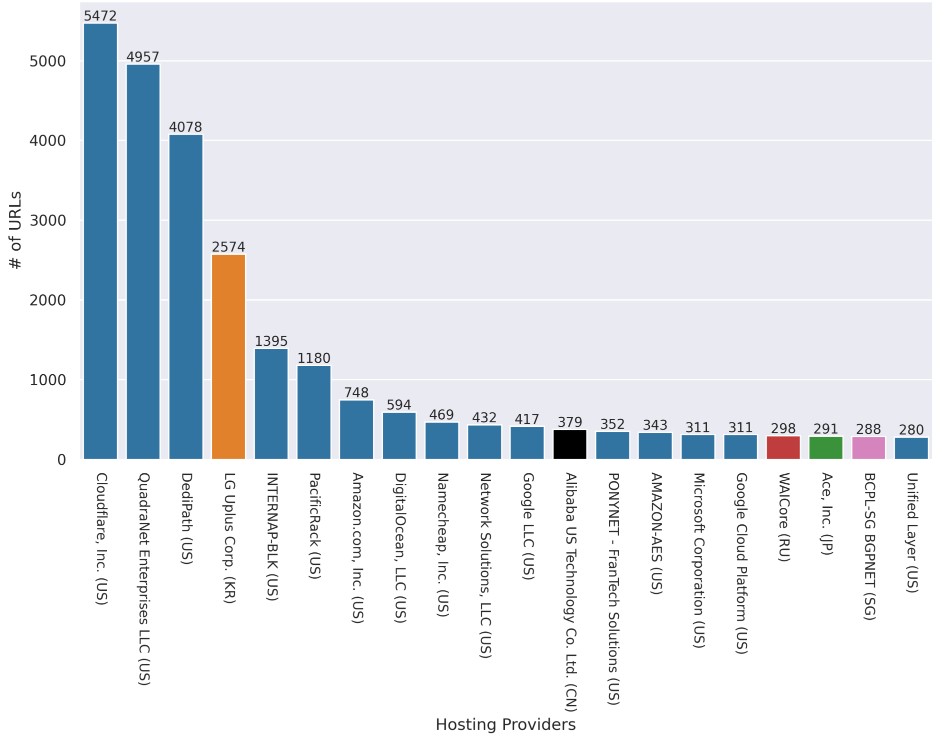


Figure 5.9: Distribution of Hosting Providers

ing URLs, we were able to analyze 31,385 URLs for which we were able to resolve names. Figure 5.8 shows the top 20 country codes that occur most frequently, along with their corresponding number of URLs. In total, there are 63 different country codes, and we found that 68.2% of IP addresses were located in the United States. Although CrowdCanary targeted phishing sites shared between English and Japanese languages for analysis, the top-ranked English-speaking countries were predominantly biased toward the United States. In other words, many phishing site web servers targeting Japanese people were also located in the United States.

Distribution of Hosting Providers. We analyzed the hosting providers that manage the aforementioned range of IP addresses. The top 20 most frequently occurring hosting providers with country codes and the number of URLs are shown in Figure 5.9. Overall, there were 457 different hosting providers, and 15 of the top 20 were managed by United States organizations. Instead of exploiting only specific hosting providers, attackers are deploying phishing sites across a wide range of hosting providers. It is possible to reduce potential victimization by collaborating with these higher-ranking providers and guiding them towards taking down the malicious sites.

Distribution of Frequent IP Addresses and Hosting Providers. We analyzed the number of IP addresses linked to unique URLs and their relevance to hosting providers. We believe that if an attacker deploys a phishing attack using the same IP address, we can find similar attacks when we detect a single URL, even if the domain names are different. The hosting providers linked to the top 20 IP addresses and the number of corresponding URLs are shown in Figure 5.10. We found that the top 20 IP addresses were managed by eight hosting providers. In particular, 1,791 and 1,400 phishing URLs were linked to a single IP address administered by “INTERNAP-BLK (US)” and “LG Uplus Corp (KR)”.

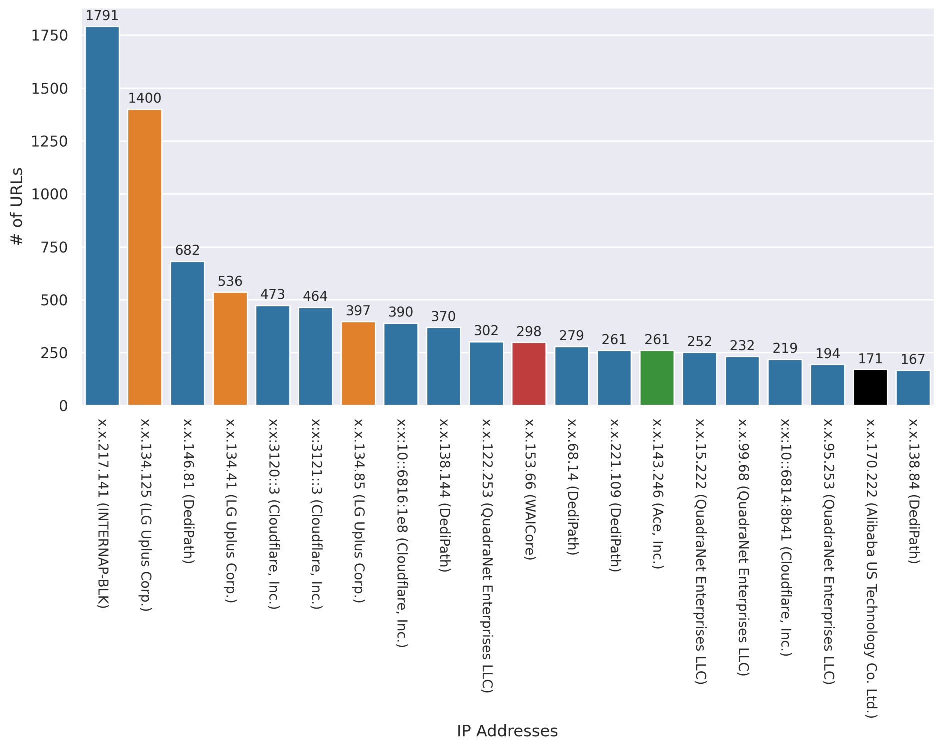


Figure 5.10: Distribution of Frequent IP Addresses and Hosting Providers

The three IP addresses (orange bars) managed by “LG Uplus Corp (KR)” have a total of 2,333 phishing URLs linked to them, indicating that the operators' anti-phishing site measures are insufficient. These results revealed that attackers deployed phishing sites using a variety of domain names, but with a bias toward specific IP addresses. As a criterion for determining whether a site is phishing or not, information such as IP addresses close in range to those already abused, or IP addresses managed by the same hosting provider, may be useful.

5.8 Discussion

We describes the potential for using CrowdCanary output information to defend against phishing attacks, the limitations of CrowdCanary, and ethical considerations of the experimental design.

5.8.1 Utilizing the Intelligence Collected for Phishing Attack Defense

We have demonstrated that CrowdCanary can collect threat intelligence on a large number of phishing attacks with greater accuracy than existing technologies. How can this collected intelligence be applied to actual defensive strategies? We believe that intelligence can be used from two main perspectives.

First, the phishing information collected can add to the intelligence in the block lists. It has been reported that the spread of phishing attacks does not end with the first wave of attacks; the second and third waves of attacks are sometimes spread using the same domain names [150]. By extracting information about the attack as early as possible, such as during the first wave, and feeding it into blocklists (e.g., email spam filters), it may be possible to protect users who may become victims of the second and third waves. It was also reported that among users who receive phishing emails, the average time difference between the timing of the first user to click on the URL and the last user to click on the URL is 21 hours [115]. By sharing information with the browser vendor's block list during this time difference, the browser can warn the user and protect them from phishing attacks if they visit the same URL.

Second, the characteristics of phishing attacks contained in the collected information can be analyzed and used as countermeasure information for similar attacks that may occur in the future. It has been reported that phishing sites change domain names frequently, but may continue to be hosted at a particular IP address [151]. For example, using passive DNS (e.g., Farsight DNSDB [79]), it is possible to detect attacks early using CrowdCanary intelligence if the A record of a newly appearing domain name is linked to the same IP address as a domain name that has been exploited for phishing attacks in the past. In addition, phishing sites created using phishing toolkits often have the identical HTML source, images on the site, and scripts [152]. This information can be useful in techniques such as content-based phishing site detection [153]. In addition, information about phishing attacks received by many users can be used to understand trends in company brands being exploited in attacks and to keep an eye on companies and industries that attackers will be targeting in the future.

5.8.2 Role as a Platform for Threat Information Sharing

As a platform for information sharing, UGC platforms such as Twitter have numerous advantages over closed information reporting channels such as those of government and academic institutions. First, when a new cyber threat arises, information may be shared immediately on UGC platforms. By properly discarding and selecting such information, it is possible to immediately take countermeasures. In other words, the real-time exchange of information inherent in UGC platforms facilitates early response to threats. In addition, because information from users of various characteristics is gathered, the coverage of threat-related information is superior. This will allow the entire community to exchange information on knowledge to counter threats, not just specific organizations or individuals.

However, because it is an open forum, the threat information shared may be misinformed or misinterpreted. Moreover, UGC platforms can be a source of cybersecurity threats themselves, such as phishing attacks and the spread of malware. Users may expose themselves to threats by clicking on the wrong links or downloading from unreliable sources. Given these factors, it is extremely important for users to have appropriate knowledge and the ability to judge whether their information is reliable or not in order to utilize the information on the UGC platform.

Threat reporting channels established by specific organizations or companies have existed for a long time and have contributed significantly to security measures in the past. These channels provide information that has been confirmed and verified by experts and is considered reliable. However, delays in providing and updating information may delay responses to new threats. Therefore, by using threat information on UGC platforms, whose users have recently increased and will continue to increase, as a complementary countermeasure, more robust cybersecurity measures can be taken. It is important for the community to maintain incentives and motivation for users of the UGC platform to continue to share useful threat information in the future.

5.8.3 Limitation

Our study has three limitations.

First, our study does not focus on extracting reports only from information about the final destination of phishing sites that involve user interaction or redirection. For example, some users may only share a screenshot on Twitter with the URL of the final destination after the entry or redirect occurs. CrowdCanary cannot properly extract reports in this case because it has no information about the user's input or redirection behavior on the browser. In particular, CrowdCanary is a system that collects URLs that are the seeds of phishing attacks. CrowdCanary does not focus on phishing attacks that do not redirect without an acceptable referrer or can only be reached by clicking. These attacks can be handled by crawling URLs extracted by CrowdCanary as seeds in existing researches [140, 147].

Second, the features designed in this paper are chosen to be invariant with respect to user reporting of phishing attacks. However, the system's accuracy will inevitably decrease over time, and the system will need to be relearned each time. For example, when trends in the appearance of shared images change, or when phishing sites that look completely different become trending, it is necessary to reannotate and relearn the classification model. Since we do not believe that the way users share information on Twitter itself will change significantly, and accuracy will not drop significantly immediately, evaluating how much accuracy will decrease as trends change is one of the issues we will address in the future.

Finally, depending on recent Twitter specification changes [154], equivalent information may no longer be available via the API. Since CrowdCanary is a system that extracts phishing attacks based on user reports, if the number of users using Twitter decreases (i.e., fewer users share phishing reports), the number of candidate phishing reports will decrease as tweets to be analyzed. Therefore, both the quantity and quality of threat information related to phishing attacks extracted by the proposed system will inevitably decline. Any social networking service that allows users to post photos and text, as popular as Twitter, can be used as a source of threat intelligence in the same way. In addition, CrowdCanary is adaptable to changes in Twitter because it was designed based on the characteristics of users sharing information about phishing attacks, rather than using Twitter-dependent features.

5.8.4 Ethical Consideration

We took into account the ethical considerations of collecting data from Twitter on a large scale. Although the collection and analysis targets contain massive amounts of information about Twitter accounts, the content of their tweets is public. In other words, since both expert and non-expert reports are shared with other users in public webspace for the purpose of alerting them, our experiment did not violate their intended use. Then, we believe there is no ethical issue because we did not take any actions that directly harmed users (e.g., actions on victims' email addresses or Twitter accounts).

We used common open source tools to collect data from Twitter at scale and send requests accordingly. We conducted the experiments according to the best practices of related research on Twitter's usage guidelines, minimizing the influence on the platform. In this experiment, we sent only 40 requests to Twitter (20 Security Keywords + 20 Co-occurrence keywords) per hour in English and Japanese. Therefore, we believe that the availability of the platform was unaffected.

5.9 Related Work

We describe the related research on identifying malicious tweets and generating threat intelligence from Twitter.

Identification of "Malicious" Tweets. Numerous studies [53, 155, 71, 54, 67] have analyzed phishing attacks that direct users to external malicious sites from Twitter. Gao et al. proposed a system that can detect malicious posts in real time using features common to Twitter and Facebook, such as user connections and the number of characters in a post [52]. *These studies analyze only malicious tweets (i.e., those distributed by attackers with malicious intent). However, our study extracts benign tweets (i.e., shared by users with good intentions), and the information in the benign tweets, such as URLs or domain names, is phishing information; thus, the analysis targets are completely different.*

Threat Intelligence Extraction from Twitter. Research on threat intelligence generation using Twitter information has been conducted from various perspectives [101, 156, 102, 103, 104]. Shin et al. proposed a system to extract four types of information from a text on Twitter and external blogs: URLs, domain names, IP addresses, and hash values related to cyberattacks [103]. It has been demonstrated that the proposed system can detect threats, especially malware-related threats, earlier than other threat intelligence systems. Roy et al. focused on defanging and phishing attack-related hashtag strings, extracted information about phishing attacks from Twitter, and analyzed the characteristics of the accounts posting information [128]. It has been shown that information that interacts with other accounts, such as replies and retweets to the information posted on Twitter, is reflected more quickly in the block list. *Unlike our studies, tweets were collected from security experts by account names or limited keywords; thus, only limited information on Twitter was analyzed.*

5.10 Conclusion

This paper proposed CrowdCanary, a system that harvests phishing information from tweets of users who have discovered or encountered phishing attacks. The results suggest that reports from infrequent contributors (i.e., non-experts) contain a lot of valuable information for countering phishing attacks that is not included in the information posted by security experts. In addition, we identified tendencies in the domain names and hosting providers to which phishing sites were actually deployed, and indicated characteristics that are useful for detecting new phishing sites. Since this research showed the usefulness of information about new observation points on Twitter, we are ready to operate CrowdCanary in the future and to provide the data obtained to the national CSIRTs. We hope that the findings of this paper will be useful for future researches and countermeasure developments. We plan to share anonymized sample datasets with interested researchers upon request at `https://crowdcanary.github.io/`.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

With the spread of UGC, it has become easier for anyone to disseminate information in today's society. Because UGC allows individuals and communities to freely communicate their opinions and knowledge, the expertise and credibility of information providers are often uncertain, and there is a risk of spreading misinformation and prejudice. On the other hand, by selecting information and making good use of UGC, individuals can quickly obtain useful information of interest to them, and companies can use customer contributions to improve their products and services. This thesis investigates and analyzes the impact of these UGC in cybersecurity.

In Chapter 2, we categorized the platforms into five types based on the characteristics of UGC and organized them with actual services. We also discussed what kind of security threats may occur on those platforms and what kind of approaches have been analyzed and investigated in existing studies. Then, in order to investigate the impact of UGC on cybersecurity, we summarized the current situation that has not been clarified by existing studies and the issues that arise in clarifying them, and discussed what we will address in this thesis.

In Chapter 3, we investigated the vulnerabilities hidden in the content shared by ordinary users on the Q&A website. For this purpose, we proposed a method to identify the same code by calculating the similarity between code snippets, which are fragments of source code that exist on the web, and bytecode in apps. Using the proposed method, we conducted a investigation of the actual situation by matching the source code on Stack Overflow, a website used by many users when developing applications, with the code in the Android application. As a result, we found that the vulnerabilities in some Android applications used by a large number of users were caused by the source code. Developers were not able to select and discard appropriate information, and UGC had a negative impact on the actual apps.

In Chapter 4, we investigated the activity of fake accounts created by attackers on multiple UGC platforms. To this end, we defined an attack that directs a large number of users to a malicious site in the context of a notable event, and proposed a method to detect such attacks with high accuracy. Using the proposed method, we conducted a comprehensive study of attacks across four prominent UGC platforms: Twitter, Facebook, YouTube, and Reddit. The results revealed that the same group of attackers directs users from multiple UGC platforms to malicious sites for social engineering attacks targeting prominent events such as the World Cup. UGC by attackers on prominent UGC platforms had a negative impact on a large number of users.

In Chapter 5, we evaluated the usefulness of information about phishing attacks shared on a social networking service. For this purpose, we proposed a system to extract reports of attacks shared by well-meaning users in addition to security experts from a large set of tweets. The proposed system detected a large number of threat information on Twitter, and it was found that most of these information were shared by well-meaning users. The system was also able to extract the information more quickly than existing anti-phishing technologies.

6.2 The Future of UGC Platforms

UGC platforms are having a major impact on the way information is distributed. These platforms have become places where the general public becomes a source of information and where a wide range of perspectives and opinions are shared. Recent technological advances have had a significant impact on the shape and function of UGC platforms. In particular, the proliferation of mobile devices and advances in AI technology have played a major role in platform transitions. The proliferation of mobile devices has enabled users to create and share content anytime and anywhere, resulting in the rapid adoption of short text and video sharing platforms. Meanwhile, advances in AI technology have enabled personalized content recommendations based on user preferences, contributing to a better user experience.

However, these advancements bring with them new security threats. Due to the nature of the content that users can freely create and share, the spread of misinformation and false information is inevitable. Leakage of personal information, invasion of privacy, and posting of inappropriate content are also serious problems. To address these threats, platforms need to take appropriate measures, such as filtering content using AI and machine learning, monitoring user behavior, and strengthening systems for reporting and removing inappropriate content. At the same time, it is important for users themselves to learn how to defend themselves.

In addition, legal and regulatory measures are also required. For example, regulations such as Europe's GDPR (General Data Protection Regulation) have been enacted in many countries to strengthen the protection of personal information. However, on the other hand, this may also impede the free flow of information, thus an appropriate balance must be struck. This shows the tension between information accessibility and privacy protection. Therefore, the evolution of UGC platforms and countermeasures against the security threats associated with them is an issue that requires a multifaceted approach. In addition to technical measures, legal regulations and user education are also important factors. And these measures should aim to maintain an appropriate balance between the free distribution of information and the protection of personal privacy.

6.3 Future Work

This thesis investigated and analyzed the impact of UGC on cybersecurity. We investigated several scenarios in which cybersecurity threats could be generated by UGC in the real world, and found that it affects a large number of users. We also evaluated the use of cybersecurity countermeasure technologies for UGC and found that UGC can contribute to the development of future countermeasure technologies for attacks, as it can extract a range of information that is not covered by existing technologies. Based on this research, we would like to collaborate with platform operators to develop countermeasures against malicious activities of attackers on UGC platforms, which are currently insufficient. Furthermore, we would like to contribute to the development of countermeasure technologies in cooperation with security vendors by routinely collecting information necessary to incorporate into actual countermeasure technologies.

Acknowledgements

First, I would like to express my sincere gratitude to my supervisor, Professor Tsutomu Matsumoto, for his tremendous support, suggestions, and encouragement throughout my research. In particular, I learned a great deal from the advice from a broad perspective, looking at the overall research and the future. His insights and valuable suggestions on my research have been a great encouragement and inspiration to me in my research pursuits. I am extremely grateful to Professor Katsunari Yoshioka for teaching me how to conduct research and write academic papers. I am also grateful for the support and guidance I have received not only in research but also in job hunting and daily life over a total of six years since I was assigned to the laboratory as an undergraduate student. Without his help, I would not have been able to complete my dissertation. I also gratefully acknowledge Professor Junji Shikata whose insightful comments and advice significantly contributed to improving my dissertation. I appreciate many comments from the viewpoint of cryptology, which will lead to the further development of my research. I would like to thank Professor Tatsunori Mori and Associate Professor Shinichi Shirakawa for serving on my dissertation committee. Thank you very much for the valuable feedback on my research from the perspective of an expert in two different research fields, natural language processing and machine learning.

Next, I would also like to express my sincere gratitude to Dr. Daiki Chiba, Dr. Takashi Koide, Dr. Fumihiro Kanai, Dr. Yuta Takata, Mr. Naoki Fukushi, Dr. Mitsuaki Akiyama, Dr. Takeshi Yagi, Mr. Takeo Hariu and my colleagues at my former organization NTT Secure Platform Laboratories and NTT Security (Japan) KK for fruitful discussions, insightful comments, and a beneficial work environment. I am grateful to Dr. Daiki Chiba for being my mentor since I joined NTT Secure Platform Laboratories, giving me advice not only in my research but also in my company life. Furthermore, I would like to thank him for leading me as a team leader after I moved to NTT Security (Japan) KK. Likewise, I am thankful to Dr. Takashi Koide for supporting my doctoral course from a close perspective as a senior student from the same Yoshioka Laboratory at Yokohama National University. I am also indebted to members of the Matsumoto Laboratory, especially Ms. Mio Narimatsu and Ms. Tomoko Ishidate. The assistance with paperwork for the thesis submission and with materials and preparation for the dissertation hearings was very helpful. I am also grateful to Dr. Rui Tanabe, Dr. Akira Fujita, and Mr. Hiroshi Mori, my seniors in the laboratory, for their research instruction and mental support from the time I was assigned to the laboratory, which continues to be very supportive. I would like to thank all my friends who accompanied me to play games or go out for drinks when I was tired from my research.

Finally, I would like to thank my family. Thanks to the support of my older brothers (Mr. Fumiki, Mr. Yuki, Mr. Masaki, and Mr. Naoki), I was able to complete my doctoral course and I am very thankful. My parents, Mr. Goro and Ms. Fujiko, have given me a lot of trouble since I was preparing for the university entrance examination, but they have taken care of me and I was able to graduate from the doctoral course without any problems. Thank you very much. The person I would like to thank the most is my wife, Ms. Miho, who has been closest to me throughout my bachelor's thesis, master's thesis, and doctoral dissertation, and without her support, I could not have completed them all. Also, thanks to the presence of my unborn son, Mr. Haruto, I was able to do my best every day, thank you very much. I appreciate that my beloved cats, Mr. Ren and Ms. Kiki, gave me a lot of comfort by sleeping and playing with me every day. I am grateful to everyone who got involved.

List of Publications

Reviewed Papers in Journals

- **Hiroki Nakano**, Daiki Chiba, Takashi Koide, Naoki Fukushi, Takeshi Yagi, Takeo Hariu, Katsunari Yoshioka and Tsutomu Matsumoto, "Understanding Phishing Reports from Experts and Non-experts on Twitter," IEICE Transactions on Information and Systems, Vol.E107-D, No.17, 2024.
- Naoki Fukushi, Takashi Koide, Daiki Chiba, **Hiroki Nakano** and Mitsuaki Akiyama, "Understanding Security Risks of Ad-Based URL Shortening Services Caused by Users' Behaviors," Journal of Information Processing, Vol.30, 2022.
- **Hiroki Nakano**, Daiki Chiba, Takashi Koide, Mitsuaki Akiyama, Katsunari Yoshioka and Tsutomu Matsumoto, "Exploring Event-synced Navigation Attacks across User-generated Content Platforms in the Wild," Journal of Information Processing, Vol.30, 2022.
- **Hiroki Nakano**, Fumihiro Kanei, Yuta Takata, Mitsuaki Akiyama and Katsunari Yoshioka, "Towards Finding Code Snippets on a Question and Answer Website Causing Mobile App Vulnerabilities," IEICE Transactions on Information and Systems, Vol.E101-D, No.11, 2018.

Reviewed Papers in International Conference Proceedings

- Naoki Fukushi, Toshiki Shibahara, **Hiroki Nakano**, Takashi Koide and Daiki Chiba "Noisy Label Detection for Multi-labeled Malware," IEEE Consumer Communications & Networking Conference (CCNC 2024), 2024.
- Takashi Koide, Naoki Fukushi, **Hiroki Nakano** and Daiki Chiba "PhishReplicant: A Language Model-based Approach to Detect Generated Squatting Domain Names," The 39th Annual Computer Security Applications Conference (ACSAC 2023), 2023.
- **Hiroki Nakano**, Daiki Chiba, Takashi Koide, Naoki Fukushi, Takeshi Yagi, Takeo Hariu, Katsunari Yoshioka and Tsutomu Matsumoto, "Canary in Twitter Mine: Collecting Phishing Reports from Experts and Non-experts," Proc. The 18th International Conference on Availability, Reliability and Security (ARES 2023), 2023.
- Naoki Fukushi, Takashi Koide, Daiki Chiba, **Hiroki Nakano** and Mitsuaki Akiyama, "Analyzing Security Risks of Ad-Based URL Shortening Services Caused by Users ’ Behaviors," The 17th EAI International Conference, SecureComm 2021 (SecureComm 2021), 2021.

- **Hiroki Nakano**, Daiki Chiba, Takashi Koide and Mitsuaki Akiyama, “Detecting Event-synced Navigation Attacks across User-generated Content Platforms,” IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC 2021), 2021.

- Yosuke Kikuchi, Hiroshi Mori, **Hiroki Nakano**, Katsunari Yoshioka, Tsutomu Matsumoto and Michel van Eeten, “Evaluating Malware Mitigation by Android Market Operators,” The 9th USENIX Workshop on Cyber Security Experimentation and Test (USENIX CSET 2016), 2016.

Reviewed Poster in International Conference

- **Hiroki Nakano**, Daiki Chiba, Takashi Koide and Mitsuaki Akiyama, “Finding Social Engineering Attacks across User-generated Content Platforms,” The 22th USENIX Security Symposium (USENIX Security 2019), Poster session, 2019.

Technical Reports

- 森島 周太, **中野 弘樹**, 藤原 礼征, 吉岡 克成, 松本 勉, “多数のユーザの Web アクセスログから効率的に悪性サイトを抽出する手法,” 情報処理学会コンピュータセキュリティシンポジウム 2017, 2017.

- **中野 弘樹**, 金井 文宏, 秋山 満昭, 吉岡 克成, “Android アプリの脆弱性と Q&A サイト上のコードスニペットの関連性の分析,” 電子情報通信学会情報システムセキュリティ研究会 vol. IEICE-116, no. 522, pp. 171 - 176, 2017 **(情報システムセキュリティ研究賞)**.

- **中野 弘樹**, 楊 志勇, 森 博志, 吉岡 克成, 松本 勉, “パッキングサービスにより保護されたアプリの静的判別手法を用いた Android マーケットの実態調査,” 電子情報通信学会技術報告 ICSS2016-26, pp. 71-76, 2016.

References

[1] Number of worldwide social network users 2027 — statista. https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/.

[2] Anna Maria Jonsson and Henrik Ornebring. User-generated content and the news. *Journalism Practice*, 5(2):127–144, 2011.

[3] Oberiri Destiny Apuke and Bahiyah Omar. Fake news and COVID-19: modelling the predictors of fake news sharing among social media users. *Telematics Informatics*, 56:101475, 2021.

[4] Parth Patwa, Shivam Sharma, Srinivas PYKL, Vineeth Guptha, Gitanjali Kumari, Md. Shad Akhtar, Asif Ekbal, Amitava Das, and Tanmoy Chakraborty. Fighting an infodemic: COVID-19 fake news dataset. In Tanmoy Chakraborty, Kai Shu, H. Russell Bernard, Huan Liu, and Md. Shad Akhtar, editors, *Combating Online Hostile Posts in Regional Languages during Emergency Situation - First International Workshop, CONSTRAINT 2021, Collocated with AAAI 2021, Virtual Event, February 8, 2021, Revised Selected Papers*, volume 1402 of *Communications in Computer and Information Science*, pages 21–29. Springer, 2021.

[5] Katharine Sarikakis, Claudia Krug, and Joan Ramon Rodriguez-Amat. Defining authorship in user-generated content: Copyright struggles in *The Game of Thrones*. *New Media Soc.*, 19(4):542–559, 2017.

[6] Share a coke: The groundbreaking campaign from 'down under'. https://www.coca-colacompany.com/au/news/share-a-coke-how-the-groundbreaking-campaign-got-its-start-down-under.

[7] Apple unveils the best photos from the shot on iphone macro challenge - apple. https://www.apple.com/newsroom/2022/04/apple-unveils-the-best-photos-from-the-shot-on-iphone-macro-challenge/.

[8] Starbucks: White cup contest — storybox. https://storybox.io/blog/2017/7/25/tourism-queensland-best-job-in-the-world-z3csp.

[9] Kurt Thomas, Damon McCoy, Chris Grier, Alek Kolcz, and Vern Paxson. Trafficking fraudulent accounts: The role of the underground market in twitter spam and abuse. In Samuel T. King, editor, *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*, pages 195–210. USENIX Association, 2013.

[10] Md. Sazzadur Rahman, Ting-Kai Huang, Harsha V. Madhyastha, and Michalis Faloutsos. Efficient and scalable socware detection in online social networks. In Tadayoshi Kohno, editor, *Proceedings of the 21th USENIX Security Symposium, Bellevue, WA, USA, August 8-10, 2012*, pages 663–678. USENIX Association, 2012.

[11] Túlio C. Alberto, Johannes V. Lochter, and Tiago A. Almeida. Tubespam: Comment spam filtering on youtube. In Tao Li, Lukasz A. Kurgan, Vasile Palade, Randy Goebel, Andreas Holzinger, Karin Verspoor, and M. Arif Wani, editors, *14th IEEE International Conference on Machine Learning and Applications, ICMLA 2015, Miami, FL, USA, December 9-11, 2015*, pages 138–143. IEEE, 2015.

[12] Dhruv Kuchhal and Frank Li. A view into youtube view fraud. In Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini, editors, *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 555–563. ACM, 2022.

[13] Felix Fischer, Huang Xiao, Ching-yu Kao, Yannick Stachelscheid, Benjamin Johnson, Danial Razar, Paul Fawkesley, Nat Buckley, Konstantin Böttinger, Paul Muntean, and Jens Grossklags. Stack overflow considered helpful! deep learning security nudges towards stronger cryptography. In Nadia Heninger and Patrick Traynor, editors, *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, pages 339–356. USENIX Association, 2019.

[14] Eshwar Chandrasekharan, Mattia Samory, Shagun Jhaver, Hunter Charvat, Amy S. Bruckman, Cliff Lampe, Jacob Eisenstein, and Eric Gilbert. The internet's hidden rules: An empirical study of reddit norm violations at micro, meso, and macro scales. *Proc. ACM Hum. Comput. Interact.*, 2(CSCW):32:1–32:25, 2018.

[15] Malwarebytes Labs. Malvertising on blogspot: Scams, adult content, and exploit kits, 2023. https://www.malwarebytes.com/blog/news/2016/05/malvertising-on-blogspot-scams-adult-content-and-exploit-kits.

[16] Revengerat distributed via bit.ly, blogspot, and pastebin c2 infrastructure, 2023. https://www.bleepingcomputer.com/news/security/revengerat-distributed-via-bitly-blogspot-and-pastebin-c2-infrastructure/.

[17] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie S. Glance. What yelp fake review filter might be doing? In Emre Kiciman, Nicole B. Ellison, Bernie Hogan, Paul Resnick, and Ian Soboroff, editors, *Proceedings of the Seventh International Conference on Weblogs and Social Media, ICWSM 2013, Cambridge, Massachusetts, USA, July 8-11, 2013*. The AAAI Press, 2013.

[18] Himangshu Paul and Alexander G. Nikolaev. Fake review detection on online e-commerce platforms: a systematic literature review. *Data Min. Knowl. Discov.*, 35(5):1830–1881, 2021.

[19] Google Play. https://play.google.com/store.

[20] How to address WebView SSL Error Handler alerts in your apps. - Google Help. https://support.google.com/faqs/answer/7071387.

[21] How to resolve Insecure Hostnameverifier - Google Help. https://support.google.com/faqs/answer/7188426.

[22] How to address OpenSSL vulnerabilities in your apps - Google Help. https://support.google.com/faqs/answer/6376725.

[23] Vulnerable apps on Google Play put millions of users at risk of an attack. https://www.digitaltrends.com/mobile/google-play-open-port-attacks/.

[24] Over 400 Apps On Google's Play Store Found To Be Vulnerable To Open Port Attacks. http://www.ibtimes.com/over-400-apps-googles-play-store-found-be-vulnerable-open-port-attacks-2534541.

[25] Google launched a new bug bounty program to root out vulnerabilities in third-party apps on Google Play - The Verge. https://www.theverge.com/2017/10/22/16516670/google-play-security-rewards-program-vulnerabilities-bug-bounty.

[26] Stack Overflow - Where Developers Learn, Share, & Build Careers. http://stackoveflow.com/.

[27] Mobile Operating System Market Share Worldwide — Statcounter Global Stats. https://gs.statcounter.com/os-market-share/mobile/worldwide.

[28] Projects/OWASP Mobile Security Project - Top Ten Mobile Risks - OWASP. https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks.

[29] Github - Androbugs. https://github.com/AndroBugs/AndroBugs_Framework.

[30] Qihoo 360 Mobile Assistant. http://zhushou.360.cn/.

[31] Google play unofficial python api. https://github.com/egirault/googleplay-api.

[32] Sascha Fahl, Marian Harbach, Thomas Muders, Lars Baumgärtner, Bernd Freisleben, and Matthew Smith. Why eve and mallory love android: An analysis of android ssl (in) security. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 50–61. ACM, 2012.

[33] Manuel Egele, David Brumley, Yanick Fratantonio, and Christopher Kruegel. An empirical study of cryptographic misuse in android applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 73–84. ACM, 2013.

[34] Ryoya Furukawa, Tatsuya Nagai, Hiroshi Kumagai, Masaki Kamizono, Yoshiaki Shiraishi, Yasuhiro Takano, Masami Mohri, Yuji Hoshizawa, and Masakatu Morii. A vulnerability analysis of android applications from the view of third-party-libraries. In *IPSJ Journal*, volume 58, pages 1843–1855, 2017.

[35] Yasemin Acar, Michael Backes, Sascha Fahl, Doowon Kim, Michelle L Mazurek, and Christian Stransky. You get where you're looking for: The impact of information sources on code security. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 289–305. IEEE, 2016.

[36] Haoyu Wang, Zhe Liu, Yao Guo, Xiangqun Chen, Miao Zhang, Guoai Xu, and Jason Hong. An explorative study of the mobile app ecosystem from app developers' perspective. In *Proceedings of the 26th International*

Conference on World Wide Web, pages 163–172. International World Wide Web Conferences Steering Committee, 2017.

[37] Kai Chen, Peng Liu, and Yingjun Zhang. Achieving accuracy and scalability simultaneously in detecting application clones on android markets. In *Proceedings of the 36th International Conference on Software Engineering*, pages 175–186. ACM, 2014.

[38] Jonathan Crussell, Clint Gibler, and Hao Chen. Attack of the clones: Detecting cloned applications on android markets. In *ESORICS*, volume 12, pages 37–54. Springer, 2012.

[39] Jonathan Crussell, Clint Gibler, and Hao Chen. Andarwin: Scalable detection of semantically similar android applications. In *European Symposium on Research in Computer Security*, pages 182–199. Springer, 2013.

[40] Iman Keivanloo, Chanchal K Roy, and Juergen Rilling. Sebyte: Scalable clone and similarity search for bytecode. *Science of Computer Programming*, 95:426–444, 2014.

[41] Steve Hanna, Ling Huang, Edward Wu, Saung Li, Charles Chen, and Dawn Song. Juxtapp: A scalable system for detecting code reuse among android applications. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 62–81. Springer, 2012.

[42] Wu Zhou, Yajin Zhou, Michael Grace, Xuxian Jiang, and Shihong Zou. Fast, scalable detection of piggybacked mobile applications. In *Proceedings of the third ACM conference on Data and application security and privacy*, pages 185–196. ACM, 2013.

[43] Yuli Liu, Yiqun Liu, Ke Zhou, Min Zhang, and Shaoping Ma. Detecting collusive spamming activities in community question answering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1073–1082. International World Wide Web Conferences Steering Committee, 2017.

[44] Felix Fischer, Konstantin B^^c3^^b6ttinger, Huang Xiao, Christian Stransky, Yasemin Acar, Michael Backes, and Sascha Fahl. Stack overflow considered harmful? the impact of copy&paste on android application security. In *38th IEEE Symposium on Security and Privacy (S&P '17)*, 2017.

[45] DATAREPORTAL. Digital 2020: Global digital overview. `https://datareportal.com/reports/digital-2020-global-digital-overview`, 2020.

[46] John Seymour and Philip Tully. Weaponizing data science for social engineering: Automated e2e spear phishing on twitter. In *Black Hat USA 37*, pages 1–39, August 3-4 2016.

[47] Terry Nelms and Roberto Perdisci. Webwitness: Investigating, categorizing, and mitigating malware download paths. In *Proceedings of the 24th USENIX Security Symposium (USENIX Security)*, pages 1025–1040. USENIX Association, August 12-14 2015.

[48] Amin Kharraz, William Robertson, and Engin Kirda. Surveylance: Automatically detecting online survey scams. In *IEEE Symposium on Security and Privacy (SP)*, pages 70–86. IEEE, May 21-23 2018.

[49] Najmeh Miramirkhani, Oleksii Starov, and Nick Nikiforakis. Dial one for scam: A large-scale analysis of technical support scams. In *Proceedings of the 24th Network and Distributed System Security Symposium (NDSS)*. The Internet Society, February 26 - March 1 2017.

[50] Kaspersky. Spam and phishing in q1 2019 — securelist. https://securelist.com/spam-and-phishing-in-q1-2019/90795/, 2019.

[51] Fortinet. Free rugby world cup streaming service can be a foul play. https://www.fortinet.com/blog/threat-research/free-rugby-world-cup-streaming-foul-play.html, 2019.

[52] Hongyu Gao, Yan Chen, Kathy Lee, Diana Palsetia, and Alok N Choudhary. Towards online spam filtering in social networks. In *Proceedings of the 19th Annual Network and Distributed System Security Symposium (NDSS)*, pages 1–16. The Internet Society, February 5-8 2012.

[53] Sangho Lee and Jong Kim. Warningbird: Detecting suspicious urls in twitter stream. In *Proceedings of the 19th Network and Distributed System Security Symposium (NDSS)*. The Internet Society, February 5-8 2012.

[54] Kurt Thomas, Chris Grier, Dawn Song, and Vern Paxson. Suspended accounts in retrospect: an analysis of twitter spam. In *Proceedings of the 11th ACM SIGCOMM conference on Internet measurement (IMC)*, pages 243–258. ACM, July 23-29 2011.

[55] Chris Grier, Kurt Thomas, Vern Paxson, and Michael Zhang. @ spam: the underground on 140 characters or less. In *Proceedings of the 17th ACM conference on Computer and communications security (CCS)*, pages 27–37. ACM, October 4-8 2010.

[56] Kurt Thomas, Damon McCoy, and Chris Grier. Trafficking fraudulent accounts: The role of the underground market in twitter spam and abuse. In *Proceedings of the 22nd USENIX Security Symposium (USENIX Security)*, pages 195–210. USENIX Association, August 14-16 2013.

[57] Enrico Mariconti, Jeremiah Onaolapo, Syed Sharique Ahmad, Nicolas Nikiforou, Manuel Egele, Nick Nikiforakis, and Gianluca Stringhini. What's in a name?: Understanding profile name reuse on twitter. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*, pages 1161–1170. The International World Wide Web Conference Committee, April 3-7 2017.

[58] Payas Gupta, Roberto Perdisci, and Mustaque Ahamad. Towards measuring the role of phone numbers in twitter-advertised spam. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security (AsiaCCS)*, pages 285–296. ACM, June 4-8 2018.

[59] Hongyu Gao, Jun Hu, Christo Wilson, Zhichun Li, Yan Chen, and Ben Y Zhao. Detecting and characterizing social spam campaigns. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (IMC)*, pages 35–47. ACM, July 24-30 2010.

[60] Gianluca Stringhini, Pierre Mourlanne, Gregoire Jacob, Manuel Egele, Christopher Kruegel, and Giovanni Vigna. Evilcohort: detecting communities of malicious accounts on online services. In *Proceedings of the 24th*

USENIX Security Symposium (USENIX Security), pages 563–578. USENIX Association, August 12-14 2015.

[61] Bimal Viswanath, Muhammad Ahmad Bashir, Mark Crovella, Saikat Guha, and Krishna P Gummadi. Towards detecting anomalous user behavior in online social networks. In *Proceedings of the 23rd USENIX Security Symposium (USENIX Security)*, pages 223–238. USENIX Association, August 20-22 2014.

[62] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web (WWW)*, pages 591–600. ACM, April 26-30 2010.

[63] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. Detecting spammers on social networks. In *Proceedings of the 26th annual computer security applications conference (ACSAC)*, pages 1–9. ACM, December 6-7 2010.

[64] Chao Yang, Robert Harkreader, Jialong Zhang, Seungwon Shin, and Guofei Gu. Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on twitter. In *Proceedings of the 21st international conference on World Wide Web (WWW)*, pages 71–80. ACM, April 16-20 2012.

[65] Md Sazzadur Rahman, Ting-Kai Huang, Harsha V Madhyastha, and Michalis Faloutsos. Efficient and scalable socware detection in online social networks. In *Proceedings of the 21st USENIX Security Symposium (USENIX Security)*, pages 663–678. USENIX Association, August 8-10 2012.

[66] Hongyu Gao, Yi Yang, Kai Bu, Yan Chen, Doug Downey, Kathy Lee, and Alok Choudhary. Spam ain't as diverse as it seems: throttling osn spam with templates underneath. In *Proceedings of the 30th Annual Computer Security Applications Conference (ACSAC)*, pages 76–85. ACM, December 8-12 2014.

[67] Hiroki Nakano, Daiki Chiba, Takashi Koide, and Mitsuaki Akiyama. Detecting event-synced navigation attacks across user-generated content platforms. In *Intelligent and Resilient Computing for a Collaborative World 45th Anniversary Conference (COMPSAC)*. IEEE, July 12-16 2021.

[68] Luca Invernizzi, Paolo Milani Comparetti, Stefano Benvenuti, Christopher Kruegel, Marco Cova, and Giovanni Vigna. Evilseed: A guided approach to finding malicious web pages. In *IEEE Symposium on Security and Privacy (SP)*, pages 428–442. IEEE, May 20-23 2012.

[69] Bharat Srinivasan, Athanasios Kountouras, Najmeh Miramirkhani, Monjur Alam, Nick Nikiforakis, Manos Antonakakis, and Mustaque Ahamad. Exposing search and advertisement abuse tactics and infrastructure of technical support scammers. In *Proceedings of the 27th International Conference on World Wide Web (WWW)*, pages 319–328. The International World Wide Web Conference Committee, April 23-27 2018.

[70] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 528–540. Association for Computational Linguistics, June 1-6 2018.

[71] Srishti Gupta, Abhinav Khattar, Arpit Gogia, Ponnurangam Kumaraguru, and Tanmoy Chakraborty. Collective classification of spam campaigners on twitter: A hierarchical meta-path based approach. In *Proceedings of the 2018 World Wide Web Conference (WWW)*, pages 529–538. The International World Wide Web Conference Committee, April 23-27 2018.

[72] Nick Nikiforakis, Federico Maggi, Gianluca Stringhini, M Zubair Rafique, Wouter Joosen, Christopher Kruegel, Frank Piessens, Giovanni Vigna, and Stefano Zanero. Stranger danger: exploring the ecosystem of ad-based url shortening services. In *Proceedings of the 23rd international conference on World Wide Web (WWW)*, pages 51–62. ACM, April 7-11 2014.

[73] Google. Google trends. https://trends.google.com/trends/, 2020.

[74] Twitter. Twitter trends. https://help.twitter.com/ja/using-twitter/twitter-trending-faqs, 2020.

[75] Alexa Internet, Inc. Alexa - top sites. https://www.alexa.com/topsites, 2020.

[76] Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. Simple unsupervised keyphrase extraction using sentence embeddings. In *Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL)*, pages 221–229. Association for Computational Linguistics, October 31 - November 1 2018.

[77] Jean-Paul van Brakel. algorithm - peak signal detection in realtime timeseries data - stack overflow. https://stackoverflow.com/questions/22583391/peak-signal-detection-in-realtime-timeseries-data, 2019.

[78] M Zubair Rafique, Tom Van Goethem, Wouter Joosen, Christophe Huygens, and Nick Nikiforakis. It's free for a reason: Exploring the ecosystem of free live streaming services. In *Proceedings of the 23rd Network and Distributed System Security Symposium (NDSS)*, pages 1–15. The Internet Society, February 23-26 2016.

[79] Farsight Security Inc. Dnsdb. https://www.dnsdb.info/, 2020.

[80] VirusTotal. Virustotal. https://www.virustotal.com/, 2020.

[81] Duo Security. Security researchers partner with chrome to take down browser extension fraud network affecting millions of users. — duo security. https://duo.com/labs/research/crxcavator-malvertising-2020, 2020.

[82] Google. Google forms: Free online surveys for personal use. https://www.google.com/forms/about/, 2021.

[83] Momentive. Surveymonkey: The world ' s most popular free online survey tool. https://www.surveymonkey.com/, 2021.

[84] Karthika Subramani, Xingzi Yuan, Omid Setayeshfar, Phani Vadrevu, Kyu Hyung Lee, and Roberto Perdisci. When push comes to ads: Measuring the rise of (malicious) push advertising. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (IMC)*, New York, NY, USA, 2020.

[85] Twitter. Twitter rules enforcement. https://transparency.twitter.com/en/twitter-rules-enforcement.html, 2019.

[86] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. Design and evaluation of a real-time url spam filtering service. In *IEEE Symposium on Security and Privacy (SP)*, pages 447–462. IEEE, May 22-25 2011.

[87] Mingxuan Liu, Yiming Zhang, Baojun Liu, Zhou Li, Haixin Duan, and Donghong Sun. Detecting and characterizing sms spearphishing attacks. In *Proceedings of the 37th Annual Computer Security Applications Conference (ACSAC)*, pages 930–943. Association for Computing Machinery, 2021.

[88] Bradley Reaves, Nolen Scaife, Dave Tian, Logan Blue, Patrick Traynor, and Kevin RB Butler. Sending out an sms: Characterizing the security of the sms ecosystem with public gateways. In *Proceedings of the 37th IEEE Symposium on Security and Privacy (SP)*, pages 339–356. IEEE, December 6-10 2016.

[89] SafetyDetectives. 11 facts + stats on smishing (sms phishing) in 2021, 2021. https://www.safetydetectives.com/blog/what-is-smishing-sms-phishing-facts/.

[90] Bharat Srinivasan, Payas Gupta, Manos Antonakakis, and Mustaque Ahamad. Understanding cross-channel abuse with sms-spam support infrastructure attribution. In *Proceedings of the 21th European Symposium on Research in Computer Security (ESORICS)*, pages 3–26. Springer, September 28-30 2016.

[91] Proofpoint Inc. Smishing reports increase nearly 700% in first six months of this year, 2021. https://news.sky.com/story/smishing-reports-increase-nearly-700-in-first-six-months-of-this-year-12407504.

[92] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. Data breaches, phishing, or malware? understanding the risks of stolen credentials. In *Proceedings of the Conference on Computer and Communications Security (CCS)*, pages 1421–1434, October 30 - November 3 2017.

[93] Yun Lin, Ruofan Liu, Dinil Mon Divakaran, Jun Yang Ng, Qing Zhou Chan, Yiwen Lu, Yuxuan Si, Fan Zhang, and Jin Song Dong. Phishpedia: A hybrid deep learning based approach to visually identify phishing webpages. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security)*, pages 3793–3810, August 11-13 2021.

[94] Grant Ho, Asaf Cidon, Lior Gavish, Marco Schweighauser, Vern Paxson, Stefan Savage, Geoffrey M. Voelker, and David A. Wagner. Detecting and characterizing lateral phishing at scale. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security)*, pages 1273–1290, August 14-16 2019.

[95] Doowon Kim, Haehyun Cho, Yonghwi Kwon, Adam Doupé, Sooel Son, Gail-Joon Ahn, and Tudor Dumitras. Security analysis on practices of certificate authorities in the https phishing ecosystem. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (ASIACCS)*, pages 407–420, June 7-11 2021.

[96] Google. Google safe browsing, 2022. https://safebrowsing.google.com/.

[97] Microsoft. Microsoft defender smartscreen, 2022. https://docs.microsoft.com/en-us/windows/security/threat-protection/microsoft-defender-smartscreen/microsoft-defender-smartscreen-overview.

[98] PhishTank. Phishtank, 2022. https://www.phishtank.com/.

[99] OpenPhish. Openphish, 2022. https://openphish.com.

[100] SecurityTrails. urlscan.io, 2022. https://urlscan.io/.

[101] Fernando Alves, Ambrose Andongabo, Ilir Gashi, Pedro M Ferreira, and Alysson Bessani. Follow the blue bird: A study on threat data published on twitter. In *Proceedings of the 25th European Symposium on Research in Computer Security (ESORICS)*, pages 217–236. Springer, September 14-18 2020.

[102] Carl Sabottke, Octavian Suciu, and Tudor Dumitras. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In *Proceedings of the 24th USENIX Security Symposium (USENIX Security)*, pages 1041–1056. USENIX Association, August 12-14 2015.

[103] Hyejin Shin, WooChul Shim, Saebom Kim, Sol Lee, Yong Goo Kang, and Yong Ho Hwang. #twiti: Social listening for threat intelligence. In *Proceedings of the Web Conference 2021 (WWW)*, pages 92–104. ACM, April 12-16 2021.

[104] Hyejin Shin, WooChul Shim, Jiin Moon, Jae Woo Seo, Sol Lee, and Yong Ho Hwang. Cybersecurity event detection with new and re-emerging words. In *Proceedings of the 15th on Asia Conference on Computer and Communications Security (ASIACCS)*, pages 665–678. ACM, October 5-9 2020.

[105] Siyuan Tang, Xianghang Mi, Ying Li, XiaoFeng Wang, and Kai Chen. Clues in tweets: Twitter-guided discovery and analysis of sms spam. In *Proceedings of the Conference on Computer and Communications Security (CCS)*, pages 2751–2764, November 7 - 11 2022.

[106] Ryu Saeki, Leo Kitayama, Jun Koga, Makoto Shimizu, and Kazumasa Oida. Smishing strategy dynamics and evolving botnet activities in japan. *IEEE Access*, 10:114869–114884, 2022.

[107] NIST. National vulnerability database, 2021. https://nvd.nist.gov/.

[108] WeLiveSecurity. Why do we fall for sms phishing scams so easily? — welivesecurity, 2021. https://www.welivesecurity.com/2021/01/22/why-do-we-fall-sms-phishing-scams-so-easily/.

[109] Twitter IOC Hunter. Twitter ioc hunter, 2022. http://tweettioc.com/.

[110] Hiroki Nakano, Daiki Chiba, Takashi Koide, Naoki Fukushi, Takeshi Yagi, Takeo Hariu, Katsunari Yoshioka, and Tsutomu Matsumoto. Canary in twitter mine: Collecting phishing reports from experts and non-experts. In *Proceedings of the 18th International Conference on Availability, Reliability and Security, ARES 2023, Benevento, Italy, 29 August 2023- 1 September 2023*, pages 6:1–6:12. ACM, 2023.

[111] Statista. Twitter: most-used languages 2013 — statista, 2023. https://www.statista.com/statistics/267129/most-used-languages-on-twitter/.

[112] Twitter. Search api — twitter api — docs — twitter developer platform, 2023. https://developer.twitter.com/en/docs/twitter-api/enterprise/search-api/overview.

[113] Twitter. Compliance firehose api — twitter api — docs — twitter developer platform, 2023. https://developer.twitter.com/en/docs/twitter-api/enterprise/compliance-firehose-api/overview.

[114] Twitter. Decahose api — twitter api — docs — twitter developer platform, 2023. https://developer.twitter.com/en/docs/twitter-api/enterprise/decahose-api/overview/decahose.

[115] Adam Oest, Penghui Zhang, Brad Wardman, Eric Nunes, Jakub Burgis, Ali Zand, Kurt Thomas, Adam Doupé, and Gail-Joon Ahn. Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale. In *Proceedings of the 29th USENIX Security Symposium (USENIX Security)*, pages 361–377. USENIX Association, August 12-14 2020.

[116] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 1638–1649, August 20-26 2018.

[117] Megagon Labs. megagonlabs/transformers-ud-japanese-electra-base-ginza hugging face, 2021. https://huggingface.co/megagonlabs/transformers-ud-japanese-electra-base-ginza.

[118] Glenn Jocher et al. ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements, 2020.

[119] Tesseract OCR. Tesseract ocr, 2022. https://github.com/tesseract-ocr/tesseract.

[120] IETF Tools. RFC 3986, 2005. https://datatracker.ietf.org/doc/html/rfc3986.

[121] IETF Tools. RFC 1035, 1987. https://datatracker.ietf.org/doc/html/rfc1035.

[122] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proceedings of the 26th Network and Distributed System Security Symposium (NDSS)*. The Internet Society, February 24-27 2019.

[123] PeterDaveHello. Url shorteners, 2022. https://github.com/PeterDaveHello/url-shorteners.

[124] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 6105–6114. PMLR, June 09-15 2019.

[125] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristin" Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 4171–4186. Association for Computational Linguistics, June 3-5 2019.

[126] Eoghan Keany. Borutashap : A wrapper feature selection method which combines the boruta feature selection algorithm with shapley values., 2020.

[127] Miron B. Kursa and Witold R. Rudnicki. Feature selection with the boruta package. *Journal of Statistical Software*, 36(11):1–13, 2010.

[128] Sayak Saha Roy, Unique Karanjit, and Shirin Nilizadeh. Evaluating the effectiveness of phishing reports on twitter. In *Proceedings of the APWG Symposium on Electronic Crime Research (eCrime)*, December 1-3 2021.

[129] The Spamhaus Project. The top 10 most abused tlds, 2022. https://www.spamhaus.org/statistics/tlds/.

[130] Haikel Alhichri, Asma S. Alswayed, Yakoub Bazi, Nassim Ammour, and Naif A. Alajlan. Classification of remote sensing images using efficientnet-b3 cnn model with attention. *IEEE Access*, 9:14078–14094, 2021.

[131] Goncalo Marques, Deevyankar Agarwal, and Isabel de la Torre Diez. Automated medical diagnosis of covid-19 through efficientnet convolutional neural network. *Applied Soft Computing*, 96:106691, 2020.

[132] Per Christian Hansen. The truncatedsvd as a method for regularization. *BIT Numerical Mathematics*, 1987.

[133] Ashutosh Adhikari et al. Docbert: Bert for document classification, 2019.

[134] Fangxiaoyu Feng et al. Language-agnostic bert sentence embedding, 2020.

[135] Rodrigo Nogueira et al. Passage re-ranking with bert, 2019.

[136] vladkens. Twitter api scrapper with authorization support., 2023. https://github.com/vladkens/twscrape.

[137] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[138] Spotify. Luigi is a python module that helps you build complex pipelines of batch jobs., 2023. https://github.com/spotify/luigi.

[139] opmusic. Spamhunter_dataset, 2023. https://github.com/opmusic/SpamHunter_dataset/blob/main/sms_spam_urls/tweet_sms_url_latest.txt.

[140] Takashi Koide, Daiki Chiba, and Mitsuaki Akiyama. To get lost is to learn the way: Automatically collecting multi-step social engineering attacks on the web. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security (ASIACCS)*, pages 394–408. ACM, October 5-9 2020.

[141] Peng Peng, Limin Yang, Linhai Song, and Gang Wang. Opening the blackbox of virustotal: Analyzing online phishing scan engines. In *Proceedings of the Internet Measurement Conference (IMC)*, pages 478–485. Association for Computing Machinery, October 21 - 23 2019.

[142] Ke Tian, Steve T. K. Jan, Hang Hu, Danfeng Yao, and Gang Wang. Needle in a haystack: Tracking down elite phishing domains in the wild. In *Proceedings of the Internet Measurement Conference (IMC)*, pages 429–442. Association for Computing Machinery, October 31 - November 2 2018.

[143] Ziyun Zhu and Tudor Dumitras. Chainsmith: Automatically learning the semantics of malicious campaigns by mining threat intelligence reports. In *Proceedings of the 3rd IEEE European Symposium on Security and Privacy (EuroSP)*, pages 458–472. IEEE, April 24-26 2018.

[144] Cisco Umbrella. On the trail of malicious dynamic dns domains - cisco umbrella, 2023. https://umbrella.cisco.com/blog/on-the-trail-of-malicious-dynamic-dns-domains.

[145] dynamic.domains. 25 dynamic dns (ddns) providers - dynamic.domains, 2022. https://dynamic.domains/dynamic-dns/providers-list/default.aspx.

[146] Penghui Zhang, Adam Oest, Haehyun Cho, Zhibo Sun, RC Johnson, Brad Wardman, Shaown Sarker, Alexandros Kapravelos, Tiffany Bao, Ruoyu Wang, et al. Crawlphish: Large-scale analysis of client-side cloaking techniques in phishing. In *Proceedings of the 42nd IEEE Symposium on Security and Privacy (SP)*, pages 1109–1124. IEEE, May 24-27 2021.

[147] Terry Nelms, Roberto Perdisci, Manos Antonakakis, and Mustaque Ahamad. WebWitness: Investigating, categorizing, and mitigating malware download paths. In *Proceedings of the 24th USENIX Security Symposium (USENIX Security)*, pages 1025–1040. USENIX Association, August 12-14 2015.

[148] Anti-Phishing Working Group. Unifying the global response to cybercrime, 2022. https://apwg.org/.

[149] MaxMind. Geoip2 databases, 2023. https://www.maxmind.com/en/geoip2-databases.

[150] Chao Yang, Robert Harkreader, Jialong Zhang, Seungwon Shin, and Guofei Gu. Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on twitter. In *Proceedings of the 21st International Conference on World Wide Web (WWW)*, pages 71–80. The International World Wide Web Conference Committee, April 16-20 2012.

[151] Qian Cui, Guy-Vincent Jourdan, Gregor V. Bochmann, Russell Couturier, and Iosif-Viorel Onut. Tracking phishing attacks over time. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*, pages 667–676. The International World Wide Web Conference Committee, April 3-7 2017.

[152] Hugo Bijmans, Tim Booij, Anneke Schwedersky, Aria Nedgabat, and Rolf van Wegberg. Catching phishers by their bait: Investigating the dutch phishing landscape through phishing kit detection. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security)*, pages 3757–3774. USENIX Association, August 11-13 2021.

[153] Guang Xiang, Jason Hong, Carolyn P Rose, and Lorrie Cranor. Cantina+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information and System Security (TISSEC)*, 14(2):21, 2011.

[154] Twitter Dev. Twitter dev, 2023. https://twitter.com/TwitterDev/status/1615405842735714304.

[155] Anupama Aggarwal, Ashwin Rajadesingan, and Ponnurangam Kumaraguru. Phishari: Automatic realtime phishing detection on twitter. In *Proceedings of the APWG Symposium on Electronic Crime Research (eCrime)*, October 23-24 2012.

[156] Rupinder Paul Khandpur, Taoran Ji, Steve Jan, Gang Wang, Chang-Tien Lu, and Naren Ramakrishnan. Crowdsourcing cybersecurity: Cyber attack detection using social media. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM)*, pages 1049–1057, November 6-10 2017.