International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 10 Issue 2 ǁ 2022 ǁ PP. 27-30

Automated Emboss Of Mobile (Android) Phishing APPS via Graphical User Interface- DoS ATTACK

Achu Thomas Philip[1], Reshma Suku[2], Dr. Smita C Thomas[3]
[1] *M Tech Student, APJ Abdul Kalam Technological University, Kerala, India*
[2] *Asst. professor, Mount Zion College of Engineering, Kadammanitta, Kerala, India*
[3] *Professor, Mount Zion College of Engineering, Kadammanitta, Kerala, India*

Abstract - *-Phishing is a plan of attack to gain personal information for the intention of identity theft, usually by means of fraudulent E-mail. Attackers use emails, social media to trick victims into providing sensitive information/data or visiting malicious URL (Uniform Resource Locator) in the attempt to compromise their systems. Mobile phishing attacks, such as mimic mobile browser pages, masquerade as permissible applications by leveraging repackaging or clone techniques, have caused varied yet symbolic security concerns. Therefore, detection methods have been receiving increasing attention. In this paper, we propose a new attacking technique, called GUI-DoS (Squatting) attack, which can generate phishing apps automatically and effectively on the Android platform. This method embraces image processing and deep learning algorithms, to enable significant and large-scale attacks. We detect that a successful phishing attack requires two situations, page confusion and logic deception during attacks synthesis. Our experimental calculation reveal that existing phishing securities are less effective against such emergent attacks and may therefore stimulate more efficient detection techniques. To further reveal that our generated phishing apps can not only bypass existing detection techniques, but also deceive real users.*
***Key Words*: Phishing, Mobile Phishing, DoS (Squatting) Attack.**

Date of Submission: 14-02-2022 Date of acceptance: 28-02-2022

I. INTRODUCTION

Phishing is a technique of gathering sensitive information of a target such as username, password etc., by disguising as a trustworthy entity. In traditional phishing attacks, attackers send SMS (Short Message Service) or emails containing malicious links to redirect the browser to external phishing web pages or inducing download activities to install malicious applications on users' devices [1][7]. According to the FBI, phishing was the most reported cybercrime in 2020.

Moreover, phishing attacks are not necessarily sent in bulks but can be highly targeted, such as credential spear phishing [9] and whaling attacks [4].

1.1 MOBILE PHISHING

Mobile device-oriented phishing attack, SMS (Short Message Service) phishing uses text messaging to convince victims to disclose account credentials or install malware. Phishing attacks depend on more than simply sending an email to victims and hoping that they click on a malicious link or open a malicious link or attachment. Mobile phishing is becoming more prevalent and more difficult to spot. Instead, mobile phishing is their new approach and are targeting services like SMS, Whats App, Facebook, and fraudulent mobile apps. Cybercriminals are adept at using social engineering techniques to make their content appear authentic.

1.2 Squatting attack

A Squatting attack [10] is a form of denial-of-service (DoS) attack where a program interferes with another program through the use of shared synchronization objects. There exist several attack derivatives for different scenarios, such as typo-squatting attack, skill-squatting attack, and voice-squatting attack.

II. LITERATURE SURVEY

Web Phishing: - Gupta et al. [18] summarized that web phishing attacks have two traditional strategies: spoofed emails and fake websites. Spoofed emails induce users to click links in the email and redirect to a malicious website from untrusted servers to extract victims' information. Numerous approaches have been proposed to filter out phishing emails. Fette et al. [3][4] utilized machine learning to classify the spoofed emails with a high accuracy. CANTINA [7] proposed a content-based approach to detect phishing websites, based on the TF-IDF information retrieval algorithm. Pan et al. [5] examined anomalies in web pages (e.g., the discrepancy between a website's identity) to detect phishing web pages. Fu et al. [6] and Liu et al. [4][8] used visual similarity comparison to

distinguish phishing web pages. DOMAntiPhish [12] leveraged layout similarity information to distinguish malicious and benign web pages. Ma et al. [19] trained a predictive classifier based on the web URLs to identify phishing URLs. However, since attributes in mobile apps are different from those in web pages, these detection techniques are not applicable to mobile systems. In this, we focus on phishing attacks under mobile environments.
Mobile Phishing App-based phishing attack is a major problem on mobile devices [11], [13], [7], and phishing apps are one of the most popular types in malicious apps [5], [6], [7], [3], [2], [9]. Repackaged apps are the most useful technique to perform similarity attacks (spoofing attacks) for mobile phishing [10]. RESDROID [4] leverage new features extracted from core resources and source code to detect repackaged apps; however, phapps do not rely on repackaging techniques. Sun et al. [6] introduced that attackers can analyse the GUI code of the original apps, modify the corresponding layout code, and then add logical code to manipulate the original logic. However, developers can obfuscate or pack their apps to avoid repackaging malware attacks (e.g., repackaging phishing attacks). Meanwhile, this process heavily relies on the attacker's knowledge about the original app code. Bianchi et al. [8] extracted API call sequences via static code analysis to detect phishing apps, however, static analysis is limited to known attack vectors, and many similarity attacks don't require specific API calls. DROIDEAGLE [6] used the similarity of layout tree between official apps and third-party apps to detect mobile phishing apps. Marforio et al. [1], [2] leveraged personalized security indicators as a mechanism to avoid mobile phishing attacks. MOBIFISH (APPFISH) [7][3], [14] used OCR techniques to extract texts from the screenshot of a login interface. It identifies the identity from the extracted texts, and compares it with the actual identity from a remote server of mobile apps. If two identities are different, there is a warning presented to users. However, it has two shortcomings: (1) Many login pages do not contain app identities; (2) A whitelist of legitimate domains are required, in addition to a database of suspicious applications that needs to first be constructed and continuously updated. In this paper, we propose GUI-Squatting attacks; however, code obfuscations and packs will not affect the capability of our approach, and knowledge of the original app code is not essential. Moreover, our approach can bypass the state-of-the-art repackaging or clone detection techniques [20]. In addition to similarity attacks, window overlay and task hijacking are common mechanisms to execute mobile phishing attacks [1], [6], [7]. Although we do not focus on these two methods, our approach can also help generate the similar UI pages that can be leveraged by these two attacks. However, these two methods can be detected and mitigated by many cutting-edge detection techniques [2], [9], [10]. A recent defence solution has been proposed in [15] based on GUI-related APIs/permissions. WINDOWGUARD proposed a security model, Android Window Integrity [9] (AWI), to protect the system against all GUI attacks, including window overlay and task hijacking, The generated phapps are able to bypass all of these detection techniques successfully.

III. EXISTING SYSTEM

Many researchers have previously been carried out in this field. We have gathered the information from various such works and have profoundly reviewed them which has helped us in motivating our own methodologies in the process of making a more secure and accurate system.

Phishing, as a type of social engineering attack [15], [18], is often used to steal user information, such as login credentials. It occurs when an attacker masquerades as a trusted entity (resembling the original web page or application) [13]. Web phishing attacks date back to 1995 [17], but recently, attackers have shifted their attention to mobile devices [17]. Due to the small screen size and lack of identity indicators of URLs seen next to online web sites, mobile users have become more vulnerable to phishing attacks. On mobile devices, 81% of phishing attacks are carried out using phishing apps, SMS, or web pages [13].

Mobile oriented phishing attacks are classified into two strategies: (1) masquerade as original apps; or (2) hijack existing original apps. Mobile phishing attacks can be classified into three types based on the above two strategies.

Similarity attacks (spoofing attacks) analyse the GUI code of the original app and partially modify the GUI code. Attackers then add logic code to manipulate the original app logic [6]. For example, attackers can crack payment apps to bypass the payment functionality. Window overlay attacks render a window on top of mobile screen, either partially or completely (e.g., similar UI pages) overlapping the original app window [11], [12]. For example, attackers choose a particular time to render the phishing UI pages by monitoring the occurrence of the original app's login activity. This attack usually leverages the flaws of design mechanism in mobile Task hijacking attacks trick the system into modifying the app navigation behaviours or the tasks (back stacks) in the system [3], [6]. For example, the back button is popular with users because it allows users to navigate back through the history of activities. However, attackers may abuse the back button to mislead the user into a phishing activity .In short, attackers try to modify the tasks and back stack to execute phishing attacks.

IV. PROPOSED SYSTEM

In existing mobile phishing attack techniques: a successful phishing app requires two constrains: page confusion and logic deception. In this paper, we propose a new powerful Dos attack called "GUI-Squatting Attack"

based on fully automated generation of phishing UI pages and apps and novel base method. Moreover, our approach can generate similar UI pages for the phishing attacks mentioned above. The following differences make the GUI-Squatting attack more threatening than previous attacks. (1) Only the login page(s) of an app is needed and no other inputs are necessary, making a large-scale attack desirable, disregard of platform limitations. (2) No requirements of domain knowledge and conventional attack techniques (e.g., repackaging and clone techniques) make the result harder to detect. (3) It can handle a wide range of attacks due to the low cost of the generation process, and it can launch targeted attacks like credential spear phishing attacks [3]. Our generated phishing apps can successfully control every pixel of the screen and capture real users' credentials without raising the user's attention under practical GUI-Squatting attacks in the real world.

Our approach is implemented in Python 3, and leverages several open source libraries to automatically give arise to phishing application. Specifically, we use OPENCV [7]) and OCR techniques to extract components and their attributes (e.g., coordination positions, width, height, color, texts) from the screenshots of User interface pages. Meanwhile, we use Tesseract#makebox to excerpt the coordinate of each letter. To classify the types of segmented components within the User Interface screenshots, we adopt the CNN model in our model contains three convolutional layers, three pooling layers, and two fully-connected layers. Within the convolutional layer, we set the filter (screen) size as 3, the stride as 1, and padding size as 1. The same setting also applies to the pooling layer. For two fully-connected layers, both have 128 neurons. We implement our network with the Tensor flow framework written in Python. We generate the login GUI code for the given UI screenshot. For each component, we use two layout attributes (i.e., android:layout margin Left and android: layout marginTop) to identify their coordinates. In addition to the basic attribute settings, we also transfer attributes of the component to corresponding layout code (e.g., android: textColor, android:inputType). After implementing the UI login code, we implement 10 types of responses from Table 2 when interactive components are clicked; each component has a different response attached within the deception code. As for the response to login actions, we randomly choose one response to be attached to the "login" button. Our implementation runs on a 64-bit.Evaluate our approach in the following five aspects: (1) UI page similarity comparison between the User interface pages of the original apps and our generated phishing application; (2) UI page generation comparison between the state-of-the-art UI generation tools and our approach; (3) Performance of our CNN classification; (4) Ability to evade detection by the state-of-the-art anti-phishing techniques; (5) A human study to identify the power and impact of our phapps. We randomly collect 500 Android apps (250 financial apps and 250 social apps) from the top 100 financial and social categories from the Google Play Store, as the apps in these two categories are usually security- and privacy critical. All apps require users to login before use. These are the most famous apps (e.g., Facebook, Twitter) with over 1,000,000+ installs, mainly originating from USA, China, and European countries. We guarantee the representativeness of the selected original apps in terms of their number of installs and representative categories. Given the screenshots of login pages and icons of these apps, we generate the corresponding 50 phishing apps using our approach. The dataset of (50 original apps and 50 phishing application in total) is used to conduct the following experiments. Besides the 50 financial apps and social apps used in our experiments, in order to reduce the influence of randomness, we further select 20 apps that were downloaded from different times off the Google Play Store that also contain login pages to validate the similarity of our results. From the comparison of results, the corresponding generated UI pages of these 20 apps are also sufficiently similar (they achieve over 95% similarity on average in terms of mean absolute error (MAE) and mean squared error (MSE)) and can be used in the GUI-Squatting attack directly. More generated phishing UI pages can be found on our website [10].

LIMITAIONS

- Our approach does not fully Grasps the font family/color of the text extracted from the EditText component, causing a small visual difference if the app uses a special font family. Fortunately, according to the evaluation, users are insensitive of such differences.
- Since we develop components with normal attributes, such as a plain background of EditTexts, if the original app uses a colourful image (e.g., photos) as the background of EditTexts, we cannot develop a perfect copy of its UI page.
- As for targeted User Interface pages with smaller resolutions, we need to scale the component to an equivalent size to deploy the same phishing application on devices with larger intension.

V. CONCLUSION

In this Paper, we propose a novel approach to automatically generate platform-independent phishing application, to allow a powerful and large-scale phishing attack (Graphical User Interface -Squatting attack) on different categories of apps within 3 seconds. Our phishing application successfully masquerades as original apps without raising users' special attention in information effusion.

REFERENCES

[1]. (2015) Alleged 'Nazi' Android FBI ransomware mastermind arrested in Russia. [Online]. Available: https://www:forbes:com/

[2]. (2018) Android Packers. [Online]. Available: https://d3gpjj9d20n0p3:cloudfront:net/AndroidPackers Hacktivity:pdf
[3]. (2018) Canny Edge Detection. [Online]. Available: https: //docs:opencv:org/3:4/da/d22/tutorial py canny:html
[4]. (2018) Dilatation Edge. [Online]. Available: https://docs:opencv:org/2:4/doc/tutorials/imgproc/erosion dilatation/erosion dilatation:html
[5]. (2018) GDPR. [Online]. Available: https://www:tripwire:com/ solutions/compliance-solutions/gdpr/
[6]. (2018) One-way analysis of variance. [Online]. Available:https://en:wikipedia:org/wiki/One-way analysis of variance
[7]. (2018) Opencv. [Online]. Available: https://opencv:org/
[8]. (2018) Optical Character Recognition. [Online]. Available:https://en:wikipedia:org/wiki/Optical character recognition
[9]. (2018) Phishers leveraging GDPR-Themed scam emailsto steal users' information. [Online]. Available: https://securityboulevard:com
[10]. (2018) Squatting attack. [Online]. Available: https://en:wikipedia:org/wiki/Squatting attack
[11]. (2018) Tesseract. [Online]. Available: https://github:com/tesseract-ocr/tesseract
[12]. (2018) UIAutomator. [Online]. Available: https://developer:android:com/training/testing/ui-automator
[13]. D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of Android malware in your pocket." in NDSS, 2014.
[14]. T. Beltramelli, "pix2code: Generating code from a graphical user interface screenshot," arXiv preprint arXiv:1705.07962, 2017.